Bag o' Tricks: Unit Testing

Anton 'tony' Bangratz
https://abangratz.github.com/
@tony_xpro
anton.bangratz@radarservices.com

2012-01-18

Introduction

The Art of Testing

Not (only) Rails

One Method, One Purpose

Consider:

```
class TitanClass
  .
  .
  .
  def doIt(*args)
    args[:external_sources].each do |source|
      args[:data_gatherers] do |gatherer|
        gatherer.doIt(source)
      end
    end
    processed_data = []
    current_gatherers.each do |gatherer|
      processed_data << process_data(gatherer)
    end
    reports = {}
    processed_data.each do |data|
      reports[data.id] = Report.new(data)
    end
    ReportPrinter.print_all(reports)
  end
end
```

One Method, One Purpose

Test:

```
require 'spec_helper'

describe TitanClass do
  it "should do everything" do
    source = double(Source)
    gatherer = double(Gatherer)
    args = {external_sources: [source],
      data_gatherers: [gatherer]}
    processed = double("ProcessedData")
    subject.should_receive(process_data)
      .with(gatherer).and_return(processed)
    report = double(Report)
    Report.should_receive(:new).with(processed).
      and_return(report)
    ReportPrinter.should_receive(:print_all).with([report])
    subject.doIt(args)
  end
end
```

One Method, One Purpose

Test:

```
require 'spec_helper'

describe TitanClass do
  it "should do everything" do
    source = double(Source)
    gatherer = double(Gatherer)
    args = {external_sources: [source],
      data_gatherers: [gatherer]}
    processed = double("ProcessedData")
    subject.should_receive(process_data)
      .with(gatherer).and_return(processed)
    # Yikes, a stub on the subject under test!
    report = double(Report)
    Report.should_receive(:new).with(processed).
      and_return(report)
    # Stubbing ".new" is rarely a good idea
    ReportPrinter.should_receive(:print_all).with([report])
    # Huh. We know that a ReportPrinter prints reports,
    # but that's more than we actually should know ...
    subject.doIt(args)
  end
```

One Test, One Assertion

Example

```
describe SmartClass do
  it "invokes the data gatherer" do
    data_gatherer = double(DataGatherer)
    source = double(Source)
    data_gatherer.should_receive(gather).with(source))
    # Our only assertion!
    subject.data_gatherer = data_gatherer # DI for the win
    subject.gather_from_source(source)
  end end
...
class SmartClass

  attr_accessor :data_gatherer

  def gather_from_source(source)
    data_gatherer_gather(source)
  end
end
```

Mocking and Stubbing

Testing Modules

Example

```
class TestSmartModule
  include SmartModule
end
describe SmartModule do
  subject { TestSmartModule.new }
  it "does something" do
    subject.something.should ...
  end
end
```

Testing IO

Well, Rails

Anonymous Controllers

Example

```
describe SomeClass, type: 'controller' do
  controller(ApplicationControllerOrSubClass) do
    def index
      ... # do something
    end
  end

  it "makes something happen" do
    ... # assert something happens
    get :index
  end
end
```

Testing Models?

Another Layer

TDD

Frameworks

MiniTest or RSpec?

That’s all, folks.

Thanks

- Stephan Kämper
- Lisa Crispin
- Elisabeth Hendricks
- Michael Bolton
- Avdi Grimm
- Steve Klabnik
- Jeremy Evans
- Martin Gamsjäger
- Robert C. Martin (Uncle Bob)
- Andreas Kopecky
- Jürgen Strobel
- Andreas Tiefenthaler
- Michael Kohl
- Floor Drees
- Friends and colleagues that inspired me