COMP5222 Group Project

Behavior-Driven Development

Rails Application with Cucumber

Qu Xiaofeng

09903198R

# What Is Behavior-Driven Development

According to the Wikipedia, behavior-driven development (BDD) is a software development process developed on test-driven development (TDD) in software engineering field. The traditional techniques and principles of TDD and thoughts from domain-driven design and object-oriented analysis and design are combined together. It provides software developers and business analysts with common tools and a common process to cooperate in software development process.

BDD is principally based on the concept that software development should be performed by both business side and technical side. The practice of BDD relies on the use of customized software tools to support this development process. This development environment is not only specifically developed for use in BDD projects, but also it can be used as a specialized form of the tool-chain that supports test-driven development. The central theme of BDD is the tools who automate the testing from specifications written with ubiquitous language.

The most important issues include:

1. The Acceptance Test is not written only by coder, but by the client. (features and scenarios)

2. The Acceptance Test can be written in natural language, but not in programming languages.

3. The definition of the product is consistent with the code, testable and reliable through the whole life cycle of the product.

## Definition of BDD

Behavior-driven development was inspired by the problems encountered by developers practice test-driven development.

* Where to start the whole development process
* What should be tested and what should not to be tested
* How much tests should be include in one test iteration
* What to call the tests
* How to understand why a test fails

The main problem the BDD concerns is the mechanism of unit testing and acceptance testing to solve above problem.

The main results are:

* The unit test name should be sentences starting with the word “should”
* The unit test should be written in order of business value
* Acceptance tests should be written using the standard agile framework of a [User story](http://en.wikipedia.org/wiki/User_story): "As a [role] I want [feature] so that [benefit]"
* Acceptance criteria should be written in terms of scenarios and implemented as classes: Given [initial context], when [event occurs], then [ensure some outcomes]

Based on above principles, the first BDD framework, JBehave, was developed by Dan North in 2003. Then it was ported to Ruby as a story-level BDD framework called RBehave which was integrated into the RSpec project. The story runner of RSpec was later replaced by Cucumber which is developed by Aslak Hellesøy. Cucumber was first developed to be merged into RSpec as a next generation story-runny module,

## Agile testing, the philosophy behind BDD

1. human

2. less is more

3. fast iteration

4. testable

5. refactor

## The concept of behavior-driven development

1. the outer and inner circles

2. cucumber and rspec

## Steps and processes of BDD

The main tool of Rails-oriented BDD is RSpec. It is a testing tool for the Ruby programming language. The BDD is its main field. The object of it is to make Test-Driven Development a productive and enjoyable experience.

According to the [rspec.info](http://rspec.info), the main features are

* A rich command line program (the rspec command)
* Textual descriptions of examples and groups (rspec-core)
* Flexible and customizable reporting
* Extensible expectation language (rspec-expectations)
* Built-in mocking/stubbing framework (rspec-mocks)

1. client spec

2. begin iter

3. write feature

4. falling the feature

5. pass the feature

6. refactor the code

7. end iter

## BDD software in different languages

See references from here [http://behaviordrivendevelopment.wikispaces.com/MoreTools](http://behaviordrivendevelopment.wikispaces.com/MoreTools).

+ ASSpec - ActionScript 3

+ Aero - PHP 5

+ Aubergine - .NET

+ BDoc - Extracting documentation from unit tests, supporting behavior driven development

+ BDD in Python - is core module doctest

+ Bumblebee - Extract documentation from JUnit tests with support for adding text, code-snippets, screenshots and more. Puts focus on the end-user.

+ beanSpec - Java

+ Behat - PHP implementation of the Gherkin Domain-specific language

+ Cedar - Objective C

+ CppSpec - C++

+ cfSpec - ColdFusion

+ CSpec - C

+ dSpec - Delphi

+ Concordion - a Java automated testing tool for BDD that uses plain English to describe behaviors.

+ Cucumber - Plain text + Ruby. Works against Java, .NET, Ruby, Flex or any web application via Watir or Selenium.

+ easyb - Groovy/Java

+ EasySpec - Groovy, usable in Java. Developer also working on Perception a tool for doing Context/Specification reporting for many different tools.

