



Cucumber

Behavior Driven Development
with elegance and joy

Getting Up and Running with BDD on Rails

Nicholas Cancelliere

email: ncancelliere@gmail.com

twitter: ozmox

Overview

- * Behavior Driven Development Basic Concepts
- * Cucumber from 10,000 ft
- * Cucumber in the Rails Environment
- * Writing your first Feature / Scenario
- * Additional Resources

Behavior Driven

- * “the next step” from Test-Driven Development
- * tests what an object *does* rather than what it *is*
- * intention more important than implementation
- * outside-in approach
- * offers all the benefits of TDD
- * it is not Test-After Development

Benefits of TDD/BDD

- * more productive (less debugging, fewer prod defects)
- * helps drive programming design (KISS/YAGNI)
- * delays implementation decisions
- * greater level of code trust
- * code more modularized, flexible and extensible

Outside-In

User

Client

Views

Controllers

Model

Value is in the **views!**

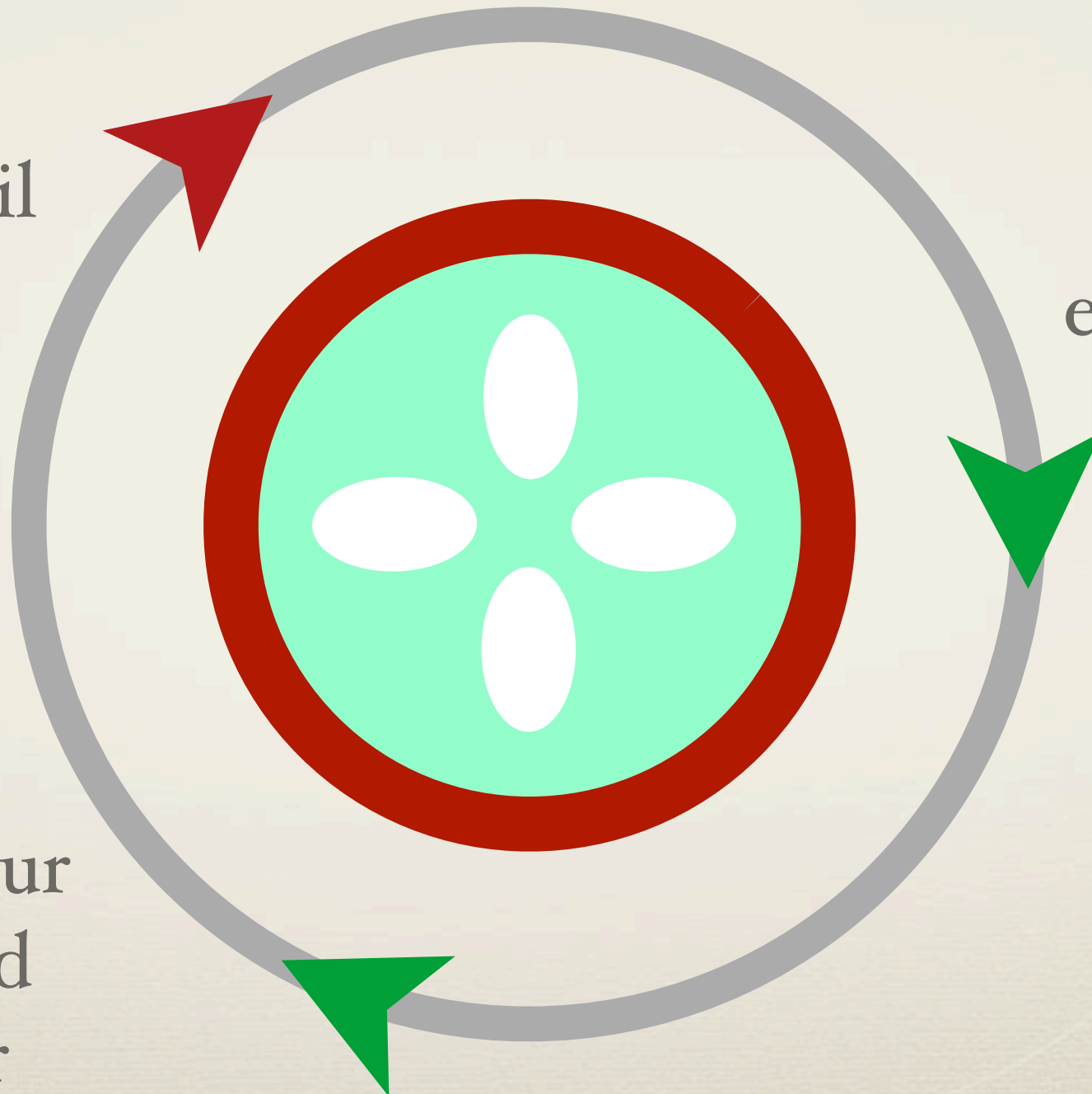
Often html but can also be programmatic interfaces (such as XML or JSON),
remember valuable to the ***user***.

