STUBS | MOCKS | SPIES

01

STUBS

STUBS

book_generator.rb

Raw

```ruby
class BookGenerator
  attr_reader :order_item, :book

  def initialize(order_item)
    @order_item = order_item
    @book = @order_item.book
  end

  def call
    images = ImageDownloader.call(book)
    composed_images = ImageComposer.call(images)
    pdf_file = PdfGenerator.call(composed_images)
    order_item.update(book_pdf: pdf_file)
  end
end
```

STUBS

book_generator_spec.rb

Raw

```ruby
describe BookGenerator do
  describe '#call' do
    it "generates and attaches a book's PDF file to the given order item" do
      order_item = OrderItem.create(type: 'Book', title: 'My big adventure', configuration: { ... })
      BookGenerator.new(order_item).call

      expect(order_item.book_pdf).not_to be_nil
    end
  end
end
```

STUBS

book_generator.rb

Raw

```ruby
class BookGenerator
  attr_reader :order_item, :book

  def initialize(order_item)
    @order_item = order_item
    @book = @order_item.book
  end

  def call
    images = ImageDownloader.call(book)
    composed_images = ImageComposer.call(images)
    pdf_file = PdfGenerator.call(composed_images)
    order_item.update(book_pdf: pdf_file)
  end
end
```

STUBS

book_generator_spec.rb

Raw

```ruby
describe BookGenerator do
  describe '#call' do
    it "generates and attaches a book's PDF file to the given order item" do
      order_item = OrderItem.create(type: 'Book', title: 'My big adventure', configuration: { ... })
      BookGenerator.new(order_item).call

      expect(order_item.book_pdf).not_to be_nil
    end
  end
end
```

STUBS

book_generator_spec.rb

Raw

```ruby
describe BookGenerator do
  describe '#call' do
    it "generates and attaches a book's PDF file to the given order item" do
      order_item = OrderItem.create(type: 'Book', title: 'My big adventure', configuration: { ... })

      allow(ImageDownloader).to receive(:call).and_return([{ page: 1, images: [] }])
      allow(ImageComposer).to receive(:call).and_return([])
      allow(PdfGenerator).to receive(:call).and_return(File.open('test_book.pdf', :r))

      BookGenerator.new(order_item).call
      expect(order_item.book_pdf).not_to be_nil
    end
  end
end
```

STUBS

book_generator.rb

Raw

```ruby
class BookGenerator
  attr_reader :order_item, :book

  def initialize(order_item)
    @order_item = order_item
    @book = @order_item.book
  end

  def call
    images = ImageDownloader.call(book)
    composed_images = ImageComposer.call(images)
    pdf_file = PdfGenerator.call(composed_images)
    order_item.update(book_pdf: pdf_file)
  end
end
```

Always returns [{ page: 1, content: ··· }]

Always returns []

Always returns a File

STUBS

- In computer science, test stubs are programs that simulate the behaviours of software components (or modules) that a module undergoing tests depends on
- They respond only with predefined answers!
- The idea is to make your module’s spec independent of other modules thus enabling you to test only the logic in the module under test
- You can write specs that pass if the modules you depend on don’t exist or are failing

articles_controller_spec.rb

Raw

```ruby
describe ArticlesController do
  describe '#create' do
    context 'passed valid params' do
      before do
        allow(Article).to receive(:save).and_return(true)
      end

      it 'redirects to the index page' do
        ...
      end
    end

    context 'passed invalid params' do
      before do
        allow(Article).to receive(:save).and_return(false)
      end

      it 'renders the new page again' do
        ...
      end
    end
  end
end
```

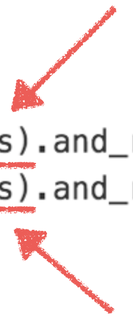
STUBS

articles_controller_spec.rb

Raw

```ruby
describe ArticlesController do
  describe '#create' do
    it "doesn't allow a second article to be created" do
      article_1_params = { ... }
      article_2_params = { ... }

      allow(Article).to receive(:create).with(article_1_params).and_return(true)
      allow(Article).to receive(:create).with(article_2_params).and_return(false)
    end
  end
end
```



