EBook Gratis

APRENDIZAJE

rspec

Free unaffiliated eBook created from
Stack Overflow contributors.

#rspec

Tabla de contenido

Acerca de 1

Capítulo 1: Empezando con rspec 2

Observaciones 2

Examples 2

Instalando RSpec 2

Un ejemplo simple de RSpec 3

Capítulo 2: Expectativas de RSpec 5

Introducción 5

Observaciones 5

Examples 5

Uso básico 5

Capítulo 3: Mock RSpec 6

Observaciones 6

Examples 6

Stubbing con permitir 6

Burlándose de establecer una expectativa de mensaje con esperar 6

Capítulo 4: RSpec Core 8

Examples 8

Ejecución de ejemplos con una etiqueta dada 8

Capítulo 5: RSpec Matcher 9

Introducción 9

Examples 9

Igualadores de igualdad 9

Creditos 12

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: rspec

It is an unofficial and free rspec ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official rspec.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con rspec

Observaciones

RSpec es una herramienta BDD que se utiliza para especificar y probar programas Ruby. Se utiliza principalmente para especificar y probar clases y métodos, es decir, para pruebas unitarias.

La gema rspec es solo una meta-gema que incorpora las tres partes de RSpec. Esas tres partes son también una forma de estructurar esta documentación.

- rspec-core proporciona la forma de RSpec de estructurar y ejecutar pruebas: el ejecutable de la línea de comandos de `rspec` , la `describe` , el `context` y `it` métodos, ejemplos compartidos, etc. Está documentado en el tema del núcleo de RSpec .
- rspec-expectations proporciona el soporte de RSpec para esperar los resultados de las pruebas: la sintaxis `expect` / `to` expectation y los emparejadores integrados de RSpec. (También proporciona la obsoleta `should` sintaxis expectativa.) Está documentado en el tema Expectativas RSpec .
- rspec-mocks proporciona el soporte de RSpec para pruebas de dobles: `double` , `allow` , `expect` , `receive` , haber `have_received` , etc. Está documentado en el tema de RSpec Mocks .

También está la gema rspec-rails, que amplía RSpec con soporte para probar los tipos de clases utilizadas en las aplicaciones de Rails, y con soporte para escribir especificaciones de características (pruebas de aceptación) que prueban la aplicación desde el punto de vista del usuario.

La documentación oficial de RSpec y rspec-rails está aquí: https://www.relishapp.com/rspec

Examples

Instalando RSpec

La forma más común de instalar la gema RSpec es usar Bundler . Agregue esta línea al `Gemfile` su aplicación:

```
gem 'rspec'
```

Y luego ejecuta `bundle` para instalar las dependencias:

```
$ bundle
```

Alternativamente, puede instalar la gema manualmente:

```
$ gem install rspec
```

Después de instalar la gema, ejecute el siguiente comando:

```
rspec --init
```

Esto creará una carpeta de `spec` para sus pruebas, junto con los siguientes archivos de configuración:

- Un directorio de `spec` en el cual colocar archivos de especificaciones.
- un archivo `spec/spec_helper.rb` con opciones de configuración predeterminadas
- un archivo `.rspec` con banderas de línea de comando predeterminadas

Un ejemplo simple de RSpec

En greeter.rb (donde sea que vaya en tu proyecto):

```
class Greeter
  def greet
    "Hello, world!"
  end
end
```

En spec / greeter_spec.rb:

```
require_relative '../greeter.rb'

RSpec.describe Greeter do
  describe '#greet' do
    it "says hello" do
      expect(Greeter.new.greet).to eq("Hello, world!")
    end
  end
end
```

Así que nuestra estructura de archivos se ve como:

```
$ tree .
.
├── greeter.rb
└── spec
    └── greeter_spec.rb

1 directory, 2 files
```

Salida

```
$rspec greeter_spec.rb
Finished in 0.00063 seconds (files took 0.06514 seconds to load)
1 example, 0 failures
```

En terminología RSpec, el archivo es una "especificación" de `Greeter` y el bloque `it` es un "ejemplo". La línea con `expect` es una expectativa. Si se cumple la expectativa, no pasa nada y la prueba pasa. Si no, la prueba falla.

