

Stanko Krtalić Rusendić  
Ruby on Rails engineer @ Infinum

—

[stanko.krtalic@gmail.com](mailto:stanko.krtalic@gmail.com)

@monorkin

<https://github.com/Stankec>

STUBS | MOCKS | SPIES



S<sup>1</sup>TUBS

# STUBS

book\_generator.rb

Raw

```
1 class BookGenerator
2   attr_reader :order_item, :book
3
4   def initialize(order_item)
5     @order_item = order_item
6     @book = @order_item.book
7   end
8
9   def call
10    images = ImageDownloader.call(book)
11    composed_images = ImageComposer.call(images)
12    pdf_file = PdfGenerator.call(composed_images)
13    order_item.update(book_pdf: pdf_file)
14  end
15 end
```

# STUBS

book\_generator\_spec.rb

Raw

```
1 describe BookGenerator do
2   describe '#call' do
3     it "generates and attaches a book's PDF file to the given order item" do
4       order_item = OrderItem.create(type: 'Book', title: 'My big adventure', configuration: { ... })
5       BookGenerator.new(order_item).call
6
7       expect(order_item.book_pdf).not_to be_nil
8     end
9   end
10 end
```

# STUBS

book\_generator.rb

Raw

```
1 class BookGenerator
2   attr_reader :order_item, :book
3
4   def initialize(order_item)
5     @order_item = order_item
6     @book = @order_item.book
7   end
8
9   def call
10    → images = ImageDownloader.call(book)
11    → composed_images = ImageComposer.call(images)
12    → pdf_file = PdfGenerator.call(composed_images)
13    → order_item.update(book_pdf: pdf_file)
14  end
15 end
```

# STUBS

book\_generator\_spec.rb

Raw

```
1 describe BookGenerator do
2   describe '#call' do
3     it "generates and attaches a book's PDF file to the given order item" do
4       order_item = OrderItem.create(type: 'Book', title: 'My big adventure', configuration: { ... })
5       BookGenerator.new(order_item).call
6
7       expect(order_item.book_pdf).not_to be_nil
8     end
9   end
10 end
```

# STUBS

book\_generator\_spec.rb

Raw

```
1 describe BookGenerator do
2   describe '#call' do
3     it "generates and attaches a book's PDF file to the given order item" do
4       order_item = OrderItem.create(type: 'Book', title: 'My big adventure', configuration: { ... })
5
6       → allow(ImageDownloader).to receive(:call).and_return([{ page: 1, images: [] }])
7       → allow(ImageComposer).to receive(:call).and_return([])
8       → allow(PdfGenerator).to receive(:call).and_return(File.open('test_book.pdf', :r))
9
10    BookGenerator.new(order_item).call
11    expect(order_item.book_pdf).not_to be_nil
12  end
13 end
14 end
```

# STUBS

book\_generator.rb

Raw

```
1 class BookGenerator
2   attr_reader :order_item, :book
3
4   def initialize(order_item)
5     @order_item = order_item
6     @book = @order_item.book
7   end
8
9   def call
10    images = ImageDownloader.call(book)
11    composed_images = ImageComposer.call(images)
12    pdf_file = PdfGenerator.call(composed_images)
13    order_item.update(book_pdf: pdf_file)
14  end
15 end
```

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

Always returns []

Always returns a File

[articles\\_controller\\_spec.rb](#)

Raw

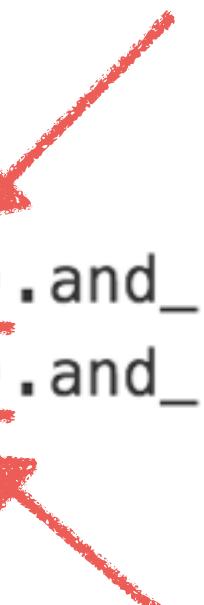
```
1 describe ArticlesController, type: :controller do
2   describe '#create' do
3     context 'passed valid params' do
4       before do
5          allow(Article).to receive(:create).and_return(true)
6       end
7
8       it 'redirects to the index page' do
9         ...
10      end
11    end
12
13    context 'passed invalid params' do
14      before do
15         allow(Article).to receive(:create).and_return(false)
16      end
17
18      it 'renders the new template' do
19        ...
20      end
21    end
22  end
23 end
```

