A Tour of Rails Testing... Sorta

By Desi McAdam

A Tour of Rails Testing - Using RSpec Framework

Agenda

- What is RSpec
- How to get up and running with RSpec
- RSpec generators
- Quick detour to cover fixtures - high level
- Describe, it
- Model Spec - Mocks and Stubs
- Controller Specs
- Before()
- Output Formats

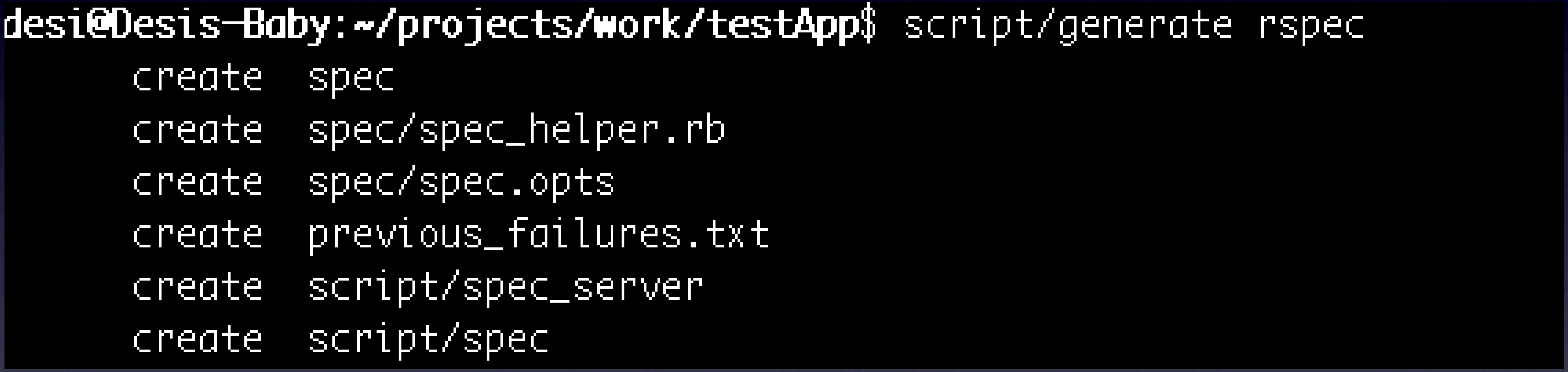
What the hell is RSpec?

- RSpec is a domain specific language written in Ruby and used to specify expected behavior of your app.

Getting RSpec running

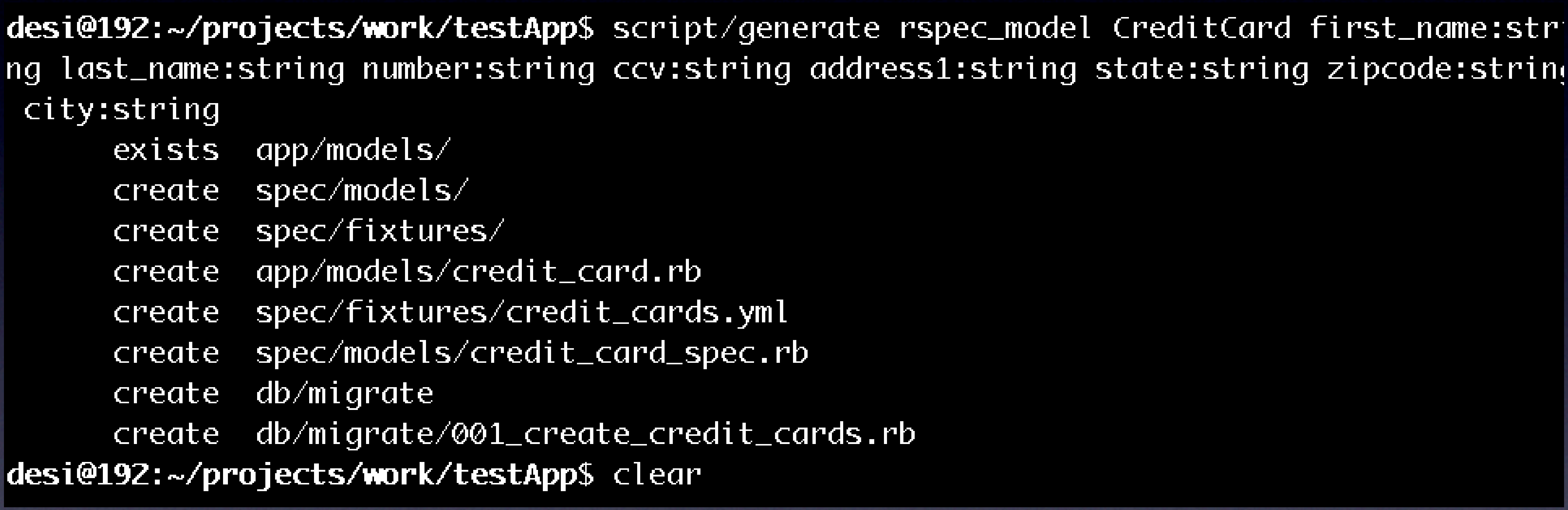
- Get RSpec the plugin
- Get RSpec on Rails plugin
- Don’t bother with the gem.
- Then run script/generate rspec - This is the drop in replacement part.
- Then delete the test folder if you want. You don’t really need it anymore.

script/generate rspec



```
desi@Desis-Baby:~/projects/work/testApp$ script/generate rspec
      create  spec
      create  spec/spec_helper.rb
      create  spec/spec.opts
      create  previous_failures.txt
      create  script/spec_server
      create  script/spec
```