Este ejemplo también muestra que los bloques de `describe` se pueden anidar, en este caso para

transmitir que el método de `greet` es parte de la clase `Greet` . El `#` en `#greet` es solo una convención para mostrar que `greet` es un método de instancia (a diferencia de '.' Para un método de clase). RSpec no interpreta la cadena en absoluto, por lo que podría usar una cadena diferente u omitir ese bloque de `describe` completo.

Lea Empezando con rspec en línea: https://riptutorial.com/es/rspec/topic/2017/empezando-con-rspec

Capítulo 2: Expectativas de RSpec

Introducción

RSpec :: Expectations le permite expresar los resultados esperados en un objeto utilizando una sintaxis DSL basada en ejemplos.

Observaciones

Este tema proporciona ejemplos de cómo esperar los resultados de las pruebas en RSpec utilizando `expect .to` y los muchos emparejadores integrados.

Esta funcionalidad es proporcionada por la gema rspec-expectations .

Examples

Uso básico

Dada una `class` siguiente manera:

```
class Cube
  attr_reader :height, :width, :depth

  def initialize(args)
    @height = args[:height] || args[:y] || 1
    @width  = args[:width]  || args[:x] || 1
    @depth  = args[:depth]  || args[:z] || 1
  end

  def volume
    height * width * depth
  end
end
```

El siguiente ejemplo pasa si `cube.volume` es igual a 60 y falla si no lo hace. Utiliza el emparejador incorporado más comúnmente utilizado, `eq` , que solo prueba la igualdad.

```
RSpec.describe Cube do
  it "calculates it's volume" do
    cube = Cube.new(x: 3, y: 4, z: 5)
    expect(cube.volume).to eq(60)
  end
end
```

Lea Expectativas de RSpec en línea: https://riptutorial.com/es/rspec/topic/4304/expectativas-de-rspec

Capítulo 3: Mock RSpec

Observaciones

Este tema documenta el soporte de RSpec para pruebas de dobles (talones, simulacros, etc.). Ese soporte es proporcionado por la gema rspec-mocks .

Examples

Stubbing con permitir

En el siguiente ejemplo se utiliza `allow` y `receive` a stub un `Cart` llamada 's a una `CreditCardService` de manera que el ejemplo no tiene que esperar una llamada de red o utilizar un número de tarjeta de crédito que el procesador conoce.

```
class Cart
  def check_out
    begin
      transaction_id = CreditCardService.instance.validate credit_card_number, total_price
      order = Order.new
      order.items = cart.items
      order
    rescue CreditCardService::ValidationFailedError
      # handle the error
    end
  end
end

describe Cart do
  describe '#check_out' do
    it "places an order" do
      allow(CreditCardService.instance).
        to receive(:validate).with("1234567812345678", 3700).and_return("transaction_id")
      cart = Cart.new
      cart.items << Item.new("Propeller beanie", 3700)
      order = cart.check_out
      expect(order.transaction_id).to eq("transaction_id")
    end
  end
end
```

`with` es opcional; Sin ella, cualquier argumento es aceptado. `and_return` es opcional también; sin ella el talón vuelve a `nil` .

Burlándose de establecer una expectativa de mensaje con esperar

El siguiente ejemplo utiliza `expect` y `receive` para `CreditCardService` una llamada de una `Order` a un Servicio de `CreditCardService` , de modo que la prueba pase solo si la llamada se realiza sin tener que hacerlo realmente.

```
class Order
  def cancel
     CreditCardService.instance.refund transaction_id
  end
end

describe Order do
  describe '#cancel' do
    it "refunds the money" do
      order = Order.new
      order.transaction_id = "transaction_id"
      expect(CreditCardService.instance).to receive(:refund).with("transaction_id")
      order.cancel
    end
  end
end
```

En este ejemplo, el simulacro está en el valor de retorno de `CreditCardService.instance` , que probablemente sea un singleton.

`with` es opcional; sin ella, cualquier llamada al `refund` satisfaría las expectativas. Se puede dar un valor de retorno con `and_return` ; en este ejemplo no se usa, por lo que la llamada devuelve `nil` .

Lea Mock RSpec en línea: https://riptutorial.com/es/rspec/topic/5678/mock-rspec

Capítulo 4: RSpec Core

Examples

Ejecución de ejemplos con una etiqueta dada

Agregar etiquetas a los bloques "describir" o "it" le permite ejecutar solo esos ejemplos con una etiqueta determinada. Use la opción `--tag` (o `-t`) para ejecutar ejemplos que coincidan con una etiqueta específica. La etiqueta puede ser un nombre simple o un par nombre: valor.