STUBS

- Independent specs
- Faster specs
- Granular control of the execution path
- Easier collaboration

DOUBLES

book_generator_spec.rb

Raw

```ruby
describe BookGenerator do
  describe '#call' do
    it "generates and attaches a book's PDF file to the given order item" do
      order_item = OrderItem.create(type: 'Book', title: 'My big adventure', configuration: { ... })

      allow(ImageDownloader).to receive(:call).and_return([{ page: 1, images: [] }])
      allow(ImageComposer).to receive(:call).and_return([])
      allow(PdfGenerator).to receive(:call).and_return(File.open('test_book.pdf', :r))

      BookGenerator.new(order_item).call
      expect(order_item.book_pdf).not_to be_nil
    end
  end
end
```

DOUBLES

book_generator_spec.rb

Raw

```ruby
describe BookGenerator do
  describe '#call' do
    it "generates and attaches a book's PDF file to the given order item" do
      book = double(:book)
      order_item = double(:order_item)

      allow(order_item).to receive(:book).and_return(book)
      allow(order_item).to receive(:update).and_return(book)

      allow(ImageDownloader).to receive(:call).and_return([{ page: 1, images: [] }])
      allow(ImageComposer).to receive(:call).and_return([])
      allow(PdfGenerator).to receive(:call).and_return(File.open('test_book.pdf', :r))

      BookGenerator.new(order_item).call
      expect(order_item.book_pdf).not_to be_nil
    end
  end
end
```

DOUBLES

- Empty objects
- Their methods can only be stubbed
- Used in place of other objects (stunt doubles)

“With great power comes great responsibility.”

–Ben Parker

articles_controller_spec.rb

Raw

```ruby
describe ArticlesController do
  describe '#create' do
    it "creates a new object" do
      article_params = { ... }

      allow(Article).to receive(:create).and_return(true)

      expect(Article.create(article_params)).to eq(true)
    end
  end
end
```

articles_controller_spec.rb

Raw

```ruby
describe Dashboard do
  describe "#posts" do
    it "returns posts visible to the current user" do
      user = double('user')
      other_user = double('other user')
      published_one = double 'post', user: other_user, published: true, title: "published_one"
      published_two = double 'post', user: other_user, published: true, title: "published_two"
      unpublished = double 'post', user: other_user, published: false, title: "unpublished"
      visible_one = double 'post', user: user, published: false, title: "visible_one"
      visible_two = double 'post', user: user, published: false, title: "visible_two"

      posts = [published_one, published_two, unpublished, visible_one, visible_two]

      allow(Post).to receive(:all).and_return(posts)
      dashboard = Dashboard.new(posts: Post.all, user: user)

      allow(posts).to receive(:visible_to).with(user).and_return(posts - [unpublished])
      result = dashboard.posts

      expect(result.map(&:title)).to match_array(%w(
        published_one
        published_two
        visible_one
        visible_two
      ))
    end
  end
end
```

articles_controller_spec.rb

Raw

```ruby

describe Dashboard do
  describe "#posts" do
    it "returns posts visible to the current user" do
      user  = double('user')
      post  = double('visible post')
      posts = [post]

      dashboard = Dashboard.new(posts: posts, user: user)

      allow(posts).to receive(:visible_to).with(user).and_return(posts)
      expect(dashboard.posts).to match_array(posts)
    end
  end
end
```

02

MOCKS

MOCKS

book_generator_spec.rb

Raw

```ruby
describe BookGenerator do
  describe '#call' do
    it "generates and attaches a book's PDF file to the given order item" do
      book = double(:book)
      order_item = double(:order_item)

      allow(order_item).to receive(:book).and_return(book)
      allow(order_item).to receive(:update).and_return(book)

      allow(ImageDownloader).to receive(:call).and_return([{ page: 1, images: [] }])
      allow(ImageComposer).to receive(:call).and_return([])
      allow(PdfGenerator).to receive(:call).and_return(File.open('test_book.pdf', :r))

      BookGenerator.new(order_item).call
      expect(order_item.book_pdf).not_to be_nil
    end
  end
end
```

