Acceptance Testing Java Applications with Cucumber, RSpec, and JRuby

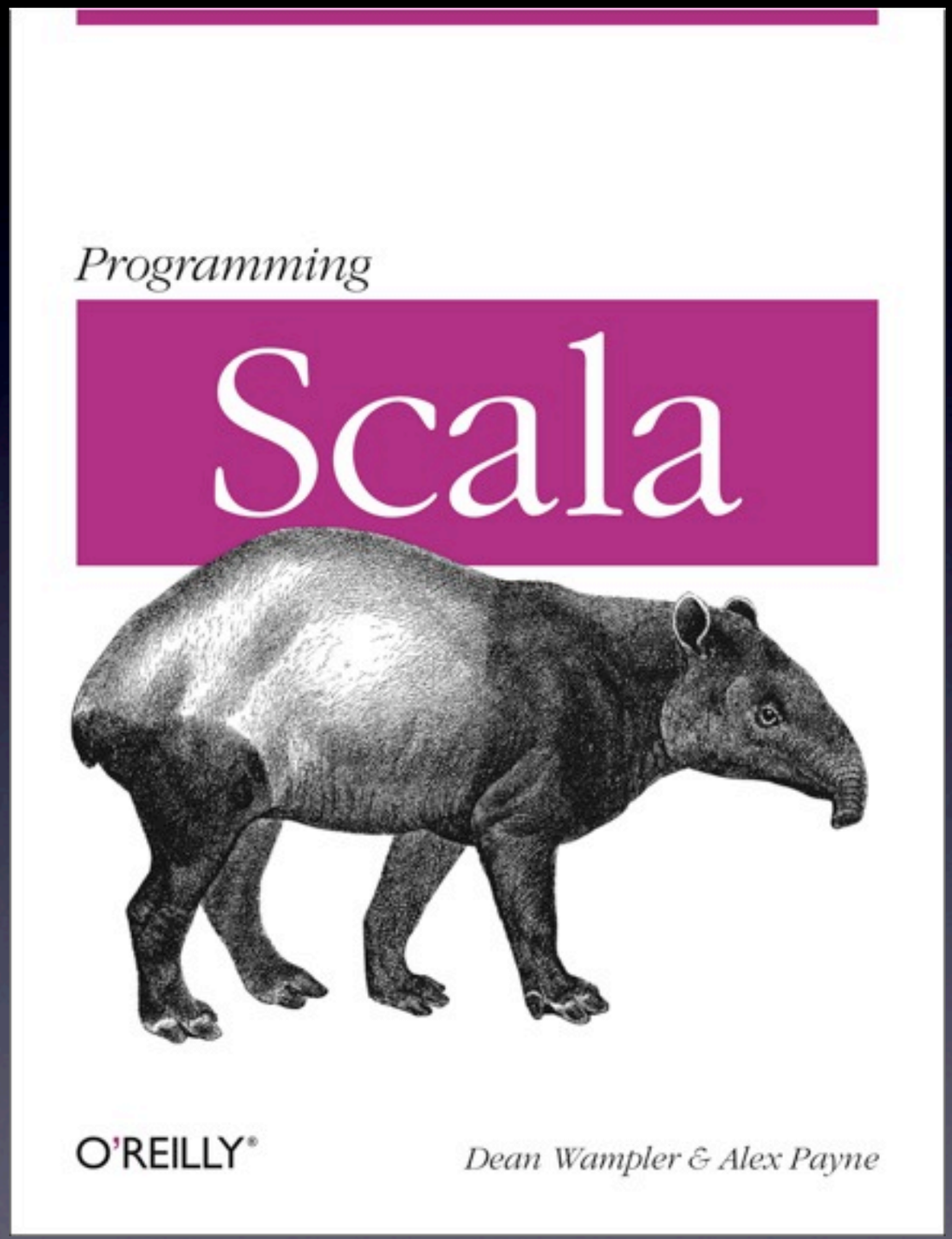
Dean Wampler
dean@objectmentor.com
@deanwampler

Aslak Hellesøy
aslak.hellesoy@gmail.com
@aslak_hellesoy

<*shameless-plug*/>

Co-author,
Programming Scala

programmingscala.com



http://cukes.info/

55000 gem downloads
1350 github followers
125 contributors
56 wiki pages
40 related tools
8 screencasts