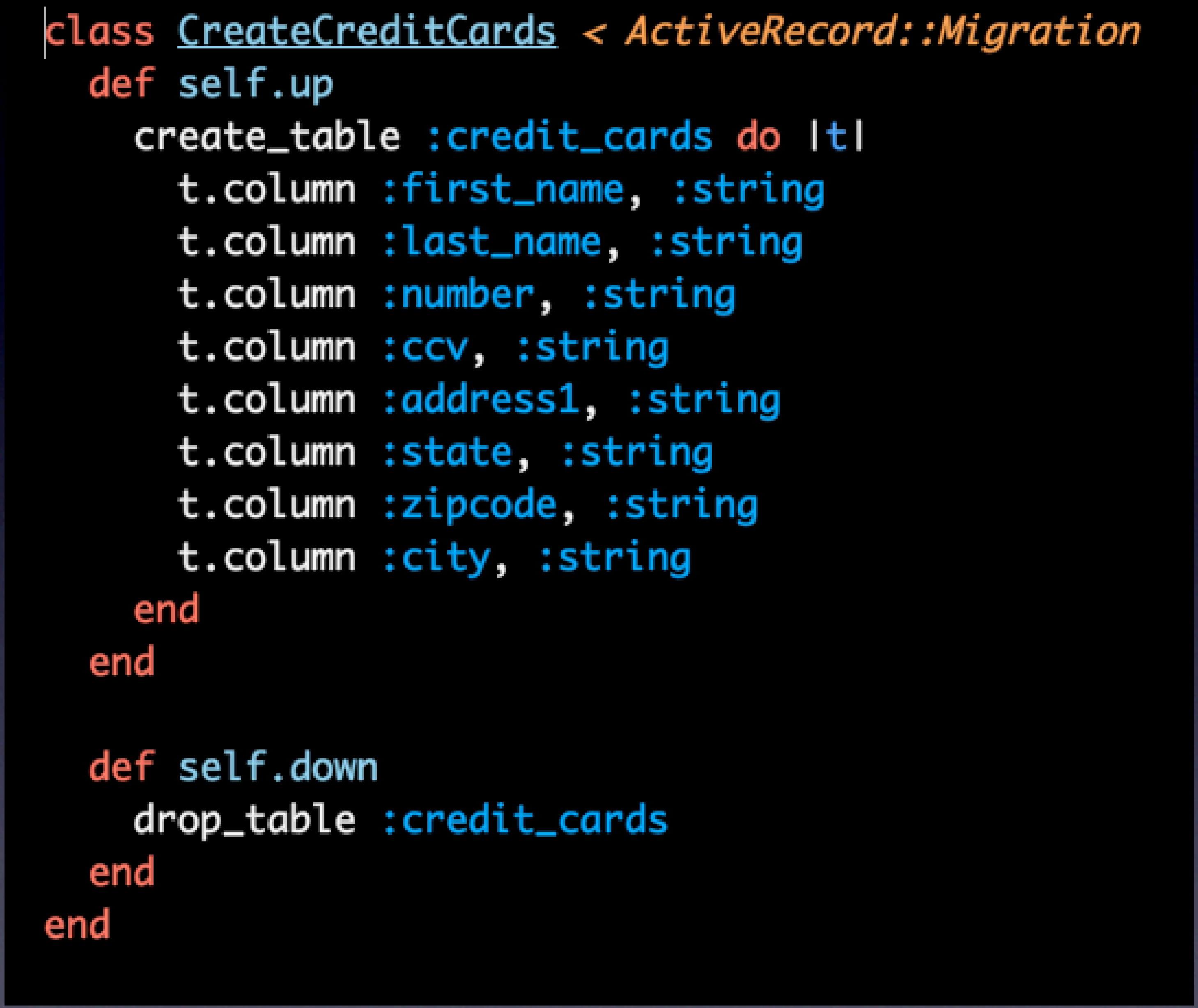
RSpec Model Generator



```
desi@192:~/projects/work/testApp$ script/generate rspec_model CreditCard first_name:stri
ng last_name:string number:string ccv:string address1:string state:string zipcode:string
 city:string
      exists  app/models/
      create  spec/models/
      create  spec/fixtures/
      create  app/models/credit_card.rb
      create  spec/fixtures/credit_cards.yml
      create  spec/models/credit_card_spec.rb
      create  db/migrate
      create  db/migrate/001_create_credit_cards.rb
desi@192:~/projects/work/testApp$ clear
```

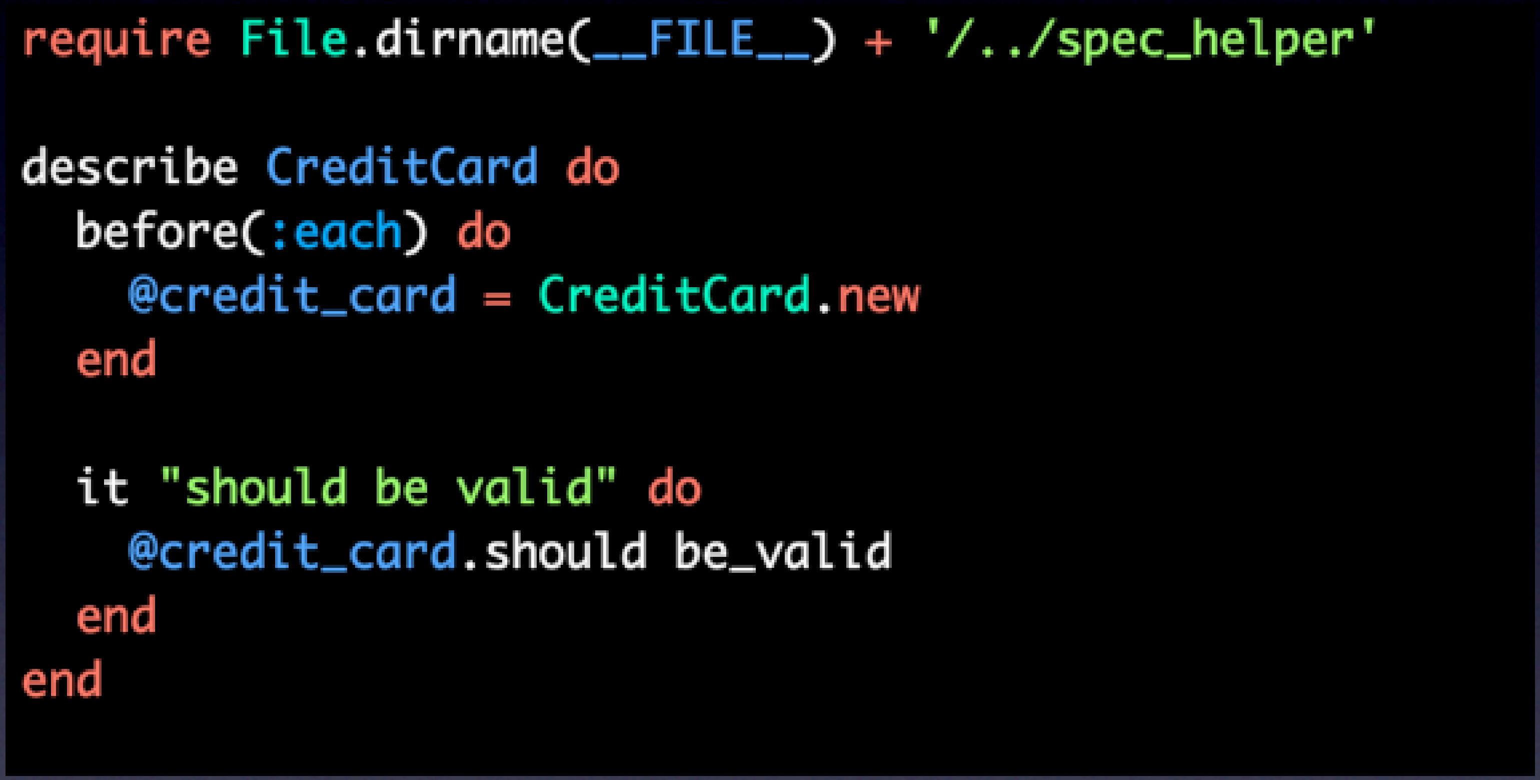
- You can use rspec_model generator instead of the default model generator.

