A bit of RSpec history

Josua Schmid
@schmijos

Railshöck Spring 2019

What is RSpec

A library for Behaviour Driven Development

The three amigos discuss specifications
and gather a common understanding.

Not verification testing **but** behaviour specification

```ruby
describe 'hash generation' do
  context 'of the first position' do
    subject { instance[0].to_h }

    let(:bbs_attributes) do
      {
        subpositions: [
          a_hash_including(product_code: bse.product_code),
          a_hash_including(product_code: bso.product_code),
          a_hash_including(product_code: bss.product_code)
        ]
      }
    end

    it { is_expected.to include(bbs_attributes) }
  end
end
```

RSpec Timeline:

* 2004 "Why your code sucks" of Dave Astel
* 2005 Reference implementation of Steven Baker
* 2006 v0.7 uses now RSpec instead of TestUnit
* 2007 v0.8 switches to expectation matchers
* 2007 v1.0 (Ruby 1.8)
* 2010 v2.0 (Ruby 1.9) takes in micronaut runner
* 2014 v3.0 (Ruby 2.1)

Cucumber Timeline:

* 2004 JBehave
* 2007 RBehave
* 2007 RBehave becomes RSpec story runner
* 2009 Cucumber gets extracted

2005 – RSpec v0.1

```
require 'spec'

class RenuoKnowledgeSpec < Spec::Context
  def setup
    @stack = ['Rails', 'Angular', 'ReactNative']
  end

  def can_do_rails
    @stack.should_include 'Rails'
  end

  def cannot_do_rust_yet
    @stack.should_not_include 'Rust'
  end
end

runner = Spec::TextRunner.new($stdout)
runner.run(RenuoKnowledgeSpec)
```

```
def should_include(sub, message=nil)
  message ||= "<#{self.inspect}> should include <#{sub.inspect}>"
  should(message) { self.include? sub }
end
```

```
def should(message=nil)
  message ||= "Expectation not met."
  if (! yield)
    raise Spec::Exceptions::ExpectationNotMetError.new(message)
  end
end
```

2006 – RSpec v0.3

```
require 'spec'

knowledge_stack = ['Rails', 'Angular', 'ReactNative']

specification 'Renuo can do Rails' do
  knowledge_stack.should_include 'Rails'
end

example 'Renuo cannot do Rust (yet)' do
  knowledge_stack.should_not_include 'Rust'
end

Spec::TextRunner.new.run
```

```
"specification" is aliased if an environment variable is defined
USER=marick bundle exec spec ./test_spec.rb
```

2007 – RSpec v0.8

```ruby
require 'spec'

renuo_knowledge = ['Rails', 'Angular', 'React']

context 'Renuo' do
  specify 'can do rails' do
    renuo_knowledge.should include 'Rails'

  end
end
```

```ruby
require 'spec'

renuo_knowledge = ['Rails', 'Angular', 'React']

context 'Renuo' do
  specify 'can do rails' do
    renuo_knowledge.should(Spec::Matchers::Include.new('Rails'))

  end
end
```

```ruby
require 'spec'

renuo_knowledge = ['Rails', 'Angular', 'React']

context 'Renuo' do
  specify 'can do rails' do
    Spec::Expectations::ExpectationMatcherHandler.handle_matcher(
      renuo_knowledge,
      Spec::Matchers::Include.new('Rails')
    )

  end
end
```

```ruby
require 'spec'

renuo_knowledge = ['Rails', 'Angular', 'React']

context 'Renuo' do
  specify 'can do rails' do
    Spec::Matchers::Include.new('Rails').matches?(renuo_knowledge).should be(true)

  end
end
```