MOCKS

book_generator_spec.rb

Raw

```ruby
describe BookGenerator do
  describe '#call' do
    it "generates and attaches a book's PDF file to the given order item" do
      book = double(:book)
      order_item = double(:order_item)

      allow(order_item).to receive(:book).and_return(book)
      expect(order_item).to receive(:update).and_return(true)

      allow(ImageDownloader).to receive(:call).and_return([{ page: 1, images: [] }])
      allow(ImageComposer).to receive(:call).and_return([])
      allow(PdfGenerator).to receive(:call).and_return(File.open('test_book.pdf', :r))

      BookGenerator.new(order_item).call
    end
  end
end
```

MOCKS

- They are very similar to stubs
- The key difference is that the spec will check if a mock has been called
- Stubs that have to be executed
- Shares the stub syntax

03

SPIES

SPIES

book_generator_spec.rb

Raw

```ruby
describe BookGenerator do
  describe '#call' do
    it "generates and attaches a book's PDF file to the given order item" do
      book = double(:book)
      order_item = double(:order_item)

      allow(order_item).to receive(:book).and_return(book)
      expect(order_item).to receive(:update).and_return(true)

      allow(ImageDownloader).to receive(:call).and_return([{ page: 1, images: [] }])
      allow(ImageComposer).to receive(:call).and_return([])
      allow(PdfGenerator).to receive(:call).and_return(File.open('test_book.pdf', :r))

      BookGenerator.new(order_item).call
    end
  end
end
```

SPIES

book_generator_spec.rb

Raw

```ruby
describe BookGenerator do
  describe '#call' do
    it "generates and attaches a book's PDF file to the given order item" do
      book = double(:book)
      order_item = double(:order_item)

      allow(order_item).to receive(:book).and_return(book)
      allow(order_item).to receive(:update).and_return(true)

      allow(ImageDownloader).to receive(:call).and_return([{ page: 1, images: [] }])
      allow(ImageComposer).to receive(:call).and_return([])
      allow(PdfGenerator).to receive(:call).and_return(File.open('test_book.pdf', :r))

      BookGenerator.new(order_item).call
      expect(order_item).to have_received(:update)
    end
  end
end
```

SPIES

spy.rb

Raw

```ruby
describe '#have_received' do
  it 'expects deliver to get called' do
    invitation = spy('invitation')
    invitation.deliver
    expect(invitation).to have_received(:deliver)
  end
end
```

SPIES

- More explicit version of mocks
- Have a different syntax

04 FAKES

FAKES

checkout_controller_spec.rb

Raw

```ruby

describe CheckoutController do
  describe "#checkout" do
    it 'registers the transaction' do
      allow(PaymentProcessor).to receive(:register_payment).and_return({ ... })

      ...
    end
  end
end
```

FAKES

```
85    def self.verify_all_cards!
86      self.verify_all_cards = true
87    end
88
89    def self.generate_transaction(options = {})
90      history_item = {
91       'timestamp' => Time.now,
92       'amount' => options[:amount],
93       'status' => options[:status]
94      }
95      created_at = options[:created_at] || Time.now
96      {
97        'status_history' => [history_item],
98        'subscription_id' => options[:subscription_id],
99        'created_at' => created_at,
100       'amount' => options[:amount]
101     }
102   end
```

README.md

New maintainer

thoughtbot stopped using Braintree but wanted this library to live on.

It was transferred on May 7th, 2015 to High Fidelity.

We hope to soon start tackling the number one outstanding issue - support for Braintree's v.zero API.

fake_braintree, a Braintree fake build failing

This library is a way to test Braintree code without hitting Braintree's servers. It uses Capybara::Server to intercept all of the calls from Braintree's Ruby library and returns XML that the Braintree library can parse. The whole point is not to hit the Braintree API.

It supports a lot of Braintree methods, but it does not support every single one of them (yet).

Supported API methods

Address

- `Braintree::Address.create`

ClientToken

FAKES

- Additional code
- You need specs for your specs
- Unexpected behaviour
- Use webmock or VCR instead

THANKS FOR LISTENING

stanko.krtalic@gmail.com
@monorkin