http://www.flickr.com/photos/twose/887903401/

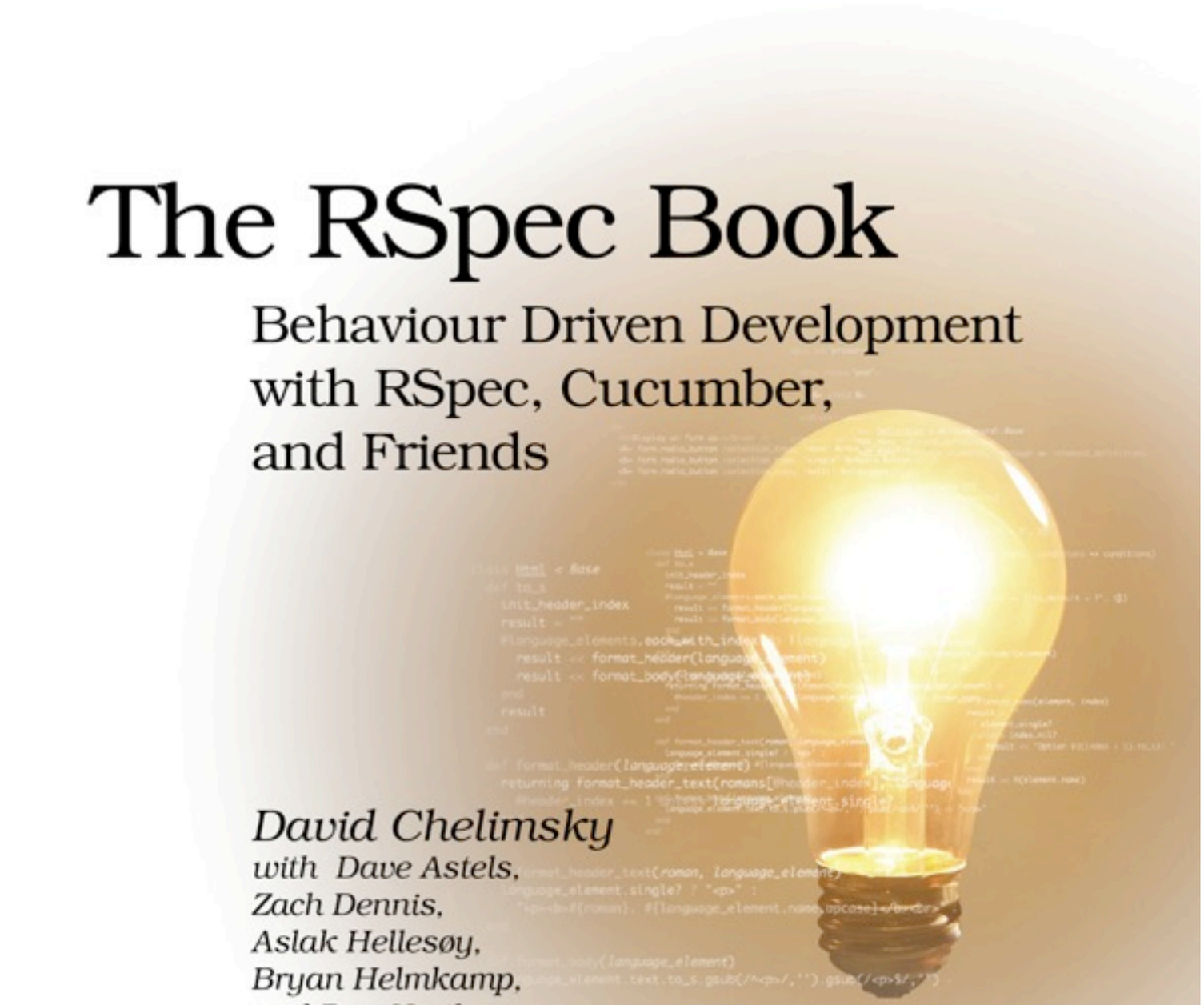
World Domination

SLIM

The Pragmatic Programmers

The RSpec Book

Behaviour Driven Development with RSpec, Cucumber, and Friends



David Chelimsky
with Dave Astels,
Zach Dennis,
Aslak Hellesøy,
Bryan Helmkamp,
and Dan North