migration



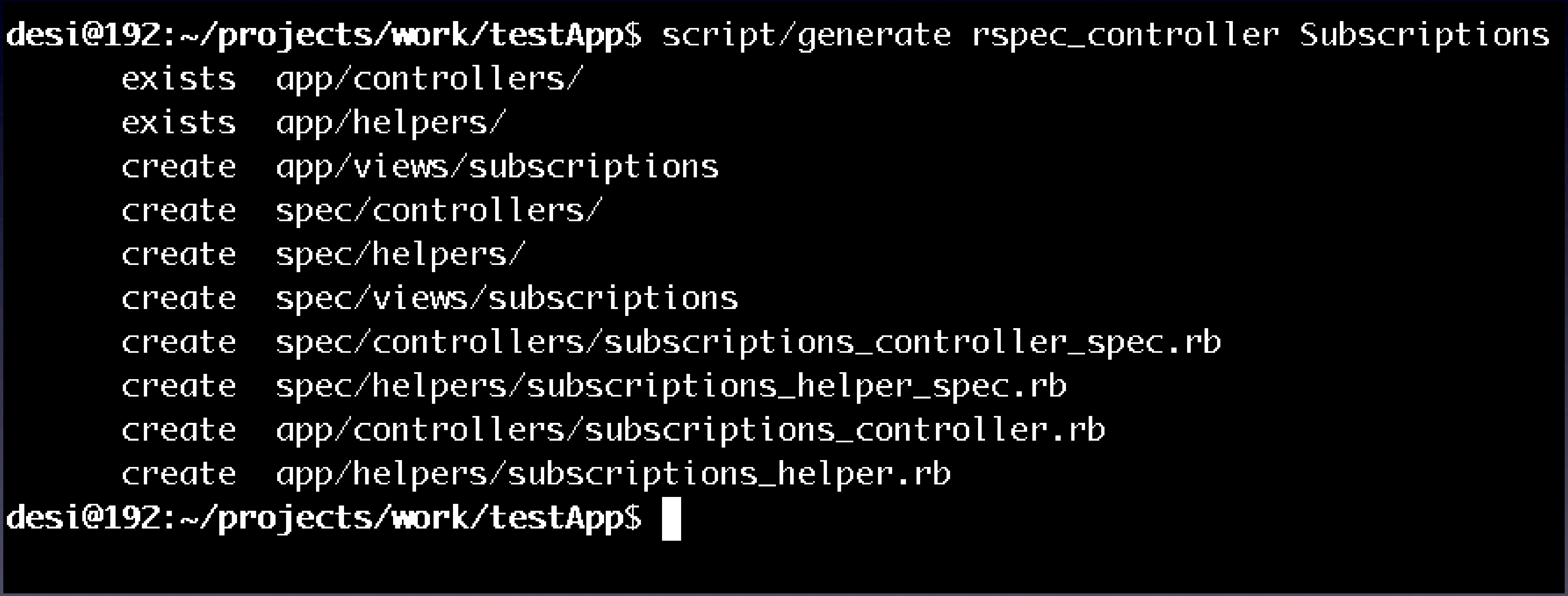
```
class CreateCreditCards < ActiveRecord::Migration
  def self.up
    create_table :credit_cards do |t|
      t.column :first_name, :string
      t.column :last_name, :string
      t.column :number, :string
      t.column :ccv, :string
      t.column :address1, :string
      t.column :state, :string
      t.column :zipcode, :string
      t.column :city, :string
    end
  end

  def self.down
    drop_table :credit_cards
  end
end
```

spec generated



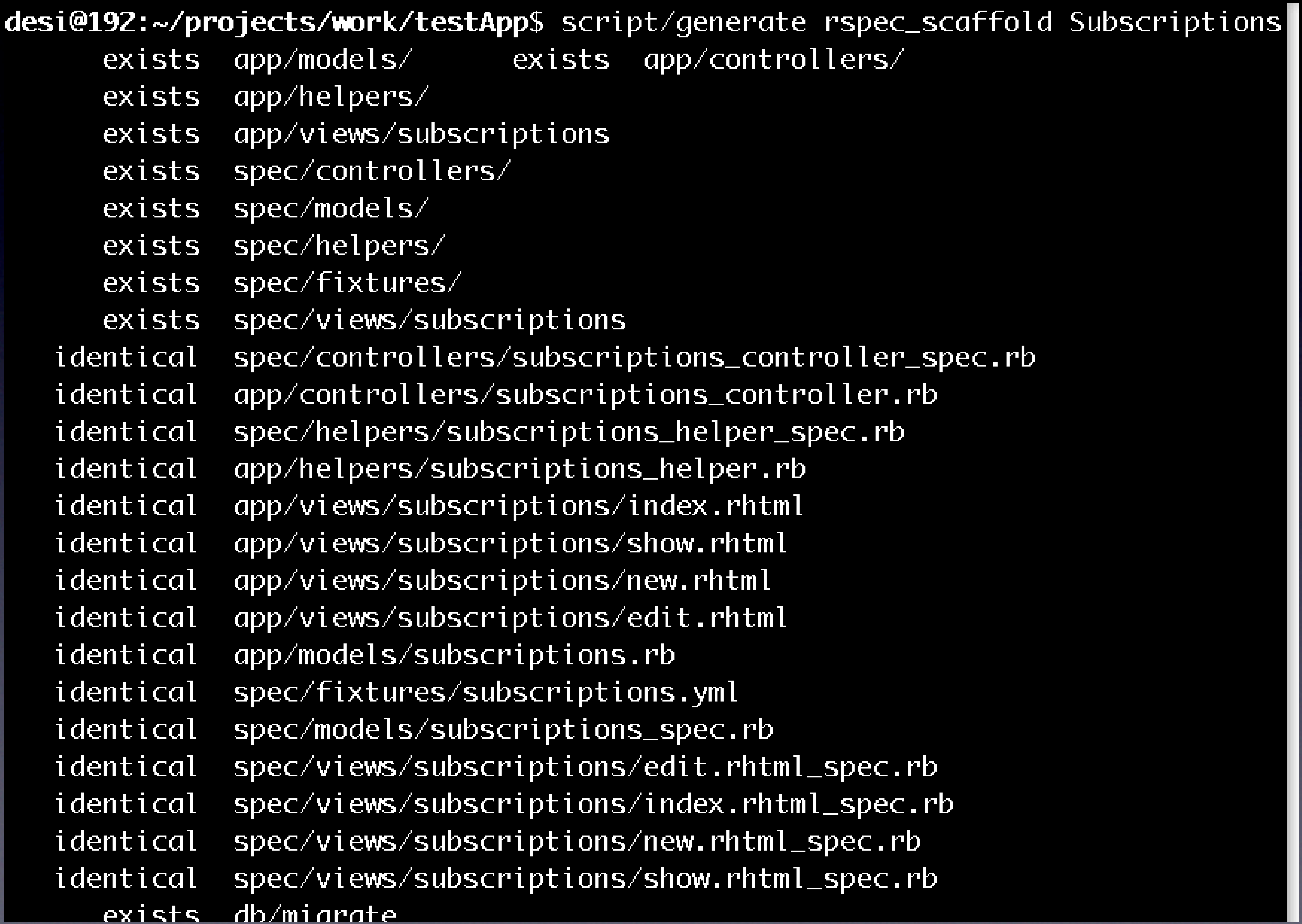
```ruby
require File.dirname(__FILE__) + '/../spec_helper'

describe CreditCard do
  before(:each) do
    @credit_card = CreditCard.new
  end

  it "should be valid" do
    @credit_card.should be_valid
  end
end
```

RSpec Controller Generator



```
desi@192:~/projects/work/testApp$ script/generate rspec_controller Subscriptions
      exists  app/controllers/
      exists  app/helpers/
      create  app/views/subscriptions
      create  spec/controllers/
      create  spec/helpers/
      create  spec/views/subscriptions
      create  spec/controllers/subscriptions_controller_spec.rb
      create  spec/helpers/subscriptions_helper_spec.rb
      create  app/controllers/subscriptions_controller.rb
      create  app/helpers/subscriptions_helper.rb
desi@192:~/projects/work/testApp$ 
```

Yep even scaffold works



```
desi@192:~/projects/work/testApp$ script/generate rspec_scaffold Subscriptions
      exists  app/models/       exists  app/controllers/
      exists  app/helpers/
      exists  app/views/subscriptions
      exists  spec/controllers/
      exists  spec/models/
      exists  spec/helpers/
      exists  spec/fixtures/
      exists  spec/views/subscriptions
   identical  spec/controllers/subscriptions_controller_spec.rb
   identical  app/controllers/subscriptions_controller.rb
   identical  spec/helpers/subscriptions_helper_spec.rb
   identical  app/helpers/subscriptions_helper.rb
   identical  app/views/subscriptions/index.rhtml
   identical  app/views/subscriptions/show.rhtml
   identical  app/views/subscriptions/new.rhtml
   identical  app/views/subscriptions/edit.rhtml
   identical  app/models/subscriptions.rb
   identical  spec/fixtures/subscriptions.yml
   identical  spec/models/subscriptions_spec.rb
   identical  spec/views/subscriptions/edit.rhtml_spec.rb
   identical  spec/views/subscriptions/index.rhtml_spec.rb
   identical  spec/views/subscriptions/new.rhtml_spec.rb
   identical  spec/views/subscriptions/show.rhtml_spec.rb
      exists  db/migrate
```