Red Green Refactor

write enough
test code to fail

write just
enough code to
pass

review your
work and
refactor

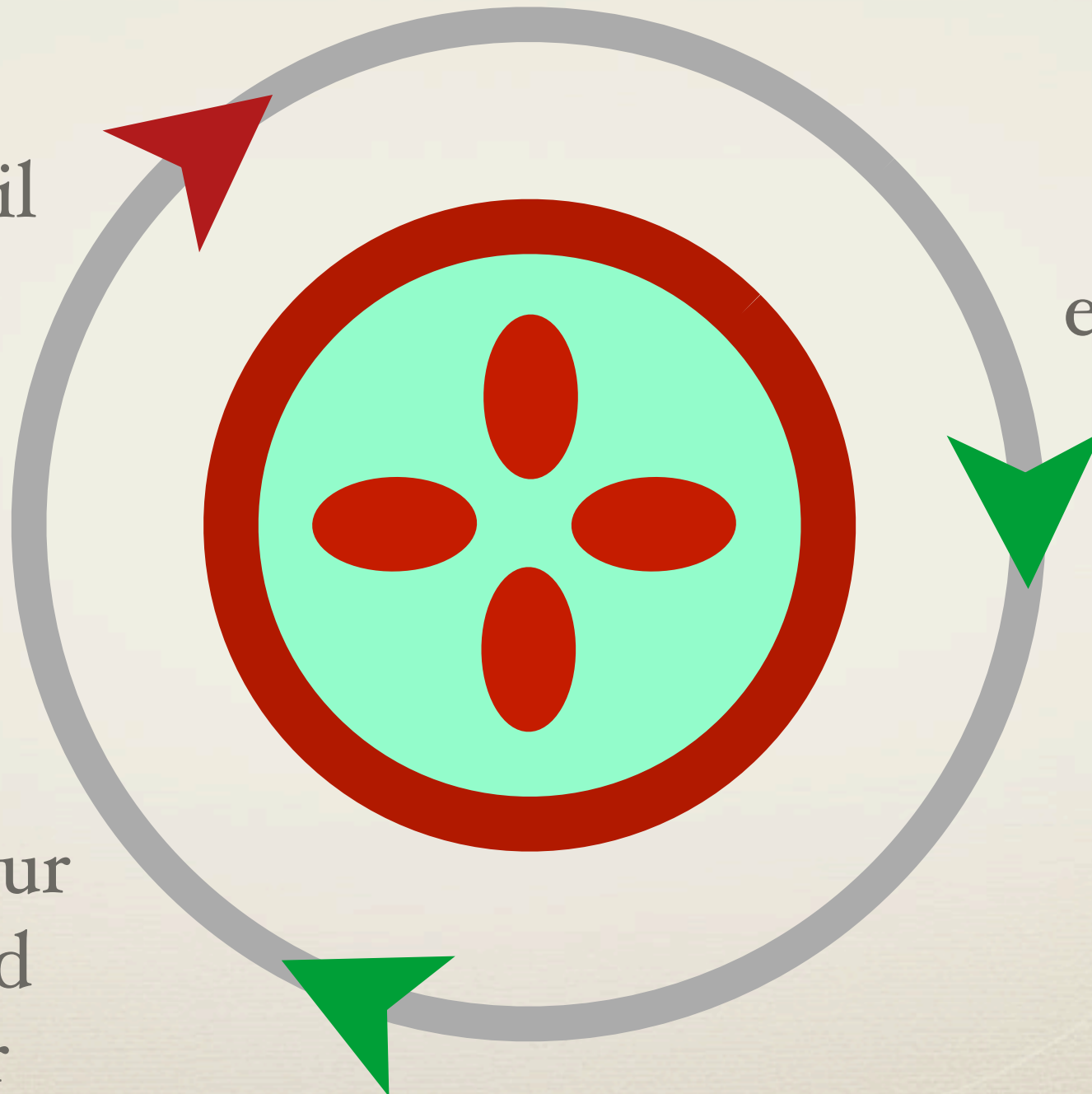


Red Green Refactor

write enough
test code to fail

write just
enough code to
pass

review your