Edited by Jacquelyn Carter

The Facets of Ruby Series

First, Using Cucumber

with Ruby


features

some.feature

step_definitions

your_steps.rb

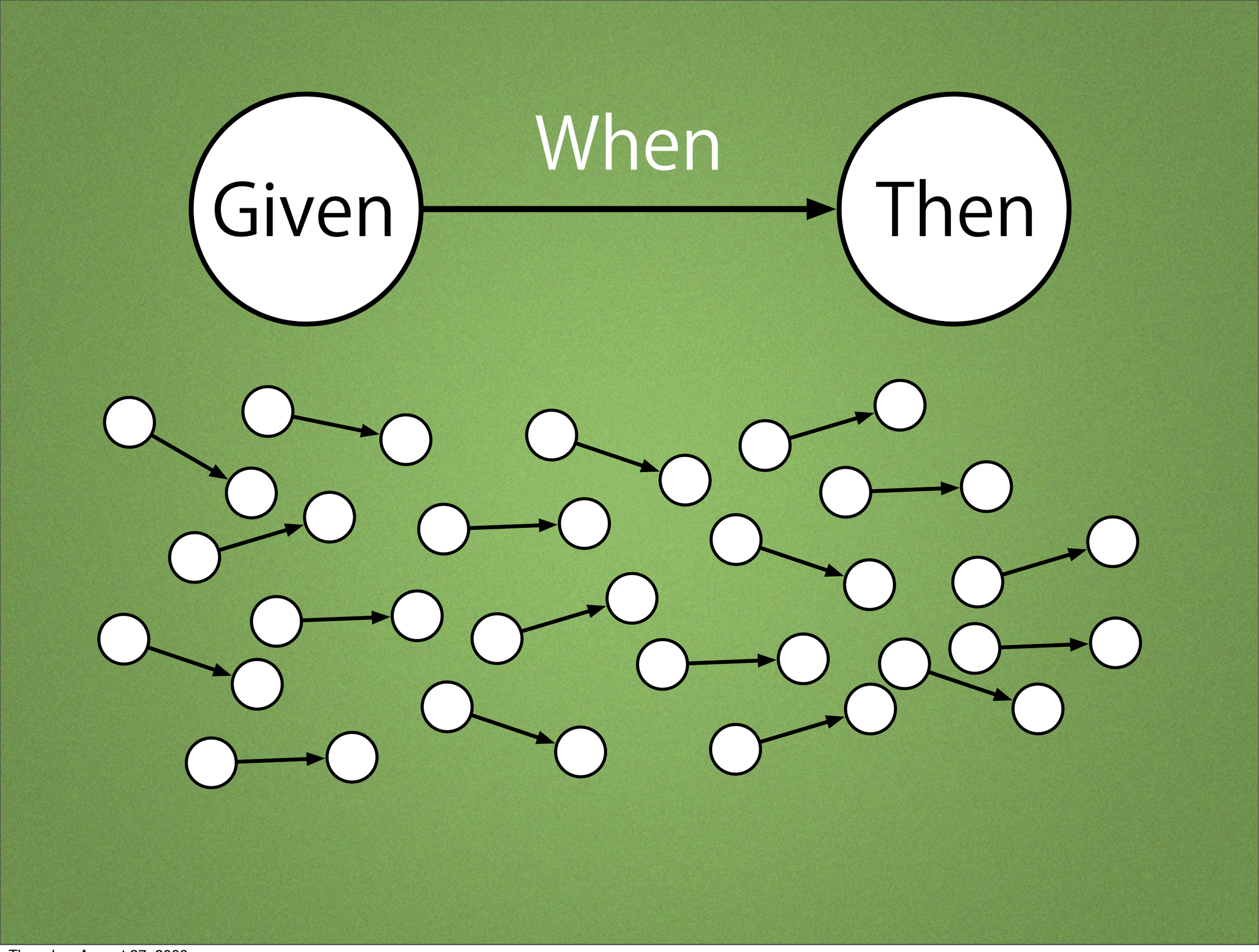
support

env.rb

Outside-In

Proposal notification

In order to reduce time spent on emailing Administrators should be able to mail proposal status to all owners


Given
When
Then

Our First Feature

```
Feature: Proposal notification
  In order to reduce time spent on emailing
  Administrators should be able to mail
  all proposals owners depending on status

  Scenario: Email accepted proposal
    Given aslak.hellesoy@gmail.com proposed Cucumber
    And the Cucumber proposal is approved
    When I send proposal emails
    Then aslak.hellesoy@gmail.com should get email
      """
      Hi aslak.hellesoy@gmail.com
      Congratulations, Cucumber was accepted.
      See you at RailsConf!
      """
```

features/proposal_notification.feature

Run the feature

```
$ cucumber features/proposal_notification.feature
```

```
Feature: Proposal notification
  In order to reduce time spent on emailing
  Administrators should be able to mail
  all proposals owners depending on status

  Scenario: Email accepted proposal
    Given aslak.hellesoy@gmail.com proposed Cucumber
    And the Cucumber proposal is approved
    When I send proposal emails
    Then aslak.hellesoy@gmail.com should get email
      """
      Hi aslak.hellesoy@gmail.com
      Congratulations, Cucumber was accepted.
      See you at RailsConf!
      """

1 scenario (1 undefined)
4 steps (4 undefined)
```

```
You can implement step definitions for undefined steps
with these snippets:

Given /^aslak\.hellesoy@gmail\.com proposed Cucumber$/ do
  pending
end

Given /^the Cucumber proposal is approved$/ do
  pending
end

When /^I send proposal emails$/ do
  pending
end

Then /^aslak\.hellesoy@gmail\.com should get email$/ do
  |string|
  pending
end
```

