Ruby

Rspec

Overview

- **TestUnit/Minitest** realiza el trabajo duro en las pruebas unitarias, pero es lindo que las pruebas sean más **descriptivas** y más expresadas en **lenguaje natural**..
- Con RSpec la **escritura** de los tests es **más intuitiva** y también lo son las **salidas** de los tests.

Instalar RSpec

- Instalación
  - gem install rspec
- Cuando se desea empezar a escribir las pruebas:
  - rspec --init
  - Esto crea un directorio spec donde deberían ir todas las pruebas
  - Crea .rspec spec_helper.rb donde se podría incluir código que aplica a todas las especificaciones que se van a escribir.

describe()

- El método de más alto nivel en RSpec se llama describe()
  - Funciona como un **agrupador de métodos**
- describe() recibe un String o una Clase como argumento.
- Todos los specs(especificaciones) **deben estar dentro de un bloque describe**()

before() and after() methods

- `before()` y `after()` son similares a `setup()` y `teardown()` de MiniTest
- Puede recibir indistintamente `:each` o `:all`(raramente usado) para **especificar** si el bloque se va a ejecutar antes/después de cada test o una sola vez para todos los tests.
- `before :all` puede ser útil, si por ejemplo quisiéramos conectarnos a la base de datos una sola vez.

it() method

- Utilizado para definir la especificación RSpec actual
- Recibe un String opcional que describe el comportamiento que se está probando.

calculator.rb

```ruby
class Calculator

  attr_reader :name

  def initialize(name)
    @name = name
  end

  def add(one, two)
    one - two
  end

  def substract(one, two)
    one + two
  end

  def divide(one, two)
    one / two
  end
end
```

```ruby
require 'rspec'
require_relative '../calculator'

describe Calculator do
  before { @calculator = Calculator.new('RSpec calculator')}

  it "should add 2 numbers correctly" do
    expect(@calculator.add(2, 2)).to eq 4
  end

  it "should subtract 2 numbers correctly" do
   expect(@calculator.subtract(4, 2)).to eq 2
  end

  it "should sum two odd numbers and become even" do
   expect(@calculator.add(3, 3)).to be_even
   expect(@calculator.add(3, 3)).not_to be_odd
  end

end
```

Ejecución del RSpec

Es más intuitivo y más english-like

```
FF.

Failures:

  1) Calculator should add 2 numbers correctly
     Failure/Error: expect(@calculator.add(2, 2)).to eq 4

       expected: 4
            got: 0

       (compared using ==)
     # ./spec/calculator_spec.rb:8:in `block (2 levels) in <top (required)>'

  2) Calculator should subtract 2 numbers correctly
     Failure/Error: expect(@calculator.subtract(4, 2)).to eq 2

       expected: 2
            got: 6

       (compared using ==)
     # ./spec/calculator_spec.rb:12:in `block (2 levels) in <top (required)>'

Finished in 0.01603 seconds (files took 0.06788 seconds to load)
3 examples, 2 failures

Failed examples:

rspec ./spec/calculator_spec.rb:7 # Calculator should add 2 numbers correctly
rspec ./spec/calculator_spec.rb:11 # Calculator should subtract 2 numbers correctly
```

Entonces

- RSpec es más expresivo y más intuitivo
- Pero.. si te gusta MiniTest, sos libre!
- Cuando escribimos los tests no es importante mientras los escribamos!