work and
refactor

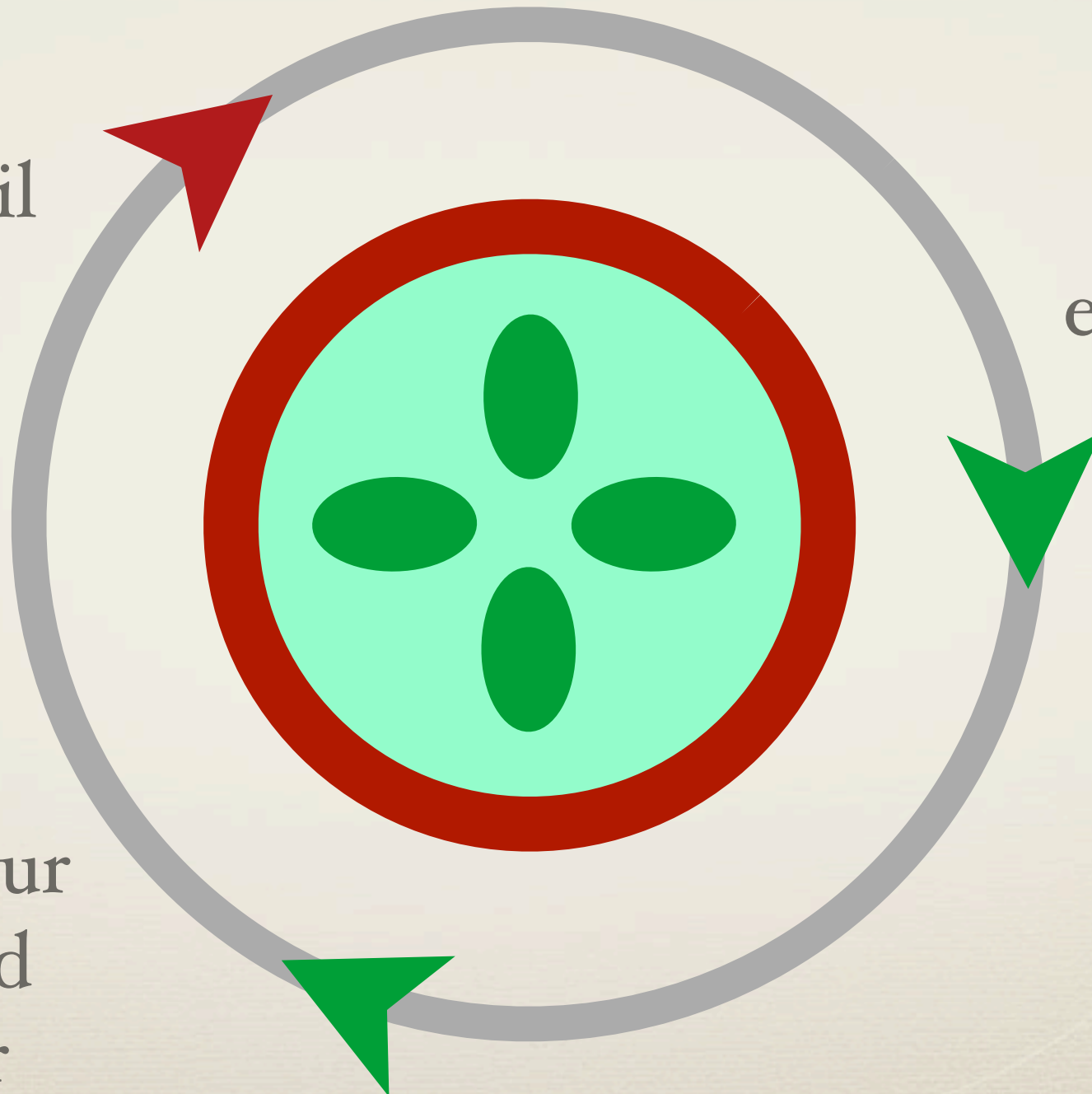


Red Green Refactor

write enough
test code to fail

write just
enough code to
pass

review your
work and
refactor

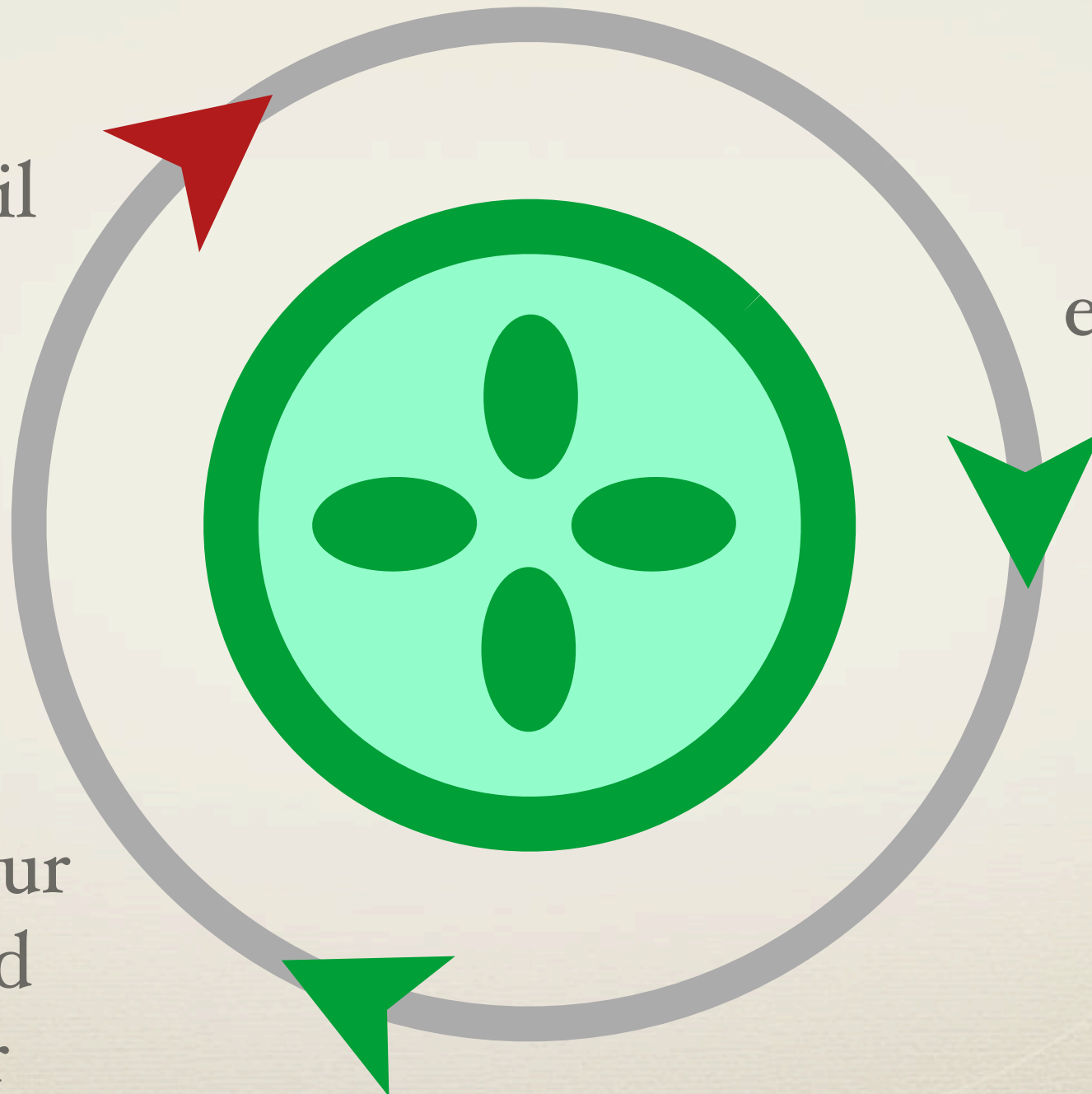


Red Green Refactor

write enough
test code to fail

write just
enough code to
pass

review your
work and
refactor



User Stories

- * concept popular in Agile practices like Scrum and XP
- * define the user's role, what it is they need/want to do, and why they need/want (the value)