+ EXTasy - Behavior-driven framework for ExtJS interfaces. Written in python.

+ FitNesse - Java, .NET, C++, Delphi, Python, Ruby, Smalltalk, Perl. Now supports BDD directly with plain text tables and scenarios.

+ Freshen - Python - clone of the Cucumber BDD framework

+ GivWenZen - Java and FitNesse

+ GivWenZen for Flex and ActionScript3 - Flex cousin of Java GivWenZen

+ GSpec - Groovy

+ Igloo - C++

+ Instinct - Java

+ Jasmine - JavaScript - framework-independent BDD with easy CI integration

+ JavaStubs - Java - BDD framework supporting partial-mocking/method stubbing

+ JBee - Java

+ JBehave - Java - The first BDD framework, now at version 3.x

+ JDave - Java

+ JFXtras Test - JavaFX

+ JSpec - JavaScript - BDD framework independent, async support, multiple reporters (terminal, dom, server, console, etc.), Rhino support, over 50 matchers and much more

+ JSSpec - JavaScript

+ Kiwi - RSpec like BDD library for iOS

+ Lettuce - a Cucumber-like BDD tool for Python

+ Morelia viridis - Cucumber clone for Python

+ MSpec - .NET

+ NBehave - .NET

+ NSpec - .NET

+ NUnit - A TDD framework in .NET which can be used for BDD examples and scenarios

+ ObjectiveMatchy - iPhone - A Matcher System for iPhone development.

+ Pyccuracy - Behavior-driven framework in Python.

+ Pyhistorian - General purpose BDD Story Runner in Python (internal DSL, not plain-text)

+ PyCukes - Cucumber-like BDD tool built on top of Pyhistorian

+ Robot Framework - Generic keyword-driven test automation framework for acceptance level testing and acceptance test-driven development (ATDD) written in Python

+ RSpec - Ruby

+ Spock - Spock is a testing and specification framework for Java and Groovy

+ SSpec - SSpec is the BDD framework for Smalltalk (multiple dialects) created by Dave Astels

+ SpecFlow - SpecFlow is inspired by Cucumber and the community around it. Binding business requirements to .NET code

+ screw-unit - JavaScript

+ ScalaTest - Scala

+ specs - Scala

+ spec-cpp - C++

+ Spectacular - Open source BDD and ATDD tool incorporating several types of tests in a single document and introduces Executable Use Cases

+ Specter - Another implementation of BDD framework in .NET with focus on specification readability

+ StoryQ - .NET 3.5, can be integrated with NUnit to provide both specification readability and testing

+ TickSpec - Gherkin based framework supporting F# and C#

+ tspec - Groovy/Java (Thai syntax)

+ Tumbler - Java. Integrated with JUnit

+ Twist - Commercial Eclipse-based tool for creating executable specifications

+ Vows - JavaScript

+ XSpec - XPath, XSLT and XQuery

# Cucumber - A Ruby Based BDD Software

The former tool of cucumber is the rspec-story

## Specification based acceptance testing

The acceptance testing is performed by cucumber.

The acceptance report should be structured like this.

Application

has\_many Features

Feature

has\_a comment

has\_many Scenarios

Scenario

Given …

When …

Then …

## Features and scenarios

Structure of scenarios

## Gherkin, a language describing the specification

## Organizing and tagging of features

# An Example Project Using Cucumber

In recent years, the cloud computing is emerging as a main trend of current information industry. Though there are many new cloud service provider, a huge number of old fashioned applications still can not be used in cloud. The aim of this project is to migrate a simple image processing application into cloud. With this project, old fashioned image processing application is empowered with new cloud based flexibility.

## Requirements of a cloud based image processing website

The major requirements are listed below.

1. upload a image to this website, and view the uploaded image
2. process this image and preview the result image
3. download the result image

## The first prototype, image uploading and showing

## The second iteration, core image processing

## The third iteration, security and style issue

## The fourth iteration, refactoring

# Defects And Pitfalls of BDD

## How to write great features

## How to organize features

## BDD using different languages