# STUBS

articles\_controller\_spec.rb

Raw

```
1 describe ArticlesController do
2   describe '#create' do
3     it "doesn't allow a second article to be created" do
4       article_1_params = { ... }
5       article_2_params = { ... }
6
7       allow(Article).to receive(:create).with(article_1_params).and_return(true)
8       allow(Article).to receive(:create).with(article_2_params).and_return(false)
9     end
10    end
11  end
```



# DOUBLES

book\_generator\_spec.rb

Raw

```
1 describe BookGenerator do
2   describe '#call' do
3     it "generates and attaches a book's PDF file to the given order item" do
4       order_item = OrderItem.create(type: 'Book', title: 'My big adventure', configuration: { ... })
5
6       allow(ImageDownloader).to receive(:call).and_return([{ page: 1, images: [] }])
7       allow(ImageComposer).to receive(:call).and_return([])
8       allow(PdfGenerator).to receive(:call).and_return(File.open('test_book.pdf', :r))
9
10      BookGenerator.new(order_item).call
11      expect(order_item.book_pdf).not_to be_nil
12    end
13  end
14end
```

# DOUBLES

book\_generator\_spec.rb

Raw

```
1 describe BookGenerator do
2   describe '#call' do
3     it "generates and attaches a book's PDF file to the given order item" do
4       book = double(:book)
5       order_item = double(:order_item)
6
7       allow(order_item).to receive(:book).and_return(book)
8       allow(order_item).to receive(:update).and_return(book)
9
10      allow(ImageDownloader).to receive(:call).and_return([{ page: 1, images: [] }])
11      allow(ImageComposer).to receive(:call).and_return([])
12      allow(PdfGenerator).to receive(:call).and_return(File.open('test_book.pdf', :r))
13
14      BookGenerator.new(order_item).call
15      expect(order_item.book_pdf).not_to be_nil
16    end
17  end
18 end
```



MOCKS

# MOCKS

book\_generator\_spec.rb

Raw

```
1 describe BookGenerator do
2   describe '#call' do
3     it "generates and attaches a book's PDF file to the given order item" do
4       book = double(:book)
5       order_item = double(:order_item)
6
7       allow(order_item).to receive(:book).and_return(book)
8       allow(order_item).to receive(:update).and_return(book)
9
10      allow(ImageDownloader).to receive(:call).and_return([{ page: 1, images: [] }])
11      allow(ImageComposer).to receive(:call).and_return([])
12      allow(PdfGenerator).to receive(:call).and_return(File.open('test_book.pdf', :r))
13
14      BookGenerator.new(order_item).call
15      expect(order_item.book_pdf).not_to be_nil
16    end
17  end
18 end
```

# MOCKS

book\_generator\_spec.rb

Raw

```
1 describe BookGenerator do
2   describe '#call' do
3     it "generates and attaches a book's PDF file to the given order item" do
4       book = double(:book)
5       order_item = double(:order_item)
6
7       allow(order_item).to receive(:book).and_return(book)
8       →expect(order_item).to receive(:update).and_return(true)
9
10      allow(ImageDownloader).to receive(:call).and_return([{ page: 1, images: [] }])
11      allow(ImageComposer).to receive(:call).and_return([])
12      allow(PdfGenerator).to receive(:call).and_return(File.open('test_book.pdf', :r))
13
14      BookGenerator.new(order_item).call
15    end
16  end
17 end
```

# MOCKS | STUBS

	Query	Command
Incoming	assert return value	assert side effects
Outgoing	do not assert	assert message sent

# MOCKS | STUBS

	Returns value	Changes state
Own method	assert return value	assert side effects
External method	do not assert	assert message sent

# MOCKS | STUBS

		Returns value	Changes state
Own method	assert	assert	
External method	stub	mock	

# MOCKS | STUBS

book\_generator.rb

```
1 class BookGenerator
2   attr_reader :order_item, :book
3
4   def initialize(order_item)
5     @order_item = order_item
6     @book = @order_item.book
7   end
8
9   def call
10    images = ImageDownloader.call(book) -> Stub
11    composed_images = ImageComposer.call(images) -> Stub
12    pdf_file = PdfGenerator.call(composed_images)
13    order_item.update(book_pdf: pdf_file) -> Mock
14  end
15 end
```