As a user with an address book, I need to add contacts so I can easily retrieve them later.

As a user with an address book, I need to edit/delete contacts so I can keep it up-to-date and manageable.

As a user with an address book, I need to send my contacts to other users to be shared.

Pop the Why Stack

- * Ask “Why” up to 5 times
- * Stop when you find the money
 - * Protect revenue
 - * Increase revenue
 - * Manage cost

Example Pop'in

c: 'People need to log in.'

d: 'Why?'

c: 'Um, identify users?'

d: 'Why do you need to identify users?'

c: 'So we know who's publishing what.'

d: 'Why would you need to know who publishes what?'

c: 'If content belongs to someone it seems trustworthy.'

d: 'Why does content need to be trustworthy?'

c: 'People will be more interested in the content if so.'

d: 'Why do you need people interested in the content?'

c: 'So people come back and visit the site?'

d: 'Why do you want people to come back and revisit?'

c: 'More visits will increase our ad revenue.'

The Story

As an author, I need to log into the site so that articles I create are associated to me and seem trustworthy, driving more traffic to the site.

The Story

So that articles I create are associated to me, seem trustworthy and drive more traffic to the site, as an author, I need to log in.

Enter Cucumber

- * Dan North ported JBehave to Ruby as RBehave
- * RSpec merged RBehave in as the Story Runner
 - * First only supported Ruby for scenarios
 - * Later supported plain text, but still limited
- * Aslak Hellesøy in 2008 rewrote Story Runner as Cucumber (named so by his fiancée)
- * Cucumber has since taken to a life of it's own

Cucumber

- * written in Ruby itself and best matched with Ruby projects, but can be used for Java, .NET or Flex
- * supports web apps in any language:
 - * integrates with Webrat, Watir, Selenium, et. al.
 - * Gherkin customization edit `/cucumber/languages.yml`
- * minimum knowledge of Ruby required, you can pick it up and get going in a few days

Gherkin: the 'unpickle' variety

- * a business readable, domain specific language that Cucumber understands
- * two-stage parser for input file (plain text):
 1. divides the file into sections (eg. feature, scenarios)
 2. sections are divided into steps
- * step-definitions are always matched by Ruby methods
- * line-oriented (like YAML); line endings terminate statements (steps) and spaces or tabs to indent

Given When Then

- * Used extensively in scenario definitions (Gherkin)
- * Use “And” to chain together

```
Given an invalid user
When I go to the sign-in page
And I fill in "userlogin" with "baduser"
And I fill in "password" with "badpassword"
And I submit "new_user_session"
Then I should see "Sorry, we could not sign you in."
And I should see "Login"
```


Installing in Rails

```
gem install cucumber rspec-rails webrat
```

```
/config/environments/test.rb
```

```
config.gem 'rspec-rails', :lib => false
```

```
config.gem 'rspec', :lib => false
```

```
config.gem 'cucumber'
```

```
config.gem 'webrat'
```

```
rake gems:install RAILS_ENV=test
```

```
script/generate cucumber
```

```
script/generate rspec
```


Test::Unit Shoulda

<http://giantrobots.thoughtbot.com/2009/2/20/mixing-cucumber-with-test-unit>

/features/support/env.rb

```
# require 'cucumber/rails/rspec'  
# require 'webrat/rspec-rails'
```

/features/step_definitions/webrat_steps.rb

```
# replace any matchers that use RSpec syntax
```

```
Then /^I should see "(.*)"/ do |text|  
  # response.body.should =~ /#{text}/m  
  assert_match /#{text}/m, @response.body  
end
```