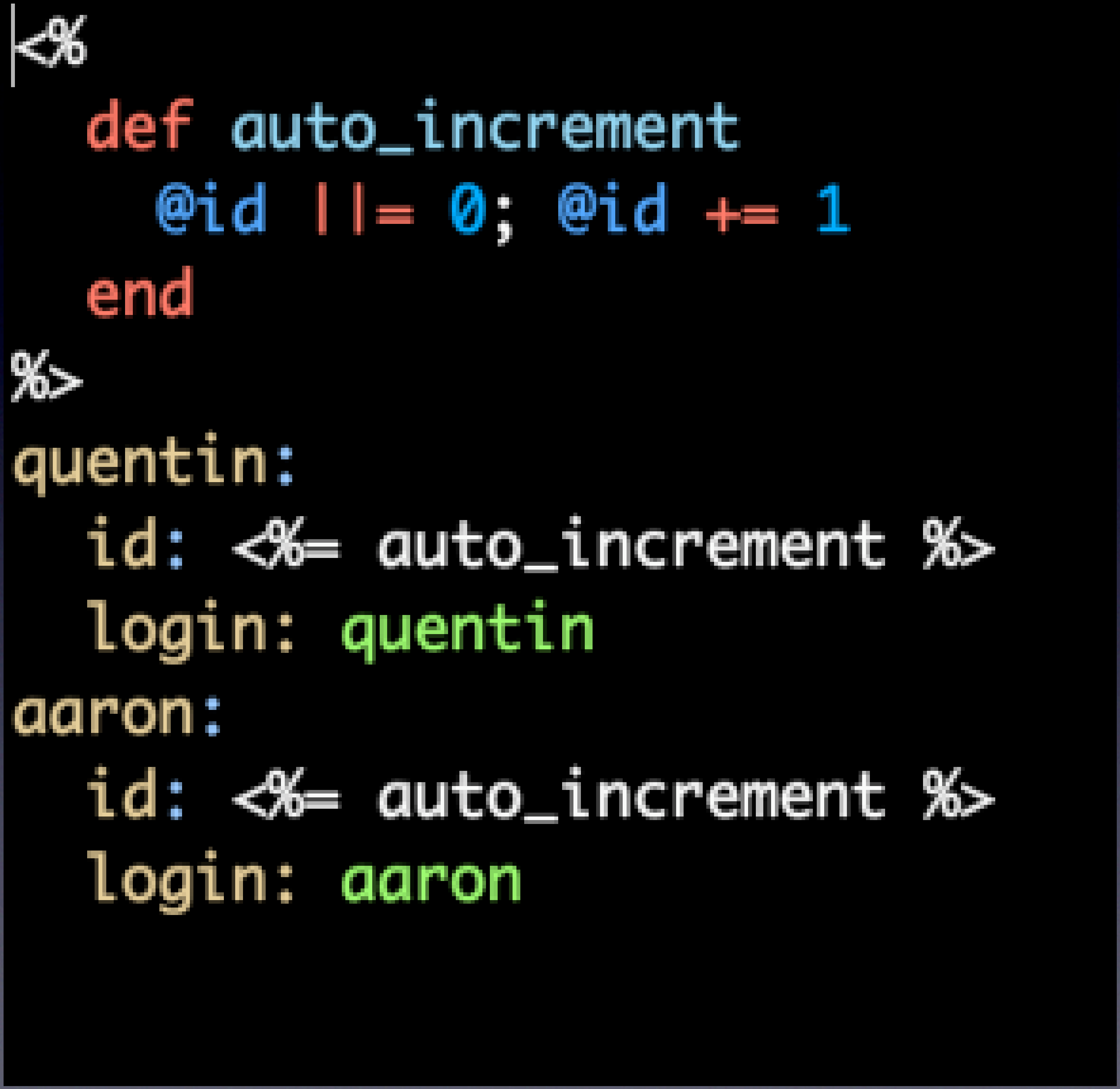
Fixtures

- Have .yml extentions
- Are created when you use Rails Scaffolding as well as when you use RSpec Scaffolding.
- Are two dimensional serialized hashes

```
quentin:
  id: 1
  username: quentin
  first_name: Quentin
  last_name: Tarantino
  email: quentin@example.com
  crypted_password: <%= Person.encrypt('test') %>
  created_at: <%= 5.days.ago.to_s :db %>
  activation_code: 8f24789ae988411ccf33ab0c30fe9106fab32e9b
  activated: true
  admin: true
  city_id: 1
  birthday: <%= 25.years.ago.to_s :db %>
  gender: m
```

Some maybe cool stuff about Fixtures

- You can have fixtures in csv format as well.
- You can include dynamic content in fixtures.



```
<%
  def auto_increment
    @id ||= 0; @id += 1
  end
%>
quentin:
  id: <%= auto_increment %>
  login: quentin
aaron:
  id: <%= auto_increment %>
  login: aaron
```

- If you wanted to add something like auto_increment from the previous example to all your fixtures then you can create a fixture_helpers.rb file to the lib dir of your project directory. Then make sure to do a require ‘fixture_helpers’ at the top of your spec_helper.rb file or test_helper.rb file.

Fixtures Couple of Thoughts

- Once your apps get big
  - Fixtures are Slow
  - Fixtures are Messy
  - Fixtures can be hard to maintain

Back to RSpec

Describe

- The describe method creates an instance of Behavior.
    - In other words “describe CreditCard” should read in your mind as “Describe the Behavior of the CreditCard Class”

it should

- The it method creates an Example. “Describe the behavior of CreditCard class.” with the examples
  - It should raise an error if the credit card is not valid
  - It should authorize and capture authorization amount if successful
  - It should raise an error if the authorization fails