Step Definitions

```
Given /^aslak\.hellesoy@gmail\.com proposed Cucumber$/ do
  pending
end

Given /^the Cucumber proposal is approved$/ do
  pending
end

When /^I send proposal emails$/ do
  pending
end

Then /^aslak\.hellesoy@gmail\.com should get email$/ do
  |string|
  pending
end
```

features/step_definitions/proposal_steps.rb

```
Feature: Proposal notification
  In order to reduce time spent on emailing
  Administrators should be able to mail
  all proposals owners depending on status

  Scenario: Email accepted proposal
    Given aslak.hellesoy@gmail.com proposed Cucumber
      TODO (Cucumber::Pending)
      features/step_definitions/proposal_steps.rb:2
      features/proposal_notification.feature:7
    And the Cucumber proposal is approved
    When I send proposal emails
    Then aslak.hellesoy@gmail.com should get email
      """
      Hi aslak.hellesoy@gmail.com
      Congratulations, Cucumber was accepted.
      See you at RailsConf!
      """
```

Do what Regexp says

```
Given /^aslak\.hellesoy@gmail\.com proposed Cucumber$/ do
  Proposal.create!({
    :email => 'aslak.hellesoy@gmail.com',
    :title => 'Cucumber'
  })
end
```

features/step_definitions/proposal_steps.rb

```
Feature: Proposal notification
  In order to reduce time spent on emailing
  Administrators should be able to mail
  all proposals owners depending on status

  Scenario: Email accepted proposal
    Given aslak.hellesoy@gmail.com proposed Cucumber
      uninitialized constant Proposal (NameError)
      features/step_definitions/proposal_steps.rb:2
      features/proposal_notification.feature:7
    And the Cucumber proposal is approved
    When I send proposal emails
    Then aslak.hellesoy@gmail.com should get email
      """
      Hi aslak.hellesoy@gmail.com
      Congratulations, Cucumber was accepted.
      See you at RailsConf!
      """
```

```
Feature: Proposal notification
  In order to reduce time spent on emailing
  Administrators should be able to mail
  all proposals owners depending on status

  Scenario: Email accepted proposal
    Given aslak.hellesoy@gmail.com proposed Cucumber
    And the Cucumber proposal is approved
      TODO (Cucumber::Pending)
      features/step_definitions/proposal_steps.rb:9
      features/proposal_notification.feature:8
    When I send proposal emails
    Then aslak.hellesoy@gmail.com should get email
      """

      Hi aslak.hellesoy@gmail.com
      Congratulations, Cucumber was accepted.
      See you at RailsConf!
      """
```

```
Feature: Proposal notification
  In order to reduce time spent on emailing
  Administrators should be able to mail
  all proposals owners depending on status

  Scenario: Email accepted proposal
    Given aslak.hellesoy@gmail.com proposed Cucumber
    And the Cucumber proposal is approved
    When I send proposal emails
      TODO (Cucumber::Pending)
      features/step_definitions/proposal_steps.rb:15
      features/proposal_notification.feature:9
    Then aslak.hellesoy@gmail.com should get email
      """
      Hi aslak.hellesoy@gmail.com
      Congratulations, Cucumber was accepted.
      See you at RailsConf!
      """
```

```
Feature: Proposal notification
  In order to reduce time spent on emailing
  Administrators should be able to mail
  all proposals owners depending on status

  Scenario: Email accepted proposal
    Given aslak.hellesoy@gmail.com proposed Cucumber
    And the Cucumber proposal is approved
    When I send mass proposal email
      No route matches "/admin" with {:method=>:get}
      features/step_definitions/proposal_steps.rb:16
      features/proposal_notification.feature:9
    Then aslak.hellesoy@gmail.com should get email
      """
      Hi aslak.hellesoy@gmail.com
      Congratulations, Cucumber was accepted.
      See you at RailsConf!
      """
```

```
Feature: Proposal notification
  In order to reduce time spent on emailing
  Administrators should be able to mail
  all proposals owners depending on status

  Scenario: Email accepted proposal
    Given aslak.hellesoy@gmail.com proposed Cucumber
    And the Cucumber proposal is approved
    When I send mass proposal email
    Then aslak.hellesoy@gmail.com should get email
      """
      Hi aslak.hellesoy@gmail.com
      Congratulations, Cucumber was accepted.
      See you at RailsConf!
      """
```

```
Feature: Proposal notification
  In order to reduce time spent on emailing
  Administrators should be able to mail
  all proposals owners depending on status

  Scenario: Email accepted proposal
    Given aslak.hellesoy@gmail.com proposed Cucumber
    And the Cucumber proposal is approved
    When I send mass proposal email
    Then aslak.hellesoy@gmail.com should get email
    """
    Hi aslak.hellesoy@gmail.com
    Congratulations, Cucumber was accepted.
    See you at RailsConf!
    """
```

Scenarios? Steps?

