Ruby

Rspec Matchers

RSpec Matchers

- RSpec utiliza los métodos **to** y **not_to** en todas las especificaciones externas.
- **to/not_to** reciben un parámetro: Un Matcher
- Ejemplos de matchers:
  - be_true/be_false
  - eq 3
  - raise_error

be_predicate

- Si el objeto sobre el cual estamos operando el test tiene un método que retorne un valor boolean, automáticamente se puede utilizar el matcher **be_predicate(boolean).**
- Entonces, por ejemplo, be_nil es un matcher válido, ya que todo objeto en ruby tiene el método **:nil?**.

```
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
```

format documentation

```
→  leccion-16 git:(master) ✗ rspec --format documentation

Calculator
  should add 2 numbers correctly (FAILED - 1)
  should subtract 2 numbers correctly (FAILED - 2)
  should sum two odd numbers and become even

Failures:

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

Ejemplo de matcher custom

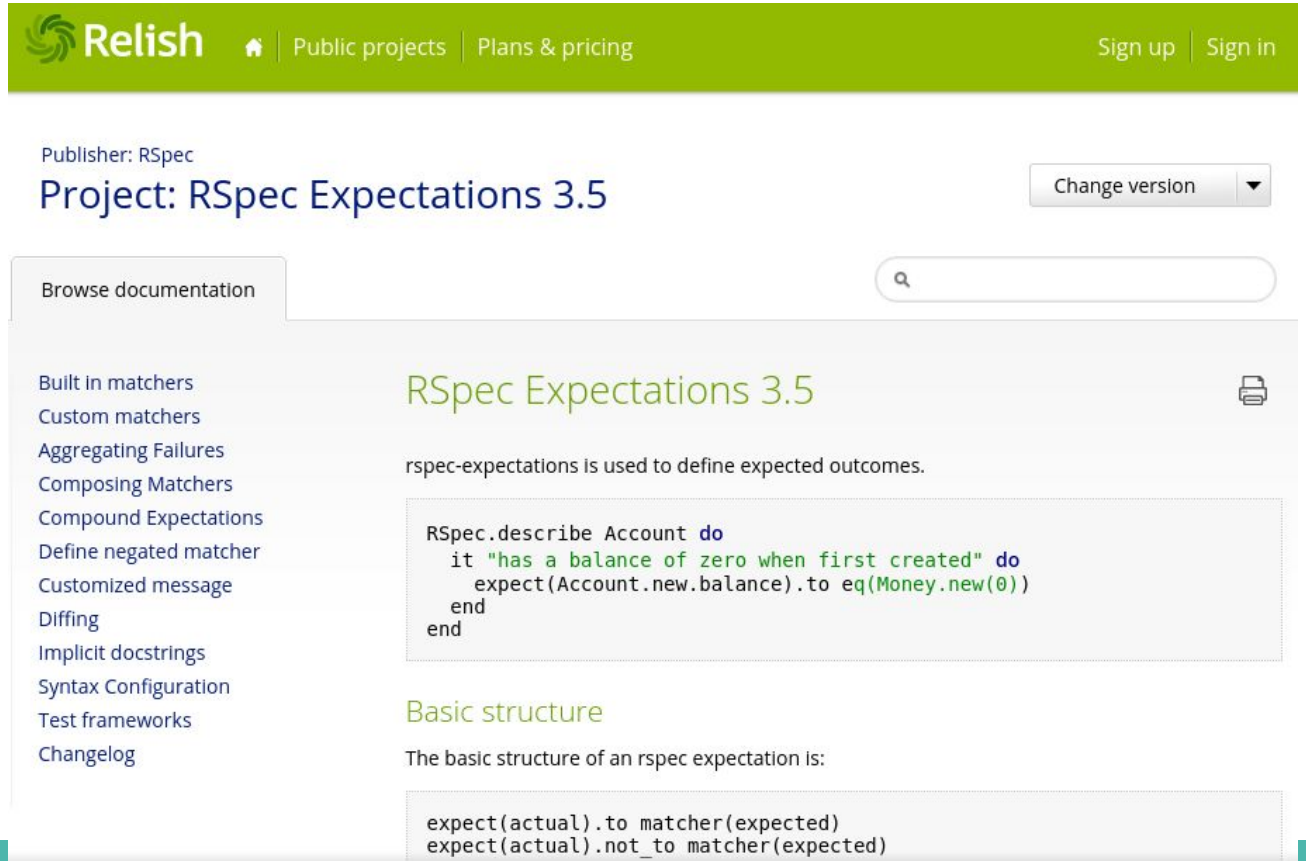
```
def algo?
  return true
end
def has_algo?
  return false
end
```

```
it "debería ser algo" do
  expect(@calculator).to be_algo
end
it "debería tener algo" do
  expect(@calculator).not_to have_algo
end
```

Ejecución de ejemplo

```
→  leccion-16 git:(master) ✗ rspec --format documentation

Calculator
  should add 2 numbers correctly (FAILED - 1)
  should subtract 2 numbers correctly (FAILED - 2)
  should sum two odd numbers and become even
  debería ser algo
  debería tener algo

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)>'
```

Mas Matchers (https://relishapp.com/rspec/rspec-expectations/docs)



Visualizar ejemplo practica_clases_spec.rb

```ruby
require 'rspec'
require 'rspec/its'
require_relative '../practica_clases.rb'

describe "Clases" do

  context "check results" do
    p1 = Persona.new("Juan", "Perez")
    p2 = Persona.new("Juan", "Benitez")
    p3 = Persona.new("Juana", "Perez")
    p4 = Persona.new("Romina", "Benitez")

    it "resultado de busqueda inesperado" do
      expect(Persona.search("Perez").size).to be == 2
    end
  end

  context "verificando propiedades de instancia" do
    subject(:john) { Persona.new("Cristian", "Rodriguez") }

    it "missing nombre" do
      is_expected.to respond_to(:nombre)
    end

    it "missing apellido" do
      is_expected.to respond_to(:apellido)
    end

  end

  context "verificando propiedades de clase" do
    subject(:class) { Persona }

    it "missing search" do
      is_expected.to respond_to(:search)
    end
  end
end
```

Ejercicio: team_players.rb

- Crear un método de clase teams_quantity que indique la cantidad de teams que existen creados.

Ejercicio

- Utilizar RSpec para:
  - Verificar que Player tenga los atributos: name, skill_level, y age
  - Verificar que Team responda al método de clase teams_quantity
  - Verificar que Team permita agregar uno o mas jugadores con el método add_players
  - Verificar que el resultado del select sea de tamaño 2.

Entonces...

RSpec tiene un montón de matchers pre-construidos listos para simplificar la escritura de tests!