Out of the Box

- * Webrat comes with a bunch of handy helpers out of the box to make life easy for web application scenarios
- * rake features - task to run all Cucumber tests

```
/features
  review_past_events.feature
/step_definitions
  event_steps.rb
  webrat_steps.rb
/support
  env.rb
  paths.rb
```


Features

Feature:

So that I can manage my own personal information securely
As a registered user
I want to log into the system

Scenario: With a valid user name and password

Given a valid user
When I go to the sign-in page
And I fill in "user_session_login" with "user007"
And I fill in "user_session_password" with "testme123"
And I submit "new_user_session"
Then I should see "Sign-in successful"
And I should see "user007"

Scenario: With an invalid user name and password

Given an invalid user
When I go to the sign-in page
And I fill in "user_session_login" with "baduser"
And I fill in "user_session_password" with "badpassword"
And I submit "new_user_session"
Then I should see "Sorry, we could not sign you in."
And I should see "Login"


```
Glawar:speakeasy nicholas$ cucumber features -s
```

```
Feature:
```

```
  As a registered user
```

```
  I want to securely log in
```

```
  So that I can manage my own personal information
```

```
Scenario: With a valid user name and password
```

```
  Given a valid user
```

```
  When I go to the sign-in page
```

```
  And I fill in "user_session_login" with "user007"
```

```
  And I fill in "user_session_password" with "testme123"
```

```
  And I submit "new_user_session"
```

```
  Then I should see "Sign-in successful"
```

```
  And I should see "user007"
```

```
Scenario: With an invalid user name and password
```

```
  Given an invalid user
```

```
  When I go to the sign-in page
```

```
  And I fill in "user_session_login" with "baduser"
```

```
  And I fill in "user_session_password" with "badpassword"
```

```
  And I submit "new_user_session"
```

```
  Then I should see "Sorry, we could not sign you in."
```

```
  And I should see "Login"
```

```
2 scenarios (2 undefined)
```

```
14 steps (12 skipped, 2 undefined)
```

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

```
Given /^a valid user$/ do
```

```
  pending
```

```
end
```

```
Given /^an invalid user$/ do
```

```
  pending
```

```
end
```

```
Glawar:speakeasy nicholas$
```


Step Definition

```
# /features/support/user_steps.rb
|
Given /^a valid user$/ do
  @user = Factory.create(:valid_user)
end

Given /^an invalid user$/ do
  @user = nil
end
```



```
Glamar:speakeasy nicholas$ cucumber features -s
```

```
Feature:
```

```
  As a registered user  
  I want to securely log in  
  So that I can manage my own personal information
```

```
Scenario: With a valid user name and password
```

```
  Given a valid user
```

```
    When I go to the sign-in page
```

```
      uninitialized constant UserSessionsController (NameError)
```

```
      (eval):2:in `^I go to (.+)$/'
```

```
      features/step_definitions/sign-in.feature:8:in `When I go to the sign-in page'
```

```
    And I fill in "user_session_login" with "user007"
```

```
    And I fill in "user_session_password" with "testme123"
```

```
    And I submit "new_user_session"
```

```
    Then I should see "Sign-in successful"
```

```
    And I should see "user007"
```

```
Scenario: With an invalid user name and password
```

```
  Given an invalid user
```

```
    When I go to the sign-in page
```

```
      uninitialized constant UserSessionsController (NameError)
```

```
      (eval):2:in `^I go to (.+)$/'
```

```
      features/step_definitions/sign-in.feature:17:in `When I go to the sign-in page'
```

```
    And I fill in "user_session_login" with "baduser"
```

```
    And I fill in "user_session_password" with "badpassword"
```

```
    And I submit "new_user_session"
```

```
    Then I should see "Sorry, we could not sign you in."
```

```
    And I should see "Login"
```

```
2 scenarios (2 failed)
```

```
14 steps (2 failed, 10 skipped, 2 passed)
```

```
Glamar:speakeasy nicholas$
```


Cukeing

```
class UserSession < Authlogic::Session::Base  
end|
```


Cukeing

Feature:

As a registered user
I want to securely log in
So that I can manage my own personal information

Scenario: With a valid user name and password

Given a valid user

When I go to the sign-in page

uninitialized constant UserSessionsController (NameError)