2009 – RSpec v1.2

```ruby
require 'spec'

describe 'Renuo knowledge' do
  context 'when not a hash' do
    let(:knowledge) { ['Rails', 'Angular', 'React'] }

    it 'can do rails' do
      knowledge.should include 'Rails', 'React'
    end

    example 'can not do Rust (yet)' do
      knowledge.should_not include 'Rust'
    end
  end
end
```

```ruby
require 'spec'

describe 'Renuo knowledge' do
  context 'when hash' do
    let(:knowledge) do
      {
        rails: ['Alessandro', 'Simon'],
        angular: ['Simon', 'Martin']
      }
    end

    it 'can do rails' do
      knowledge.should include :rails
      knowledge.should include(rails: ['Alessandro', 'Simon'])
    end
  end
end
```

```ruby
# lib/spec/matchers/include.rb

if actual.is_a?(Hash)
  if expected.is_a?(Hash)
    expected.each_pair do |k,v|
      return false unless actual[k] == v
    end
  else
    return false unless actual.has_key?(expected)
  end
else
  return false unless actual.include?(expected)
end

return true
```

2013 – RSpec v2.13

```ruby
describe 'Renuo knowledge' do
  let(:employees) do
    [
      { name: 'Alessandro', knows: %w(Rails Cancancan)},
      { name: 'Josua', knows: %w(9gag) },
    ]
  end

  RSpec::Matchers.define :an_employee_knowing do |expected|
    match { |actual| actual[:knows].include?(expected) }
  end

  it 'can do rails' do
    employees.should include(an_employee_knowing('Rails'))
    employees.should include(include(name: 'Alessandro'))
  end
end
```

```ruby
# lib/rspec/matchers/built_in/include.rb

if comparing_hash_values?(actuals, expected)
  expected.__send__(hash_predicate) { |k,v|
    actuals.has_key?(k) && actuals[k] == v
  }
elsif comparing_hash_keys?(actuals, expected)
  actuals.has_key?(expected)
elsif comparing_with_matcher?(actual, expected)
  actual.any? { |value| expected.matches?(value) }
else
  actuals.include?(expected)
end
```

2014 – RSpec v3.0

```ruby
describe 'Renuo knows' do
  let(:employees) do
    [
      { name: 'Alessandro', knows: %w(Rails Cancancan)},
      { name: 'Josua', knows: %w(9gag) }
    ]
  end

  it 'can do rails' do
    employees.should
      include(a_hash_including(knows: a_collection_including('Rails')))
  end
end
```

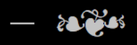
```ruby
# lib/rspec/matchers.rb

def include(*expected)
  BuiltIn::Include.new(*expected)
end
alias_matcher :a_collection_including, :include
alias_matcher :a_string_including,     :include
alias_matcher :a_hash_including,       :include
alias_matcher :including,              :include
```

```ruby
class Include < BaseMatcher; … end

class BaseMatcher
  include RSpec::Matchers::Composable
end

module Composable
  def and(matcher) …
  def or(matcher) …
  def values_match?(expected, actual) …
end
```

```ruby
# lib/rspec/matchers/composable.rb

def values_match?(expected, actual)
  expected = with_matchers_cloned(expected)
  Support::FuzzyMatcher.values_match?(expected, actual)
end
```

```ruby
# lib/rspec/support/fuzzy_matcher.rb

def self.hashes_match?(expected_hash, actual_hash)
  return false if expected_hash.size != actual_hash.size

  expected_hash.all? do |expected_key, expected_value|
    actual_value = actual_hash.fetch(expected_key) { return false }
    values_match?(expected_value, actual_value)
  end
end
```

Future



```ruby
describe 'Renuo knows and teaches' do
  let(:employees) do
    [
      { name: 'Alessandro', knows: %w(Rails Cancancan), teaches: %w('Rails')},
      { name: 'Josua', knows: %w(Ruby) }
    ]
  end

  it 'can do rails' do
    employees.should include(a_hash_including(
      a_string_matching(/knows|teaches/): a_collection_including('Rails')
    ))
  end
end
```

```ruby
# spec/rspec/support/fuzzy_matcher_spec.rb:118

it 'does not fuzzy match on keys' do
  expect(/foo/ => 1).not_to match_against("foo" => 1)
end
```

Thank you!