- Si se proporciona un nombre simple, solo se ejecutarán los ejemplos con `:name => true` . Por ejemplo, `rspec <spec_file> --tag smoke` ejecutaría el ejemplo etiquetado con "Smoke".

```
describe '#Tests' do
  it 'runs the smoke test', :smoke => true do
  end

  it 'runs the regression tests', :regression => true do
  end

  it 'runs the acceptance tests', :acceptance => true do
  end
end
```

- Si se da un par `name:value` , se ejecutarán ejemplos con `name => value` , donde el valor siempre es una cadena. Por ejemplo, `rspec <spec_file> --tag testId:101` ejecutaría el ejemplo etiquetado con `testId` "101".

```
describe '#Tests' do
  it 'runs the test with id 99', :testId => 99 do
  end

  it 'runs the test with id 101', :testId => 101 do
  end
end
```

Lea RSpec Core en línea: https://riptutorial.com/es/rspec/topic/5672/rspec-core

Capítulo 5: RSpec Matcher

Introducción

rspec-expectations se envía con una serie de emparejadores incorporados. Cada emparejador se puede usar con expect (..). To or expect (..). Not_to para definir expectativas positivas y negativas respectivamente en un objeto.

Examples

Igualadores de igualdad

comparar utilizando eq `(==)`

```
RSpec.describe "a string" do
  it "is equal to another string of the same value" do
    expect("this string").to eq("this string")
  end

  it "is not equal to another string of a different value" do
    expect("this string").not_to eq("a different string")
  end
end

RSpec.describe "an integer" do
  it "is equal to a float of the same value" do
    expect(5).to eq(5.0)
  end
end
```

Cuando ejecuto `rspec` , la salida debe contener "3 ejemplos, 0 fallos"

comparar usando `==`

```
RSpec.describe "a string" do
  it "is equal to another string of the same value" do
    expect("this string").to be == "this string"
  end

  it "is not equal to another string of a different value" do
    expect("this string").not_to be == "a different string"
  end
end

RSpec.describe "an integer" do
  it "is equal to a float of the same value" do
    expect(5).to be == 5.0
  end
end
```

Cuando ejecuto `rspec` , la salida debe contener "3 ejemplos, 0 fallos"

comparar utilizando `eql (eql?)`

```
 RSpec.describe "an integer" do
  it "is equal to another integer of the same value" do
    expect(5).to eql(5)
  end

  it "is not equal to another integer of a different value" do
    expect(5).not_to eql(6)
  end

  it "is not equal to a float of the same value" do
    expect(5).not_to eql(5.0)
  end
end
```

Cuando ejecuto `rspec` , la salida debe contener "3 ejemplos, 0 fallos"

comparar usando `equal (equal?)`

```
RSpec.describe "a string" do
  it "is equal to itself" do
    string = "this string"
    expect(string).to equal(string)
  end

  it "is not equal to another string of the same value" do
    expect("this string").not_to equal("this string")
  end

  it "is not equal to another string of a different value" do
    expect("this string").not_to equal("a different string")
  end
end
```

Cuando ejecuto `rspec` , la salida debe contener "3 ejemplos, 0 fallos"

comparar usando ser `(equal?)`

```
RSpec.describe "a string" do
  it "is equal to itself" do
    string = "this string"
    expect(string).to be(string)
  end

  it "is not equal to another string of the same value" do
    expect("this string").not_to be("this string")
  end

  it "is not equal to another string of a different value" do
    expect("this string").not_to be("a different string")
  end
end
```

Cuando ejecuto `rspec` , la salida debe contener "3 ejemplos, 0 fallos"

Lea RSpec Matcher en línea: https://riptutorial.com/es/rspec/topic/10762/rspec-matcher

Creditos

S. No	Capítulos	Contributors
1	Empezando con rspec	adarsh, Ashish Bista, Community, Dave Schweisguth, gmuraleekrishna, Mark Huk, mbigras, Scott Matthewman, Simone Carletti, Srikanth Gurram, Vishnu Y S
2	Expectativas de RSpec	Ashish Bista, Dave Schweisguth, Midwire
3	Mock RSpec	Dave Schweisguth
4	RSpec Core	Dave Schweisguth, Srikanth Gurram
5	RSpec Matcher	Faruk Hossain