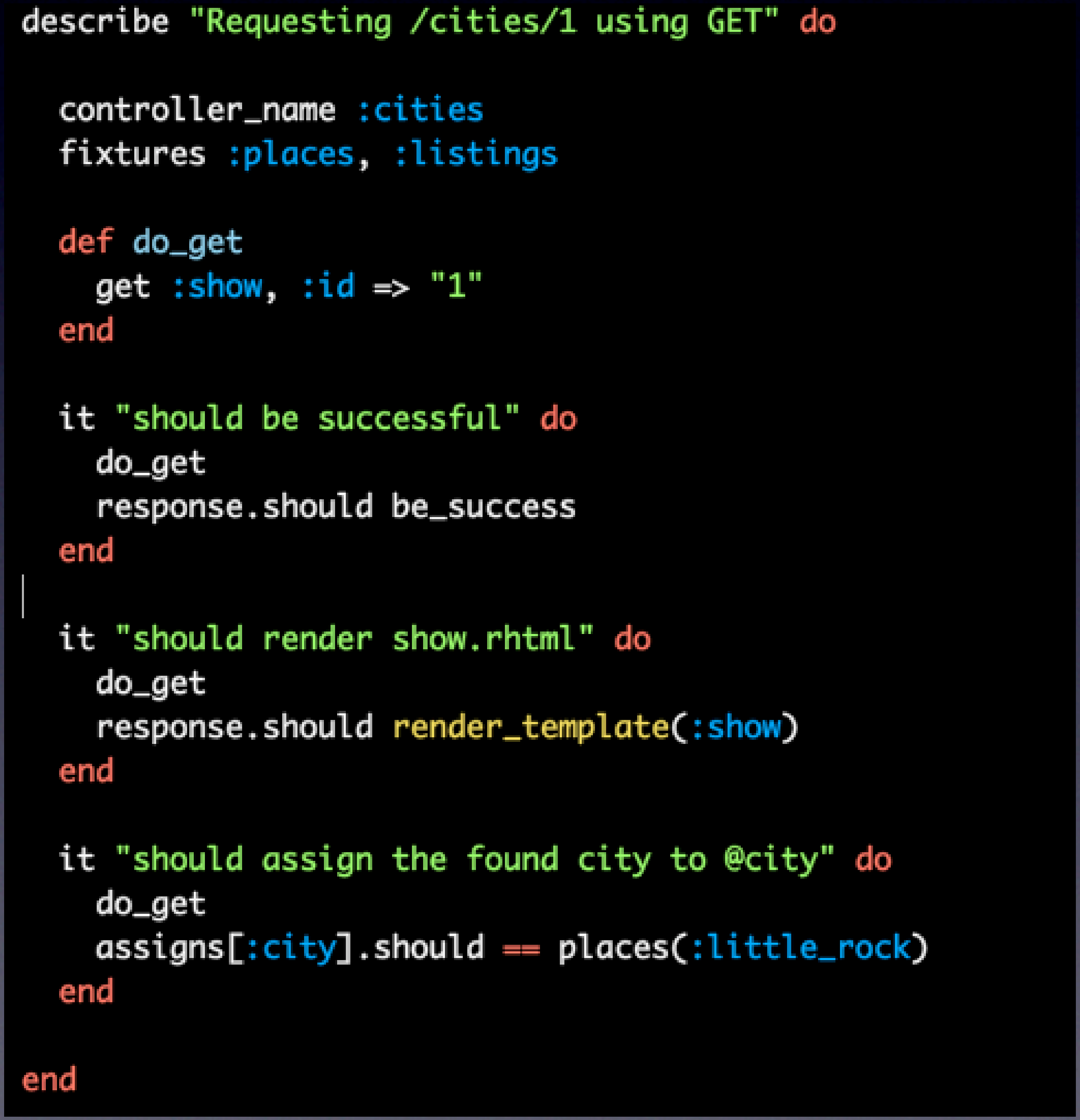
```ruby
require File.dirname(__FILE__) + '/../spec_helper'

describe CreditCard, "Given a charge to a credit card" do
  it "should raise an error if the ActiveMerchant::Billing::CreditCard is invalid" do
    credit_card = stub('credit_card', :valid? => false)
    ActiveMerchant::Billing::CreditCard.should_receive(:new).and_return(credit_card)

    lambda {
      CreditCard.new.charge(10.dollars)
    }.should raise_error("invalid card")
  end

  it "should authorize and capture the authorized amount immediately if authorization succeede
    billing_cc = stub(ActiveMerchant::Billing::CreditCard, :valid? => true)
    ActiveMerchant::Billing::CreditCard.should_receive(:new).and_return(billing_cc)

    gateway = mock(ActiveMerchant::Billing::PayflowGateway)
    response = stub('response', :success? => true, :authorization => "authorization string")
    gateway.should_receive(:authorize).with(10.dollars, billing_cc).and_return(response)
    gateway.should_receive(:capture).with(10.dollars, response.authorization)
    ActiveMerchant::Billing::PayflowGateway.should_receive(:new).and_return(gateway)

    CreditCard.new.charge(10.dollars)
  end
```

mocks and stubs

- RSpec has its own mocking framework
- You can also use Mocha and flexmock
  - Just add "config.mock_with :mocha" in your spec_helper.rb file.

```ruby
require File.dirname(__FILE__) + '/../spec_helper'

describe CreditCard, "Given a charge to a credit card" do
  it "should raise an error if the ActiveMerchant::Billing::CreditCard is invalid" do
    credit_card = stub('credit_card', :valid? => false)
    ActiveMerchant::Billing::CreditCard.should_receive(:new).and_return(credit_card)

    lambda {
      CreditCard.new.charge(10.dollars)
    }.should raise_error("invalid card")
  end

  it "should authorize and capture the authorized amount immediately if authorization succeed
    billing_cc = stub(ActiveMerchant::Billing::CreditCard, :valid? => true)
    ActiveMerchant::Billing::CreditCard.should_receive(:new).and_return(billing_cc)

    gateway = mock(ActiveMerchant::Billing::PayflowGateway)
    response = stub('response', :success? => true, :authorization => "authorization string")
    gateway.should_receive(:authorize).with(10.dollars, billing_cc).and_return(response)
    gateway.should_receive(:capture).with(10.dollars, response.authorization)
    ActiveMerchant::Billing::PayflowGateway.should_receive(:new).and_return(gateway)

    CreditCard.new.charge(10.dollars)
  end

  it "should raise response error message if the authorization fails" do
    billing_cc = stub(ActiveMerchant::Billing::CreditCard, :valid? => true)
    ActiveMerchant::Billing::CreditCard.should_receive(:new).and_return(billing_cc)
```

Example of a Controller Spec



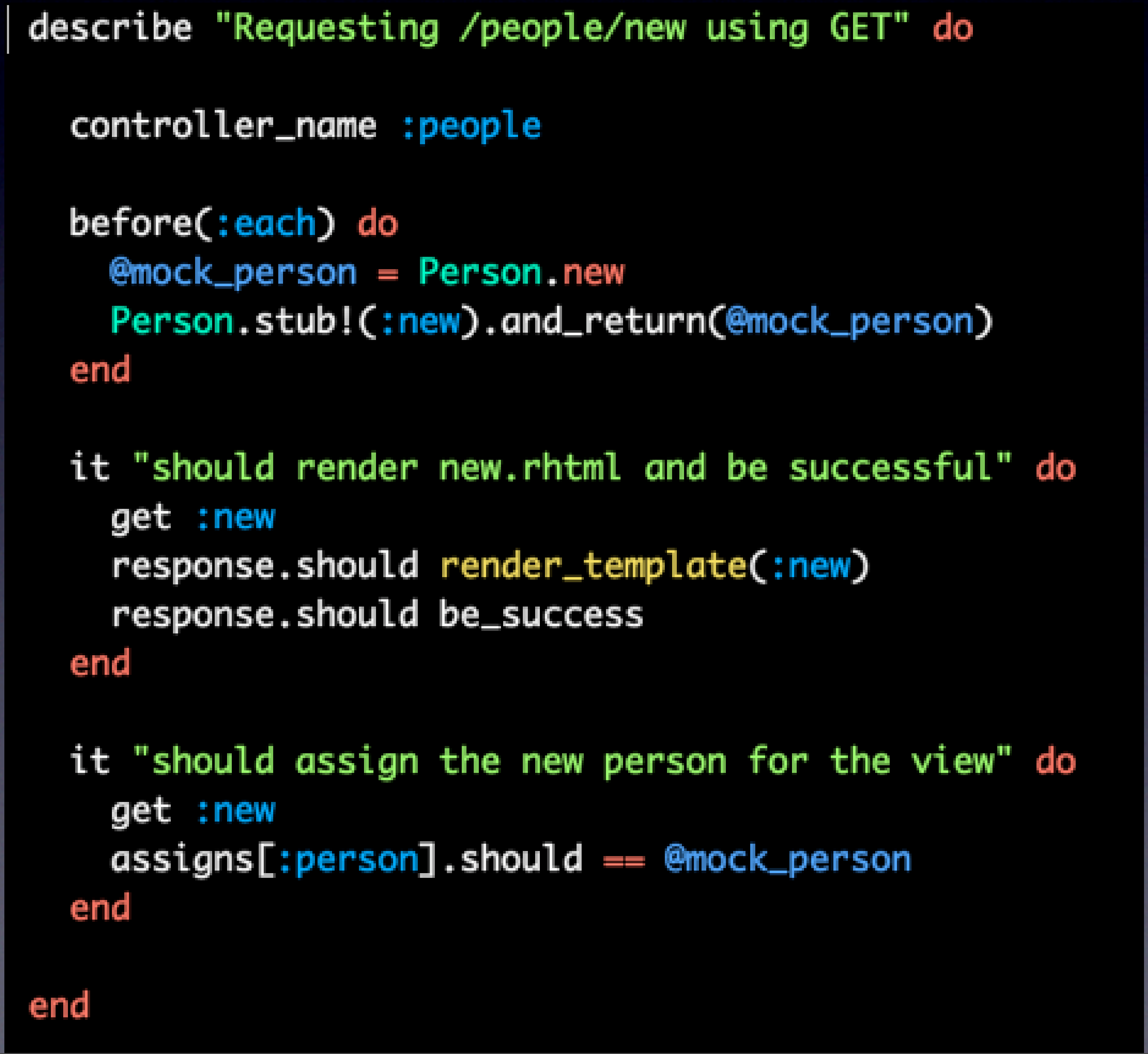
```
describe "Requesting /cities/1 using GET" do

  controller_name :cities
  fixtures :places, :listings

  def do_get
    get :show, :id => "1"
  end

  it "should be successful" do
    do_get
    response.should be_success
  end

  it "should render show.rhtml" do
    do_get
    response.should render_template(:show)
  end

  it "should assign the found city to @city" do
    do_get
    assigns[:city].should == places(:little_rock)
  end

end
```

