Test Driven Development (beyond JUnit) with JBehave

Madison JUG

May 25, 2010

Overview

- ATDD and BDD
- JBehave Basics
- Code, Advanced Topics and Integrations
  - Multi-Tenant Spring Security Authentication
  - JBehave Advanced Topics
  - Integration with Spring, DbUnit, Cobertura
- JBehave / Selenium example
- FIT, easyb and Cuke4Duke comparisons

ATDD and BDD

- ATDD – Acceptance Test Driven Dev.
  - Pioneered with FIT/Fitnesse
  - Executable tests, requires Fixture
- BDD – Behavior Driven Development
  - Introduced in 2003 by Dan North
  - Early on, referred to as "TDD done well"
  - Focuses on behavior of the system versus testing of component in isolation from its dependencies
  - Uses the language of the business
- Agile 2009 conference
  - JBehave, easyb and Cuke4Duke introductions

Specifications to Stories

- Specifications
  - Code-level BDD
    - Focus on developers
  - “describe”, “before” and “it”
  - JBehave1, RBehave, easyb, RSpec, JDave
- Stories
  - Feature-level BDD
    - Focus on wider audience (Owner, BA, QA, Dev, QC)
    - Acceptance Tests / defines “done” for a feature
  - “Given, When, Then”
  - RSpec, Cucumber, JBehave2, easyb, NBehave
    - RSpec Story Runner was merged into RSpec
  - Plain-Text Stories
    - Readable by the business – no code!
    - Executable for development and regression testing

JBehave

- Java-based BDD Framework
  - Created by Dan North (in 2003) to compare BDD to TDD
  - Liz Keogh joined in 2004 – major early contributor
  - Mauro Talevi, Paul Hammant (Selenium) and Shane Duan
- Version 1 vs Version 2
  - Version 1.0 released Mar 2007
  - Version 1.0 had lots of stuff that got pulled out
  - Version 2.0 released Sep 2008
- Current version is 2.5 (2.5.7)
  - New builds about every week or 2
  - Minor releases about every 2-3 months
  - Version 3.0 is currently in the works (beta 7)
- JBehave and JBehave Web
  - Main component includes
    - the core library as well as Ant, Maven, Spring, Pico and Guice integrations
  - Web component includes
    - the Web Runner webapp that allows entering / running stories
    - a Web-based integration for various third-party frameworks
  - BSD-like license and requires Java 5

Stories

- Plain-text stories
  - Contains a narrative and multiple scenarios
- Narrative
  - Optional
  - In order to / As a / I Want to
- Scenario(s)
  - Given / When / Then / And
  - "And" goes with Given, When or Then
  - Can also do "when"s after "then"s
  - Given Scenarios (scenarios depend on others)
  - Examples (tables of data) / "scenario templates"
- Comments (!--)

Scenario

- Java class that maps to a story
  - There is a default name mapping
    - MyRockingScenario.java = my_rocking_scenario
- Scenario class is main entry point
  - Constructor sets up the
    - Configuration (lots of defaults)
    - Candidate Steps
      - What could implement the Given/When/Then statements
      - Lifecycle events / methods
- Scenario class is a JUnit3 TestCase
  - With one test method - testScenario
- In JBehave 3, this is now called Story

Steps

- Java class with method annotations
  - JBehave step annotations
    - @Given, @When, @Then
    - "And" maps to any of these based on context
  - JBehave lifecycle Annotations
    - @BeforeStory, @AfterStory, @BeforeScenario, @AfterScenario
- Step annotations take a regex
  - Captured values are converted to method parameters
    - Various converters, including automatic list handling
  - @Alias to map various text strings to 1 method
    - Makes for more readable stories (singular / plural)
  - @Named used to explicitly map values to parameters
- JBehave figures out best match for text
  - Given steps/methods at scenario construction
  - Story parsing matches them up

Sample Code

- "Kata" is used to describe an exercise in programming which helps hone your skills through practice and repetition
  - Small code examples for common ideas
  - Term is used more in the testing framework world
- We will be using a "Multi-Tenant User Auth" kata
  - System supports multiple organizations (tenants)
  - Each org has its own authentication policy
  - Each org has multiple users
  - Those users want to login (authenticate)
  - I made this up! It was the best example I had…
- Implementing this with Spring Security