(eval):2:in `/^I go to (.+)\$/'

features/step_definitions/sign-in.feature:8:in `When I go to the sign-in page'

And I fill in "user_session_login" with "user007"

And I fill in "user_session_password" with "testme123"

And I submit "new_user_session"

Then I should see "Sign-in successful"

And I should see "user007"

Scenario: With an invalid user name and password

Given an invalid user

When I go to the sign-in page

uninitialized constant UserSessionsController (NameError)

(eval):2:in `/^I go to (.+)\$/'

features/step_definitions/sign-in.feature:17:in `When I go to the sign-in page'

And I fill in "user_session_login" with "baduser"

And I fill in "user_session_password" with "badpassword"

And I submit "new_user_session"

Then I should see "Sorry, we could not sign you in."

And I should see "Login"

Cukeing

Feature:

```
class UserSessionsController < ApplicationController
  before_filter :require_no_user, :only => [:new, :create]
  before_filter :require_user, :only => :destroy

  def new
    @user_session = UserSession.new
  end

  def create
    @user_session = UserSession.new(params[:user_session])
    if @user_session.save
      flash[:notice] = "Sign-in successful!"
      redirect_to account_path
    else
      flash[:error] = "Sorry, we could not sign you in. Double check your login and password"
      render :action => :new
    end
  end

  def destroy
    current_user_session.destroy
    flash[:notice] = "Sign-out successful!"
    redirect_to root_path
  end
end
```

And I should see "Login"

Cukeing

Feature:

As a registered user
I want to securely log in
So that I can manage my own personal information

Scenario: With a valid user name and password

Given a valid user

When I go to the sign-in page

Missing template user_sessions/new.erb in view path app/views (ActionView::MissingTemplate)

/usr/local/lib/ruby/1.8/benchmark.rb:308:in `realtime'

/usr/local/lib/ruby/1.8/benchmark.rb:308:in `realtime'

(eval):2:in `/^I go to (.+)\$/'

features/step_definitions/sign-in.feature:8:in `When I go to the sign-in page'

And I fill in "user_session_login" with "user007"

And I fill in "user_session_password" with "testme123"

And I submit "new_user_session"

Then I should see "Sign-in successful"

And I should see "user007"

Scenario: With an invalid user name and password

Given an invalid user

When I go to the sign-in page

Missing template user_sessions/new.erb in view path app/views (ActionView::MissingTemplate)

/usr/local/lib/ruby/1.8/benchmark.rb:308:in `realtime'

/usr/local/lib/ruby/1.8/benchmark.rb:308:in `realtime'

(eval):2:in `/^I go to (.+)\$/'

features/step_definitions/sign-in.feature:17:in `When I go to the sign-in page'

Cukeing

Feature:

As a registered user
I want to securely log in
So that I can manage my own personal information

Scenario: With a valid user name and password

Given a valid user

#signin_form

- form_for @user_session do |f|

 #login_field

 = f.label :login

 = f.text_field :login

 #password_field

 = f.label :password

 = f.password_field :password

 = link_to 'Recover a forgotten password...', new_password_reset_path, :style =>

Scenario: With an invalid user name and password

Given an invalid user

When I go to the sign-in page

Missing template user_sessions/new.erb in view path app/views (ActionView::MissingTemplate)

/usr/local/lib/ruby/1.8/benchmark.rb:308:in `realtime'

/usr/local/lib/ruby/1.8/benchmark.rb:308:in `realtime'

(eval):2:in `/^I go to (.+)\$/'

features/step_definitions/sign-in.feature:17:in `When I go to the sign-in page'

Cukeing

Feature:

Feature:

As a registered user
I want to securely log in
So that I can manage my own personal information

Scenario: With a valid user name and password

Given a valid user

When I go to the sign-in page

And I fill in "user_session_login" with "user007"

And I fill in "user_session_password" with "testme123"

And I submit "new_user_session"

Then I should see "Sign-in successful"

And I should see "user007"

Scenario: With an invalid user name and password

Given an invalid user

When I go to the sign-in page

And I fill in "user_session_login" with "baduser"

And I fill in "user_session_password" with "badpassword"

And I submit "new_user_session"

Then I should see "Sorry, we could not sign you in."