Controller Specs

- Run by default in isolation mode
  - view templates are not involved
  - benefit is that you can spec the controller without having to worry about your view.
- Can run in integration mode as well
  - Like traditional Rails functional tests you would exercise your views at the same time
  - use the integrate_views macro in your

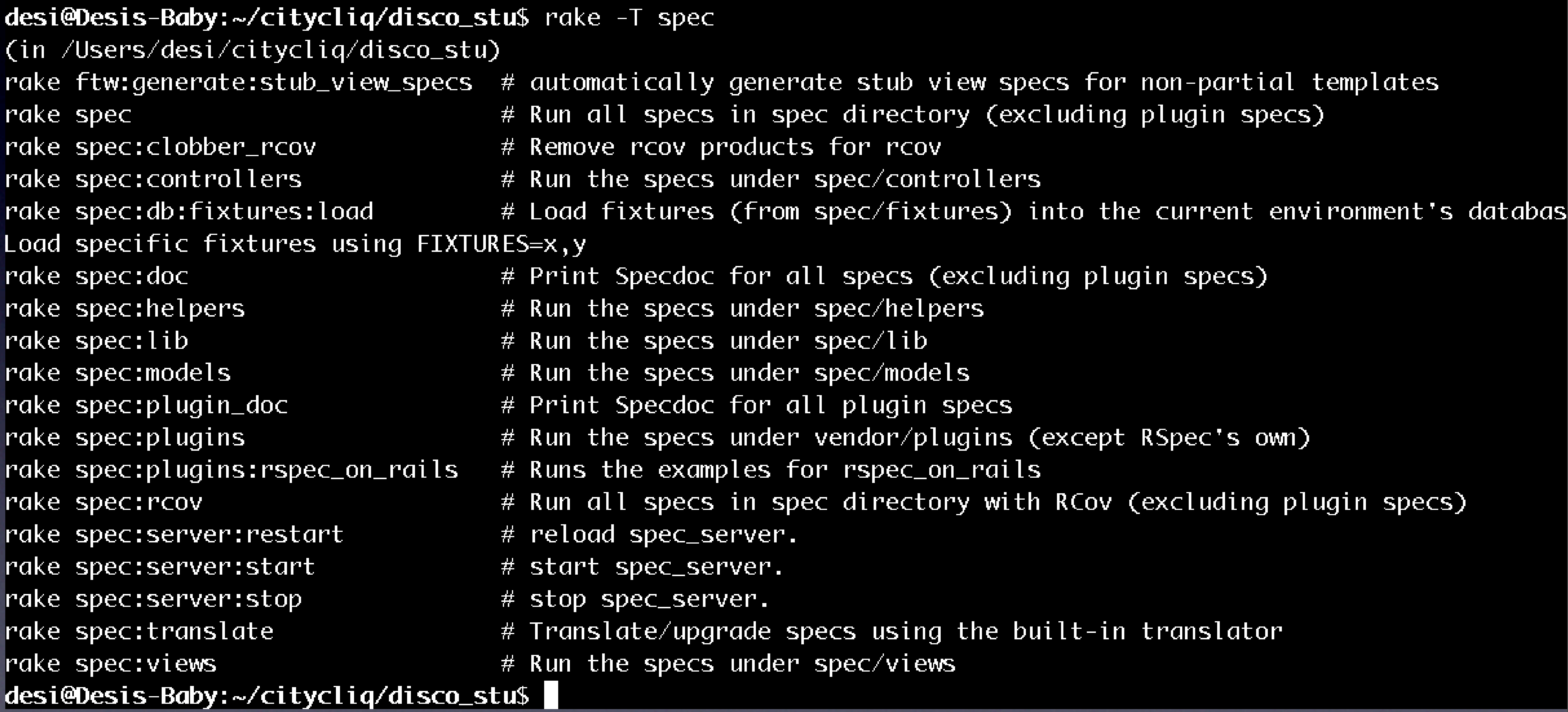
```ruby
describe PeopleController, "GET to /people" do
  integrate_views
  it "should not blow up if no person is logged in" do
    get :index
  end
end
```

Before() the setup of Rspec World



```ruby
describe "Requesting /people/new using GET" do

  controller_name :people

  before(:each) do
    @mock_person = Person.new
    Person.stub!(:new).and_return(@mock_person)
  end

  it "should render new.rhtml and be successful" do
    get :new
    response.should render_template(:new)
    response.should be_success
  end

  it "should assign the new person for the view" do
    get :new
    assigns[:person].should == @mock_person
  end

end
```

Rake Tasks



```
desi@Desis-Baby:~/citycliq/disco_stu$ rake -T spec
(in /Users/desi/citycliq/disco_stu)
rake ftw:generate:stub_view_specs  # automatically generate stub view specs for non-partial templates
rake spec                          # Run all specs in spec directory (excluding plugin specs)
rake spec:clobber_rcov             # Remove rcov products for rcov
rake spec:controllers              # Run the specs under spec/controllers
rake spec:db:fixtures:load         # Load fixtures (from spec/fixtures) into the current environment's databas
Load specific fixtures using FIXTURES=x,y
rake spec:doc                      # Print Specdoc for all specs (excluding plugin specs)
rake spec:helpers                  # Run the specs under spec/helpers
rake spec:lib                      # Run the specs under spec/lib
rake spec:models                   # Run the specs under spec/models
rake spec:plugin_doc               # Print Specdoc for all plugin specs
rake spec:plugins                  # Run the specs under vendor/plugins (except RSpec's own)
rake spec:plugins:rspec_on_rails   # Runs the examples for rspec_on_rails
rake spec:rcov                     # Run all specs in spec directory with RCov (excluding plugin specs)
rake spec:server:restart           # reload spec_server.
rake spec:server:start             # start spec_server.
rake spec:server:stop              # stop spec_server.
rake spec:translate                # Translate/upgrade specs using the built-in translator
rake spec:views                    # Run the specs under spec/views
desi@Desis-Baby:~/citycliq/disco_stu$ 
```