Configuration

- Everything is configurable (powerful!)
- Everything has a default (easy!)
- Main Configuration
  - ScenarioDefiner
    - How to load stories – default is classpath resource
    - Sub-configuration for naming and parsing
  - ScenarioReporter
    - How to report events during execution – lots of options
  - PendingErrorStrategy, ErrorStrategy
    - How to handle failure and missing steps
  - StepCreator
    - How to match up CandidateSteps to actual Steps
  - KeyWords (non-english stories)
- Steps Configuration
  - StepPatternBuilder
    - How to build regex from text / parse parameters
  - StepMonitor - reporting but at a step level
  - Parameter Converters

JBehave / Selenium

- Will demonstrate Pico Ajax Email example
  - Not loading data for me…so tests will run & fail
- Steps class has a Selenium/Waiter in it
- Step Methods call the Selenium object
  - Looser coupling than FIT-Selenium Bridge
  - Much easier to code than your own fixture
- Maven used to start/stop jetty and selenium around integration-test phase
- Doesn’t need JBehave Web
  - Uses Selenium directly

Advanced JBehave

- Ant tasks and Maven plugin
  - Could not get Ant task to work with Cobertura
- Integration with Pico/Guice (and Spring)
  - See warning on next page
- Reporting options
  - Console, Text, HTML, XML in example
- Integration with TestNG – and/or – JUnit
  - Via use of annotations / ScenarioRunner
- Stepdoc
- Non-English keywords
- JBehave-JUnit-Monitor (separate project)
  - Couldn’t access svn site…looks promising though

Pros and Cons

- Pros
  - The only plain-text story / all Java solution
  - Easily integrates into existing processes / tools
  - Helps agile teams define “done” before a story is started
    - Define this early – JBehave handles pending work
    - Becomes part of Continuous Integration for regression testing
  - Can help lower cost of projects
    - Simple format for requirements / limited waste
    - QC can focus on exploratory testing since more is automated
  - Highly customizable
    - Could pull stories direct from Agile planning system
- Cons
  - My original cons are now all fixed in latest version
  - IoC integration requires non-constructor bootstrapping
    - JUnit classes are eagerly instantiated!
    - IoC bootstrap should happen as a JUnit @RunWith / @Rule
    - AbstractSpringScenario shows use of Spring integration

FIT, easyb and Cucumber

- FIT/Fitnesse is a Java-based ATDD framework
  - Base fixture code is difficult to understand and extend
  - Works with HTML tables rather than plain-text
    - Not all that readable by the business
    - GivWenZen fixture brings BDD to FIT (@DomainStep annotation)
- easyb is Groovy-based BDD framework
  - Does both specifications and scenarios
  - Code is embedded with the before/it/given/when/then
  - Not parameterized like JBehave annotations
- Cucumber is Ruby-based BDD framework
  - Introduced feature-level BDD and plain-text stories
  - Cuke4Duke is an extension for Java
    - "Steps" are written in Java (or any JVM language)
    - Uses the JBehave annotation/regex paradigm
    - Runs with JRuby
  - Has a richer syntax (gherkin) for story writing
  - Has better reporting options than JBehave (for now)
  - Cucumber can now use Steps that extend JBehave Steps!

Resources

- BDD
  - http://behaviour-driven.org/
  - http://dannorth.net/introducing-bdd
  - http://www.ryangreenhall.com/articles/bdd-by-example.html
- JBehave
  - http://jbehave.org/
  - http://blog.m.artins.net/acceptance-tests-with-jbehave-selenium-page-objects
  - http://blogs.mikeci.com/2010/05/06/continuous-testing-with-selenium-and-jbehave-using-page-objects/
- JBehave JUnit Monitor project
  - http://code.google.com/p/jbehave-junit-monitor/
- FIT / Fitnesse
  - http://fit.c2.com/
  - http://fitnesse.org/
- easyb
  - http://www.easyb.org/
- Cucumber
  - http://cukes.info/
- Cuke4Duke
  - http://wiki.github.com/aslakhellesoy/cuke4duke/
- “Bridging the Communication Gap” by Gojko Adzic

Questions?

- Contact Information

  Brian Repko

  brian.repko@learnthinkcode.com

  612-229-6779

  Slides and code are available at
  http://www.learnthinkcode.com