Steps & Step Definitions

Step == Method invocation

```
Given aslak.hellesoy@gmail.com proposed Cucumber
```

Step Definition == Method defnition

```
Given /^aslak\.hellesoy@gmail\.com proposed Cucumber$/ do
end
```

Regexp group arguments

```
Given aslak.hellesoy@gmail.com proposed Cucumber
```

```
Given /^(.+) proposed (.+)$/ do |email, proposal_name|
end
```

```
Given aslak.hellesoy@gmail.com proposed Cucumber
```

$CUCUMBER_COLORS

Quoted arguments

```
Given I have "22" cukes in my belly
```

```
Given /^I have "([^\"]*)" cukes in my belly$/ do |n|
end
```

```
Given I have "2" cukes in my belly
```

Multiline args (String)

```
Then aslak.hellesoy@gmail.com should get email
  """
  Hi aslak.hellesoy@gmail.com
  Congratulations, Cucumber was accepted.
  See you at RailsConf!
  """
```

```
Then /^(.+) should get email$/ do |email, body|
end
```

Multiline args (Tables)

```
Given the following proposals
  | email                          | title     |
  | aslak.hellesoy@gmail.com | Cucumber |
  | bryan@brynary.com             | Webrat    |
```

```
Given /the following proposals$/ do |proposals|
  Proposal.create!(proposals.hashes)
end
```

Scenario Outline

```
Scenario Outline: Email accepted proposals
  Given the following proposals
    | email                       | title    |
    | aslak.hellesoy@gmail.com | Cucumber |
    | bryan@brynary.com           | Webrat   |
  And the <proposal> proposal is approved
  When I send proposal emails
  Then <email> should <what>

  Examples:
    | proposal | email                       | what           |
    | Cucumber | aslak.hellesoy@gmail.com | get email      |
    | Cucumber | bryan@brynary.com           | not get email |
    | Webrat   | bryan@brynary.com           | get email      |
```

Line numbers

```
Then bla bla # features/step_definitions/bla_steps.rb:16
```

Stack traces

```
When I send mass proposal email
  Could not find link "Send proposal emails"
  features/step_definitions/proposal_steps.rb:16
  features/notification.feature:9
```

```
$ cucumber features/notifications.rb:9
```

Other *Acceptance* Testing Tools

FitNesse

http://fitnesse.org

FitNesse. UserGuide.

TwoMinuteExample

TEST RESULTS [history]