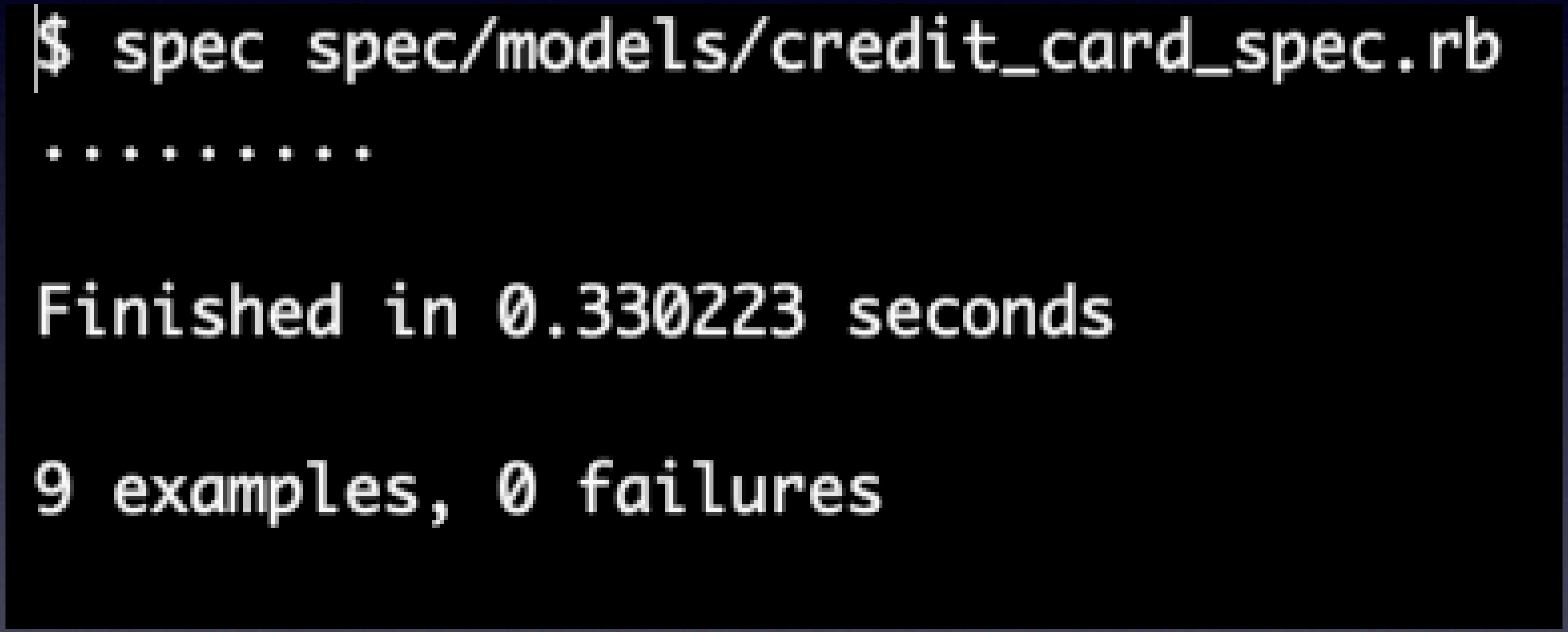
Running Specs

- Rake Tasks - Next slide
- TextMate Integration
  - Bundle for running specs from inside textmate
- Individual spec runs
  - spec <PATH TO SPEC TO RUN>

RSpec Output Formats

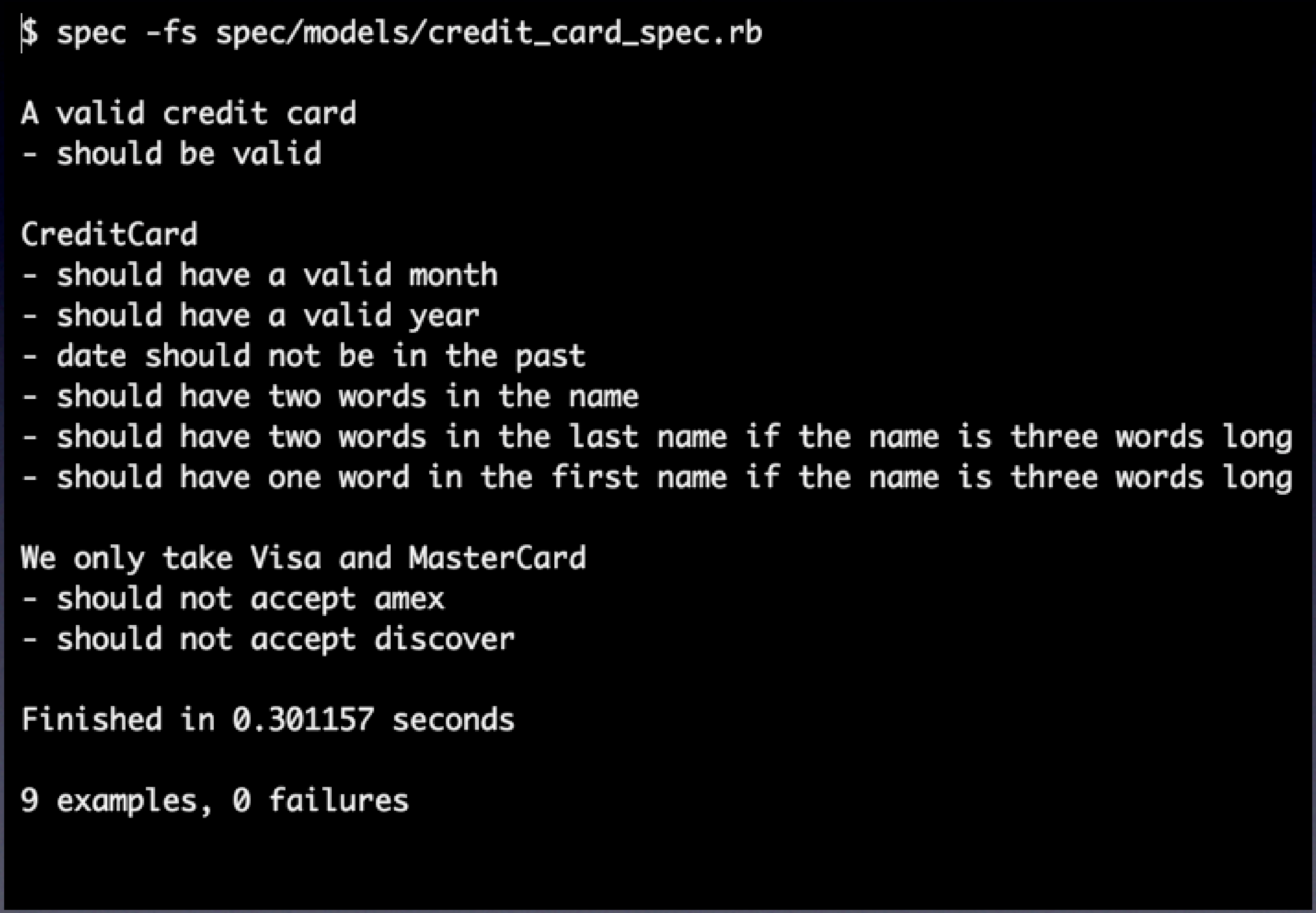
- progress - .... you get the idea.
- specdoc format - use the --format option
- Rdoc-style format - Cool too
- Color coded html output - Pretty Sweet unless you are color blind in which case you could care less.