Raw

# MOCKS | STUBS

Isolated testing  
Separation of concerns  
Speed



SPIES

# SPIES

book\_generator\_spec.rb

Raw

```
1 describe BookGenerator do
2   describe '#call' do
3     it "generates and attaches a book's PDF file to the given order item" do
4       book = double(:book)
5       order_item = double(:order_item)
6
7       allow(order_item).to receive(:book).and_return(book)
8       →expect(order_item).to receive(:update).and_return(true)
9
10      allow(ImageDownloader).to receive(:call).and_return([{ page: 1, images: [] }])
11      allow(ImageComposer).to receive(:call).and_return([])
12      allow(PdfGenerator).to receive(:call).and_return(File.open('test_book.pdf', :r))
13
14      BookGenerator.new(order_item).call
15    end
16  end
17 end
```

# SPIES

book\_generator\_spec.rb

Raw

```
1 describe BookGenerator do
2   describe '#call' do
3     it "generates and attaches a book's PDF file to the given order item" do
4       book = double(:book)
5       order_item = double(:order_item)
6
7       allow(order_item).to receive(:book).and_return(book)
8       →allow(order_item).to receive(:update).and_return(true)
9
10      allow(ImageDownloader).to receive(:call).and_return([{ page: 1, images: [] }])
11      allow(ImageComposer).to receive(:call).and_return([])
12      allow(PdfGenerator).to receive(:call).and_return(File.open('test_book.pdf', :r))
13
14      BookGenerator.new(order_item).call
15      →expect(order_item).to have_received(:update)
16    end
17  end
18 end
```

# SPIES

spy.rb

```
1 describe '#have_received' do
2   it 'expects deliver to get called' do
3     invitation = spy('invitation')
4     invitation.deliver
5     expect(invitation).to have_received(:deliver)
6   end
7 end
```

Raw



FAKES

# FAKES

checkout\_controller\_spec.rb

Raw

```
1 describe CheckoutController do
2   describe "#checkout" do
3     it 'registers the transaction' do
4       allow(PaymentProcessor).to receive(:register_payment).and_return({ ... })
5
6       ...
7     end
8   end
9 end
10 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
```

# FAKES

☰ 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



PROBLEMS

“With great power comes great responsibility.”

–Ben Parker

[articles\\_controller\\_spec.rb](#)

Raw

```
1 describe ArticlesController do
2   describe '#create' do
3     it "creates a new object" do
4       article_params = { ... }
5
6       allow(Article).to receive(:create).and_return(true)
7
8       expect(Article.create(article_params)).to eq(true)
9     end
10    end
11  end
```

[articles\\_controller\\_spec.rb](#)

Raw

```
1 describe Dashboard do
2   describe "#posts" do
3     it "returns posts visible to the current user" do
4       user = double('user')
5       other_user = double('other user')
6       published_one = double 'post', user: other_user, published: true, title: "published_one"
7       published_two = double 'post', user: other_user, published: true, title: "published_two"
8       unpublished = double 'post', user: other_user, published: false, title: "unpublished"
9       visible_one = double 'post', user: user, published: false, title: "visible_one"
10      visible_two = double 'post', user: user, published: false, title: "visible_two"
11
12      posts = [published_one, published_two, unpublished, visible_one, visible_two]
13
14      allow(Post).to receive(:all).and_return(posts)
15      dashboard = Dashboard.new(posts: Post.all, user: user)
16
17      allow(posts).to receive(:visible_to).with(user).and_return(posts - [unpublished])
18      result = dashboard.posts
19
20      expect(result.map(&:title)).to match_array(%w(
21        published_one
22        published_two
23        visible_one
24        visible_two
25      ))
26    end
27  end
28 end
```

[articles\\_controller\\_spec.rb](#)

Raw

```
1 describe Dashboard do
2   describe "#posts" do
3     it "returns posts visible to the current user" do
4       user = double('user')
5       post = double('visible post')
6       posts = [post]
7
8       dashboard = Dashboard.new(posts: posts, user: user)
9
10      allow(posts).to receive(:visible_to).with(user).and_return(posts)
11      expect(dashboard.posts).to match_array(posts)
12    end
13  end
14end
15end
```



THANKS FOR LISTENING