Tests
Executed

Assertions: 6 right, 1 wrong, 0 ignored, 0 exceptions

A One-Minute Description

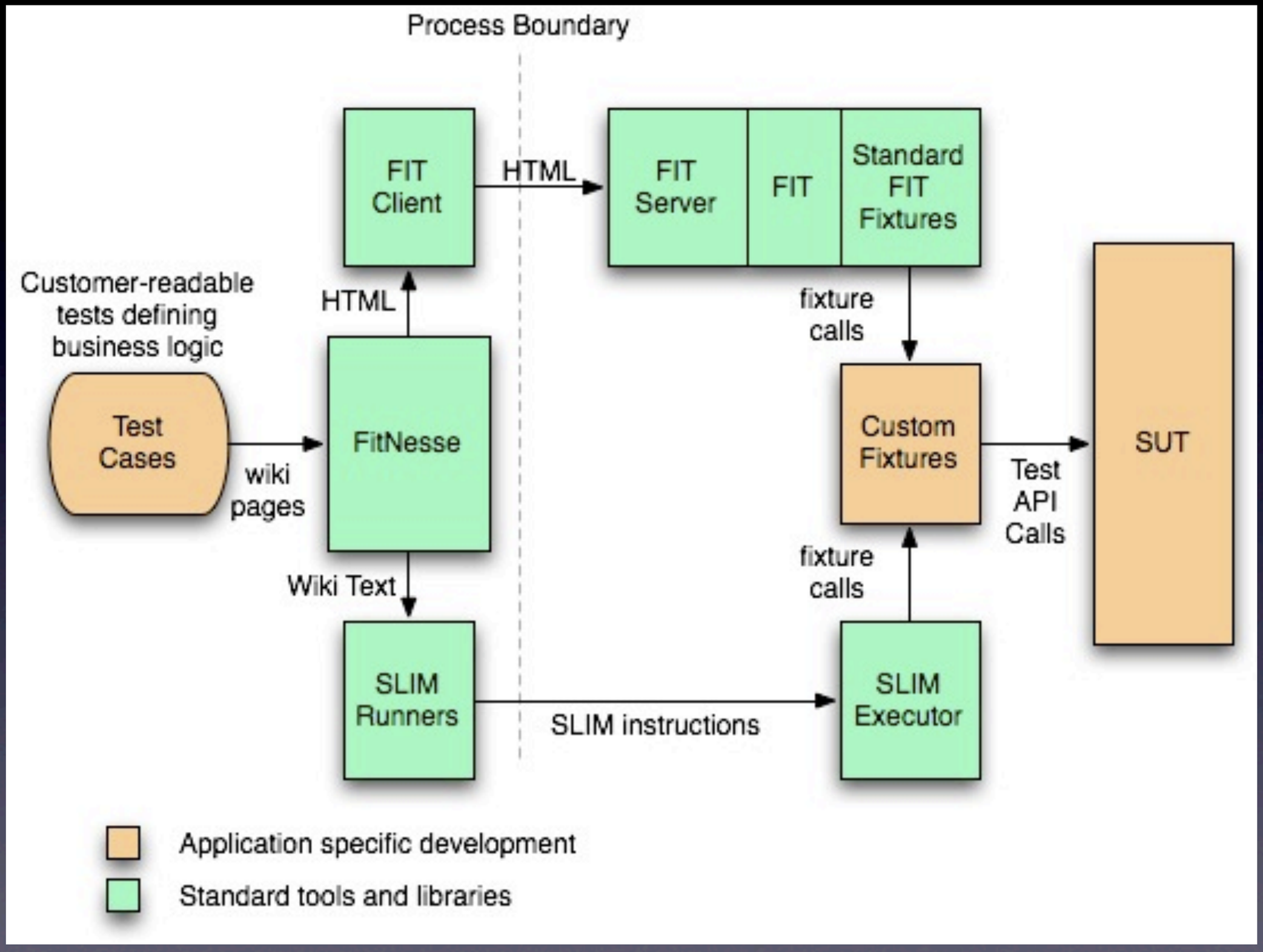
AN EXAMPLE FITNESSE TEST

If you were testing the division function of a calculator application, you might like t see some examples working. You might want to see what you get back if you ask it to divide 10 by 2. (You might be hoping for a 5!)

In FitNesse, tests are expressed as tables of **input** data and **expected output** data. Here is one way to specify a few division tests in FitNesse:

eg.Division		
numerator	denominator	quotient?
10	2	5.0
12.6	3	4.2
22	7	3.142857142857143~=3.14
9	3	3.0<5
11	2	4<5.5<6
100	4	[25.0] expected [33]

