

Behavior-Driven Development with cfSpec

The Railo logo is written in a white, cursive, handwritten-style font. The letters are connected and flow together, with a prominent loop at the top of the 'R'.

Sean Corfield
Railo Technologies
cf.Objective() 2009

Who are you?

- You are a CFML developer
- You do some testing, possibly automated
- You want to improve how you specify, build and test your applications
- You may have asked yourself:
- "What's this Behavior-Driven Thingy then?"

Who am I?

- CTO, Railo Technologies US
- Fast Free Open Source CFML Engine
- Development / Support Consultancy
- Blogger, speaker, curmudgeon (FAQU)
- Eight years of CFML
 - Always been Java, always been OO!
- Long history of IT before that (grey hair!)

What will we cover?

- Testing - why is it hard work?
- Changing how we think about testing
- Starting with requirements
- Introducing cfSpec

Testing

Kenilo

Testing your code

- Who tests their code?
- Who manually tests their code?
- Who writes code to test their code?

Writing tests

- Typically mean unit testing
- Write small pieces of code to test operations on small pieces of code

Writing tests first

- Write the test first - so it fails
- Write the minimum code - so it passes
- Refactor code - so it still passes
- Rinse and repeat

Writing tests first

- This is Test-Driven Development (TDD)
 - Who went to Marc Esher's Thursday talk?
- a.k.a. Red Green Refactor
 - If you're at CFUnited, check out Adam Haskell's talk!

Unit testing is hard!

Kenite

Why is it hard?

- Trying to predict implementation
- Trying to test state and boundaries
- Written at a low level

Is TDD enough?

- Test-Driven Development is beneficial
- It gets you thinking about your design
- But
 - It's by programmers for programmers
 - It's often bottom-up instead of top-down
 - Think of an object, then write tests for it

Making it easier

Kenite

Using the right words

- What if we used business terms?
- What if we wrote tests that could be read like English?
- What if our specifications could be run?

Using the right words

- The idea is that the words you use influence the way you think about something.
- Neuro-Linguistic Programming

Requirements

Kenite

It's all behavior

- What are requirements?
- Specify how the system should **behave**

It's all behavior

- describe "Account, when first created"
 - it should "have a balance of \$0"
- describe "Account, with sufficient funds"
 - it should "allow cash withdrawal"
 - it should "allow transfer of funds"

Show'n'tell

- Basic Account Specification

It's all behavior

- Why do you write code?
- Why do you write a test?
- Every behavior of a system should be there because it adds concretely to the business value of the system as a whole.

Setting expectations

- Behavior is about expecting certain results
- balance should be \$0
- withdrawal should not throw an exception
- should, should not, should, should not

Show'n'tell

- Detailed Account Specification

cfSpec

- Written by Ron Hopper of Adelphus
- Modeled on RSpec - a Ruby framework
- Uses custom tags to create a Domain Specific Language (DSL)
 - For describing features of a system
 - For writing expectations of behavior

cfSpec's DSL

- `<cfimport prefix="" taglib="/cfspec"/>`
- `<describe hint="Some feature">`
 - `<it should="behave in this way">`
 - ... code the expectation ...
 - `</it>`
- `</describe>`

cfSpec's DSL

- `<beforeAll>` - execute once at start of run
- `<before>` - execute before every behavior
- `<after>` - execute after every behavior
- `<afterAll>` - execute once at end of run
- `<suite/>` - run every spec in directory tree
- `$(obj)` - turns obj into an expectation

Stubs

- When you write a spec, you have no code
- Sometimes you need an unwritten object
- A stub is a fake object
 - You can call methods you specify
 - Those calls return what you specify

Show'n'tell

- Mock Account Specification

Mocks

- Stubs are useful but they're not very smart
- They cannot tell you if calls were made in the correct sequence or the correct number of calls were made
- Mocks are stubs that **expect** method calls
 - cfSpec will have mocks shortly!

Show'n'tell

- cfSpec's own mockSpec
 - Of course it is developed using itself!

Summary

Kenite

Behavior-Driven

- Every system starts with requirements
- Requirements specify the (valuable) **behavior** that we should develop
- BDD builds on the principles of TDD
 - Takes it back to requirements and the language of the problem domain
 - Offers readable, executable specifications

More testing strategies

- Test scripts for components
 - e.g., Unit tests
 - Automate with MXUnit (and ant)
- Test scripts for completed system
 - e.g., Acceptance tests
 - Automate with Selenium (and ant)

Resources

- <http://behaviour-driven.org/> - Dan North
- <http://cfspec.riaforge.org/> - Project Home
- <http://github.com/adelphus/cfspec/> - BER
- <http://rspec.info/documentation/> - RSpec

Q&A

Getraile

Sean Corfield
sean@getraile.com
<http://getraile.com/>
<http://corfield.org/>