And I should see "Login"

2 scenarios (2 passed)

14 steps (14 passed)

Glawar:speakeasy nicholas\$

(eval):2:in `^I go to (.+)\$/'

features/step_definitions/sign-in.feature:17:in `When I go to the sign-in page'

Backgrounds

Feature:

So that I can understand what I'm weakest in
As a swimmer
I want to review all of my times from past events

Scenario: reviewing all events

Given a swimmer logged in
When something
Then something else

Scenario: reviewing others' events

Given a swimmer logged in
When something
Then something else

Scenario: reviewing future events

Given a swimmer logged in
When something
Then something else

Backgrounds

Feature:

So that I can understand what I'm weakest in
As a swimmer
I want to review all of my times from past events

Background:

Given a swimmer logged in

Scenario: reviewing all events

When something
Then something else

Scenario: reviewing others' events

When something
Then something else

Scenario: reviewing future events|

When something
Then something else

Backgrounds

Feature:

So that I can understand what I'm weakest in

Fe As a swimmer

I want to review all of my times from past events

Background:

Given a swimmer logged in

Scenario: reviewing all events

When something

Then something else

Scenario: reviewing others' events

When something

Then something else

Scenario: reviewing future events

When something

Then something else

5 scenarios (3 undefined, 2 passed)

23 steps (9 undefined, 14 passed)

Scenario Outlines

Feature:

So that I can understand what I'm weakest in
As a swimmer
I want to review all of my times from past events

Background:

Given a swimmer logged in

Scenario Outline: stats

When something <status>
Then shows something <thing>

Examples:

status	thing	
good	'Great job!'	
average	'Average'	
poor	'Needs Improvement'	

Scenario Outlines

Feature:

So that I can understand what I'm weakest in
As a swimmer
I want to review all of my times from past events

Background:

Given a swimmer logged in

Scenario Outline: stats

When something <status>
Then shows something <thing>

Examples:

status	thing	
good	'Great job!'	
average	'Average'	
poor	'Needs Improvement'	

5 scenarios (3 undefined, 2 passed)

23 steps (9 undefined, 14 passed)

Remember...

- * **Cucumber** defines the *features* you wish you had.
- * **RSpec** (or **Test::Unit**) defines the *interactions* and *objects* you wish you had.
- * You're building a business domain language!

A Good Investment

- * Conversation
- * Acceptance Criteria
- * Design
- * Documentation
- * Automated Functional and Integration Tests

What do I use?

Selenium

Webrat

Unit Testing

Speed



Integration

Cucumber Smells

- * Relying too much on state in your step-definitions
- * Tests with no user value
- * Too much concrete, less abstract

Cucumber Smells

- * Relying too much on state in your step-definitions
- * Tests with no user value
- * Too much concrete, less abstract

```
Given /^state$/ do
  @article = Article.create!
end
Given /^coupled by state/ do
  @article.title = 'Bad'
end
```


Cucumber Smells

- * Relying too much on state in your step-definitions
- * Tests with no user value
- * Too much concrete, less abstract

```
Given /^check the db/ do
  Article.find(1).should_not == nil
end
```


Cucumber Smells

- * Relying too much on state in your step-definitions
- * Tests with no user value
- * Too much concrete, less abstract

```
Given I go to the login page  
And I fill in "username" with "john"  
And I fill in "password" with "testpass"  
And I click "login"
```


Cucumber Smells

- * Relying too much on state in your step-definitions
- * Tests with no user value
- * Too much concrete, less abstract

Given I'm logged in

Cucumber Smells

- * Relying too much on state in your step-definitions
- * Tests with no user value
- * Too much concrete, less abstract

```
Given /I am logged in/ do
  @user = User.create!(:login => 'john',
                        :password => 'testpass')
  And 'I fill in "username" with "john"'
  And 'I fill in "password" with "testpass"'
  And 'I click "login"'
end
```


Additional Resources / Topics

- * <http://cukes.info>
- * The RSpec Book (beta @ PragProg.com)
- * Textmate Bundles are available!
- * Extending Cucumber with World
- * Using Hooks (careful they're global)