This style of FitNesse test table is called a Decision Table, each row represents a complete scenario of example inputs and outputs. Here, the "numerator" and "denominator" columns are for inputs, and the question mark in the "quotient?" column tells FitNesse that this is our column of expected outputs. Notice our "10/2


Process Boundary
FIT Client
HTML
FIT Server
FIT
Standard FIT Fixtures
Customer-readable tests defining business logic
HTML
fixture calls
Test Cases
wiki pages
FitNesse
Custom Fixtures
Test API Calls
SUT
Wiki Text
fixture calls
SLIM Runners
SLIM instructions
SLIM Executor
Application specific development
Standard tools and libraries

Advantages

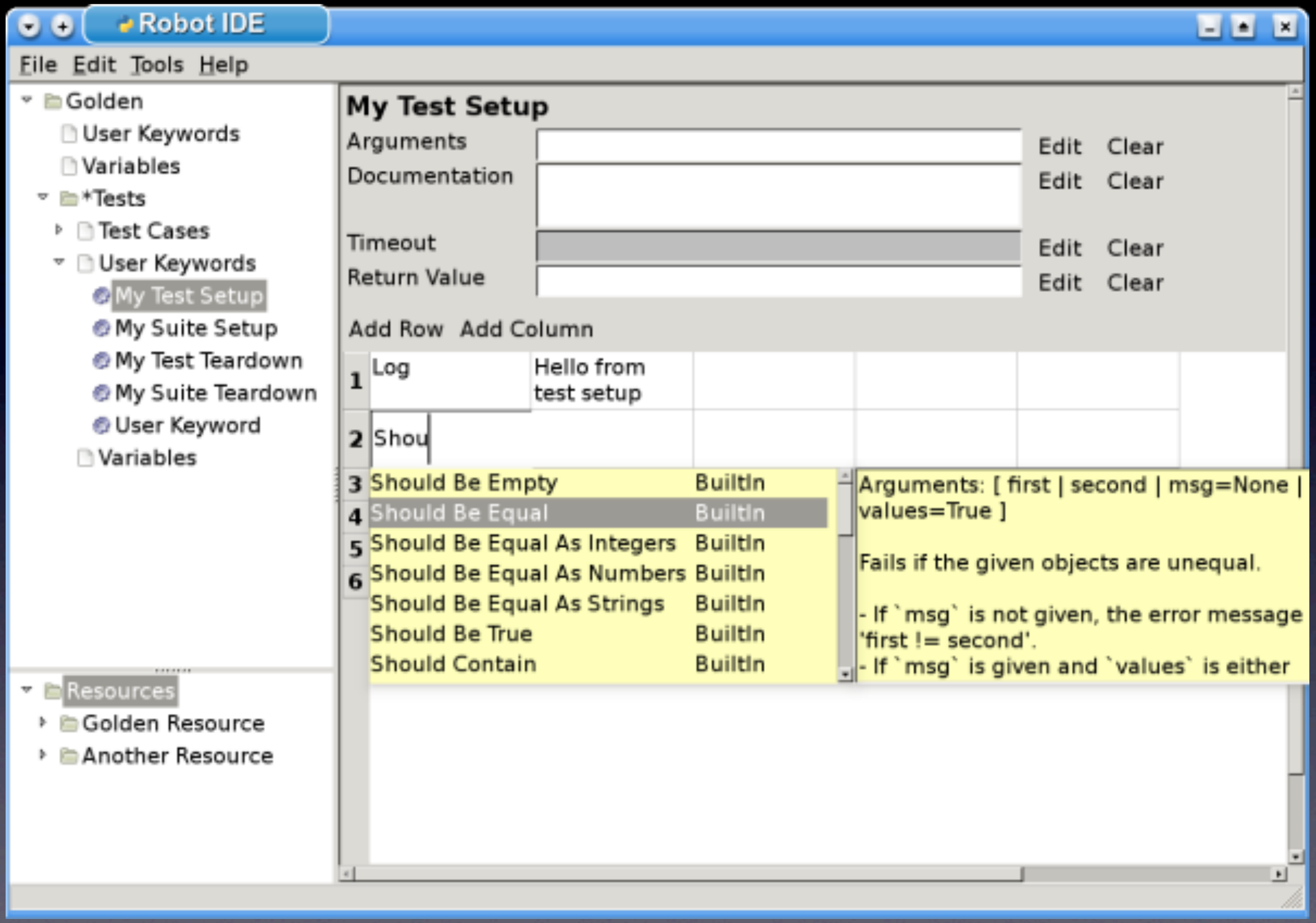
- Wiki UI
- Language neutral
- Table-based data format

Disadvantages

- Nonstandard wiki markup
- Developers must learn API
- No IDE, CM integration

Robot Framework

http://code.google.com/p/robotframework/

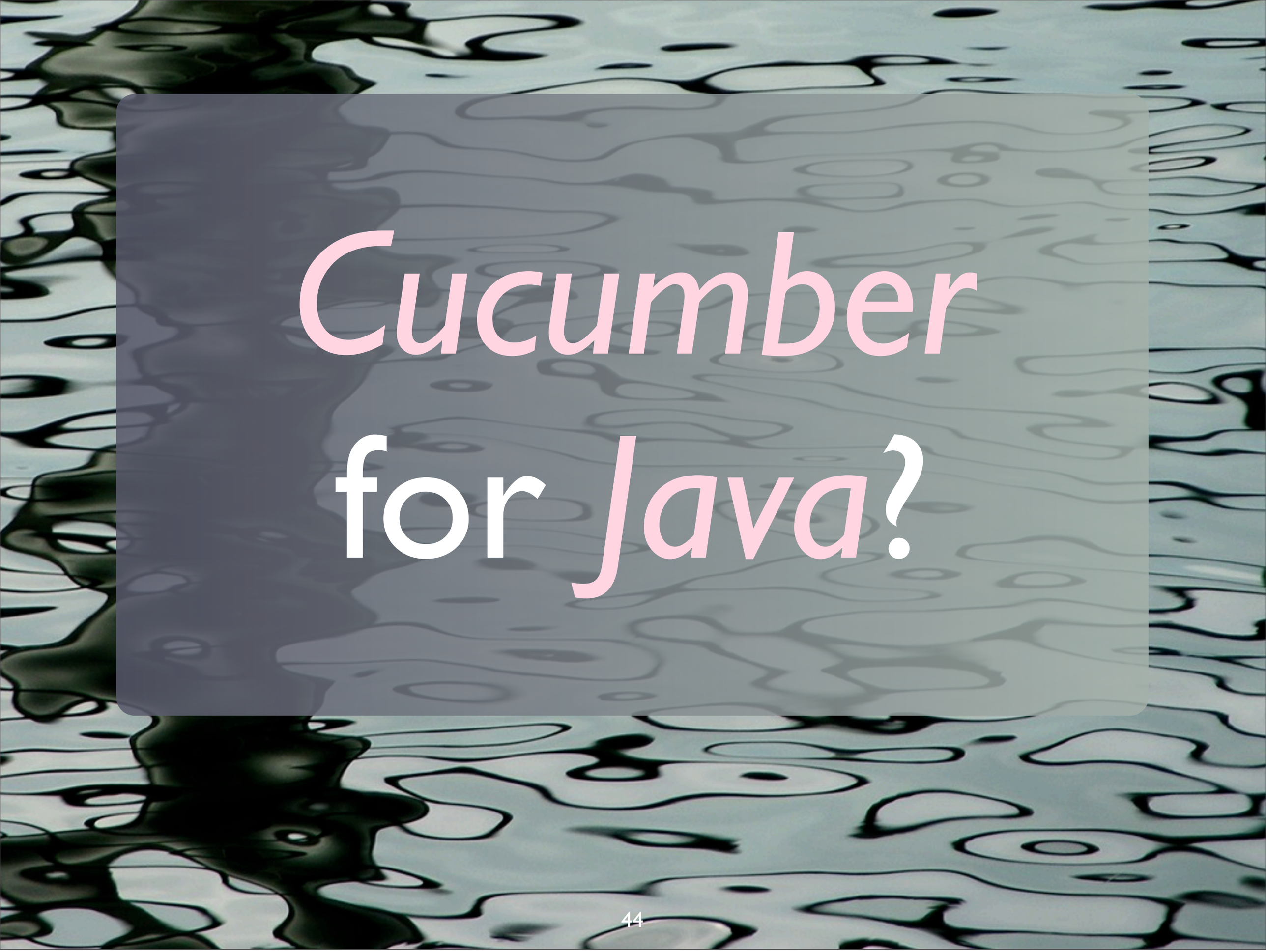

Robot IDE
File Edit Tools Help
Golden
User Keywords
Variables
*Tests
Test Cases
User Keywords
My Test Setup
My Suite Setup
My Test Teardown
My Suite Teardown
User Keyword
Variables
Resources
Golden Resource
Another Resource
My Test Setup
Arguments
Documentation
Timeout
Return Value
Edit Clear
Add Row Add Column
1 Log Hello from test setup
2 Shou
3 Should Be Empty BuiltIn
4 Should Be Equal BuiltIn
5 Should Be Equal As Integers BuiltIn
6 Should Be Equal As Numbers BuiltIn
Should Be Equal As Strings BuiltIn
Should Be True BuiltIn
Should Contain BuiltIn
Arguments: [first | second | msg=None | values=True]
Fails if the given objects are unequal.
- If `msg` is not given, the error message 'first != second'.
- If `msg` is given and `values` is either

Advantages

- Tabular, keyword-driven syntax
- *RIDE* with code completion
- Selenium integration

Disadvantages

- Developers must learn API
- RIDE somewhat bare bones
- RIDE not as convenient as a wiki
- No IDE, CM integration


Cucumber
for Java?

Advantages

- Simple, seductive DSL
- Easy developer adoption
- Full power of Ruby

Disadvantages

- No stakeholder UI

Cuke4Duke

http://wiki.github.com/aslakhellesoy/cuke4duke

Use *Java* for *Step Definitions*

```java
package cukes;
import cuke4duke.Given;
import java.util.List;
import java.util.ArrayList;

public class BellySteps {
 private List<String> belly = …;

 @Given("I have (\\d+) cukes in my belly")
 public void bellyCukes(int cukes) {
  for(int i = 0; i < cukes; i++) {
   belly.add("cukes");
  }
 }
}
```

For comparison,

here's the *Ruby*:

```ruby
Given /I have (\d+) cukes in my belly/ do |n|
  @belly ||= []
  n.to_i.times do |i|
    @belly << "cuke"
  end
end
```

Which would you prefer to write??

Or,

Use *Groovy*

```
this.metaClass.mixin(
  cuke4duke.GroovyDsl)

Given(~/I have entered (\d+) into the
calculator/) { int number ->
    calc = new calc.Calculator()
    calc.push number
}
```

JUnit vs. *RSpec* for *Unit* Testing Java

Advantages

- Seductive DSL
- Full power of Ruby

Disadvantages

- No IDE refactoring support
- Poor TDD cycle compared to using JUnit

We prefer *JUnit* for *Unit* testing Java

Exercises

Downloads

- Git
  - Windows: msysgit
  - Mac OSX:
    - sudo port install git-core
    - *or use* git-osx-installer
- Ant
- Ivy

Getting Started

```
$ git clone git://github.com/aslakhellesoy/
Cuke4DukeCodebreaker.git
$ cd Cuke4DukeCodebreaker/
$ ant install-deps  # lots of downloads!!
$ ant cucumber
```

You're at "Master"

- In CodeBreakerSteps.java, add

```
@Given("the secret code is (.*)")
public void theSecretCodeIs(
    String secret) {
  // create Game class...
}
```

- Solution is "ex01" branch...

Exercise #2

```
$ git checkout origin/ex01

$ ant cucumber
```

Exercise #2

- Add the rest of the steps
- @When and @Then
- Hard-code result string in game “fixme”.

Exercise #3

```
$ git checkout origin/ex02

$ ant cucumber
```

Exercise #3

- Get the scenarios to pass:
  - Remove spaces in strings!
  - Match secret[i] and guess[i] => ‘b’
    - secret[i] = ‘_’ (so it won’t get reused)
    - Match secret[i] and guess[j] => ‘w’
    - Sort all b’s before w’s
- Solution in ex03

Exercise #4

```
$ git checkout origin/ex03
$ ant cucumber
```

- Everything should be green!

Exercise #4

- Now get the scenarios to pass using Groovy
  - Create CodeBreakerSteps.groovy in same directory as the .java file.
    - Change the *.java regexs to not match
    - Port *.java steps to groovy.
      - Use JUnit assertions
- Solution in ex04

Final

```
$ git checkout origin/ex04
$ ant cucumber
```

- Everything should be green!

Extra Credit

- Port Groovy code to JRuby.

Thanks!

dean@objectmentor.com
@deanwampler

aslak.hellesoy@gmail.com
@aslak_hellesoy

programmingscala.com