Progress Format



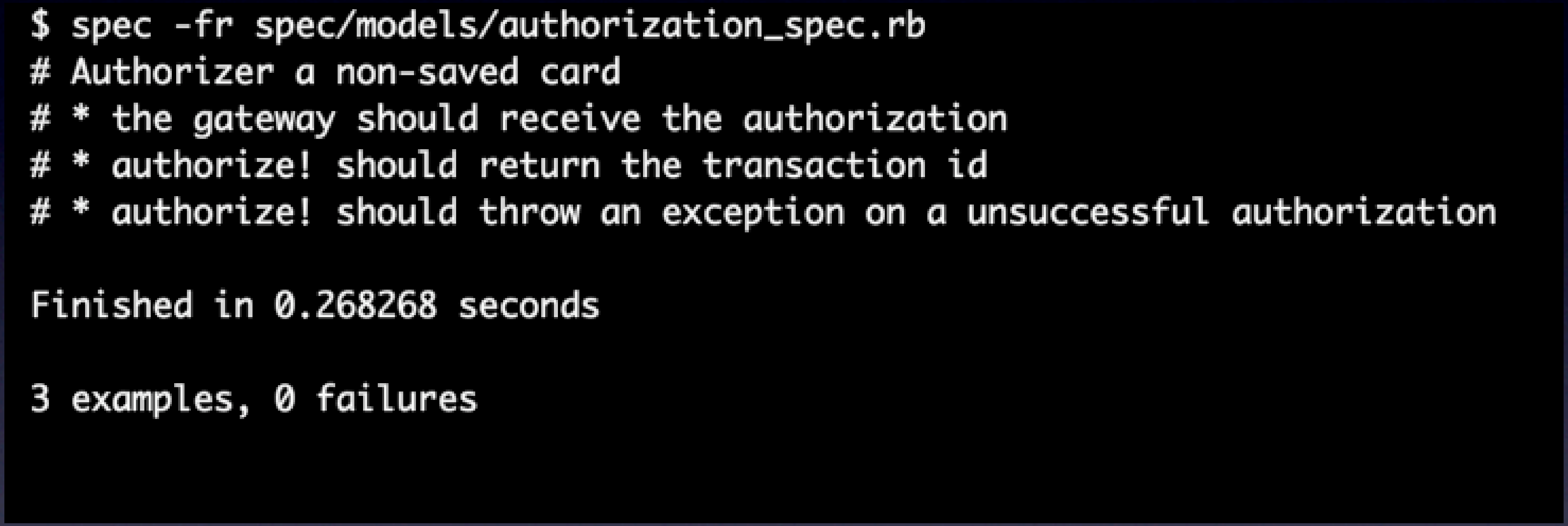
```
$ spec spec/models/credit_card_spec.rb
.........

Finished in 0.330223 seconds

9 examples, 0 failures
```

Spec Doc Format



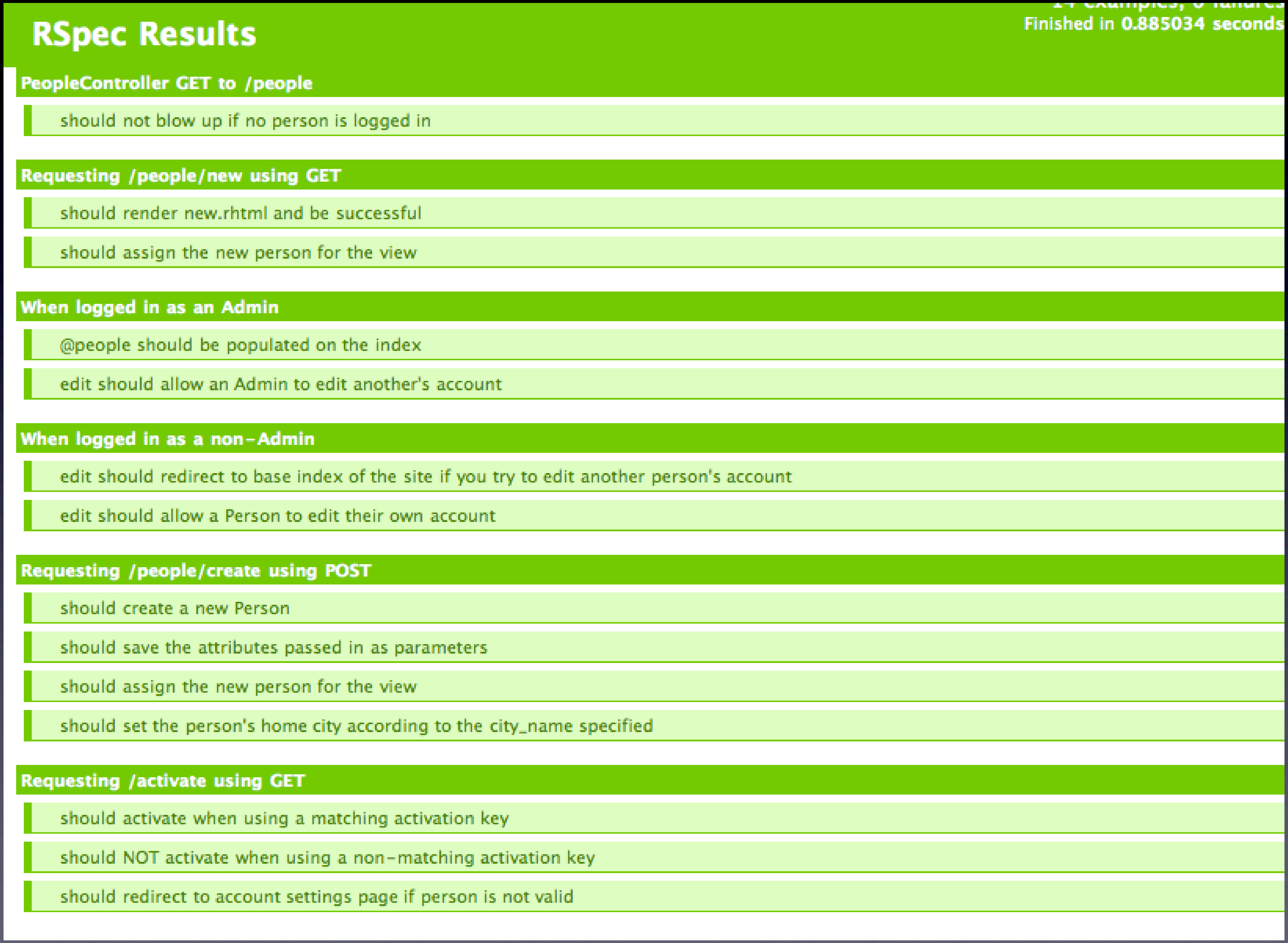
```
$ spec -fs spec/models/credit_card_spec.rb

A valid credit card
- should be valid

CreditCard
- should have a valid month
- should have a valid year
- date should not be in the past
- should have two words in the name
- should have two words in the last name if the name is three words long
- should have one word in the first name if the name is three words long

We only take Visa and MasterCard
- should not accept amex
- should not accept discover

Finished in 0.301157 seconds

9 examples, 0 failures
```

RDoc format



```
$ spec -fr spec/models/authorization_spec.rb
# Authorizer a non-saved card
# * the gateway should receive the authorization
# * authorize! should return the transaction id
# * authorize! should throw an exception on a unsuccessful authorization

Finished in 0.268268 seconds

3 examples, 0 failures
```

Color coded TextMate

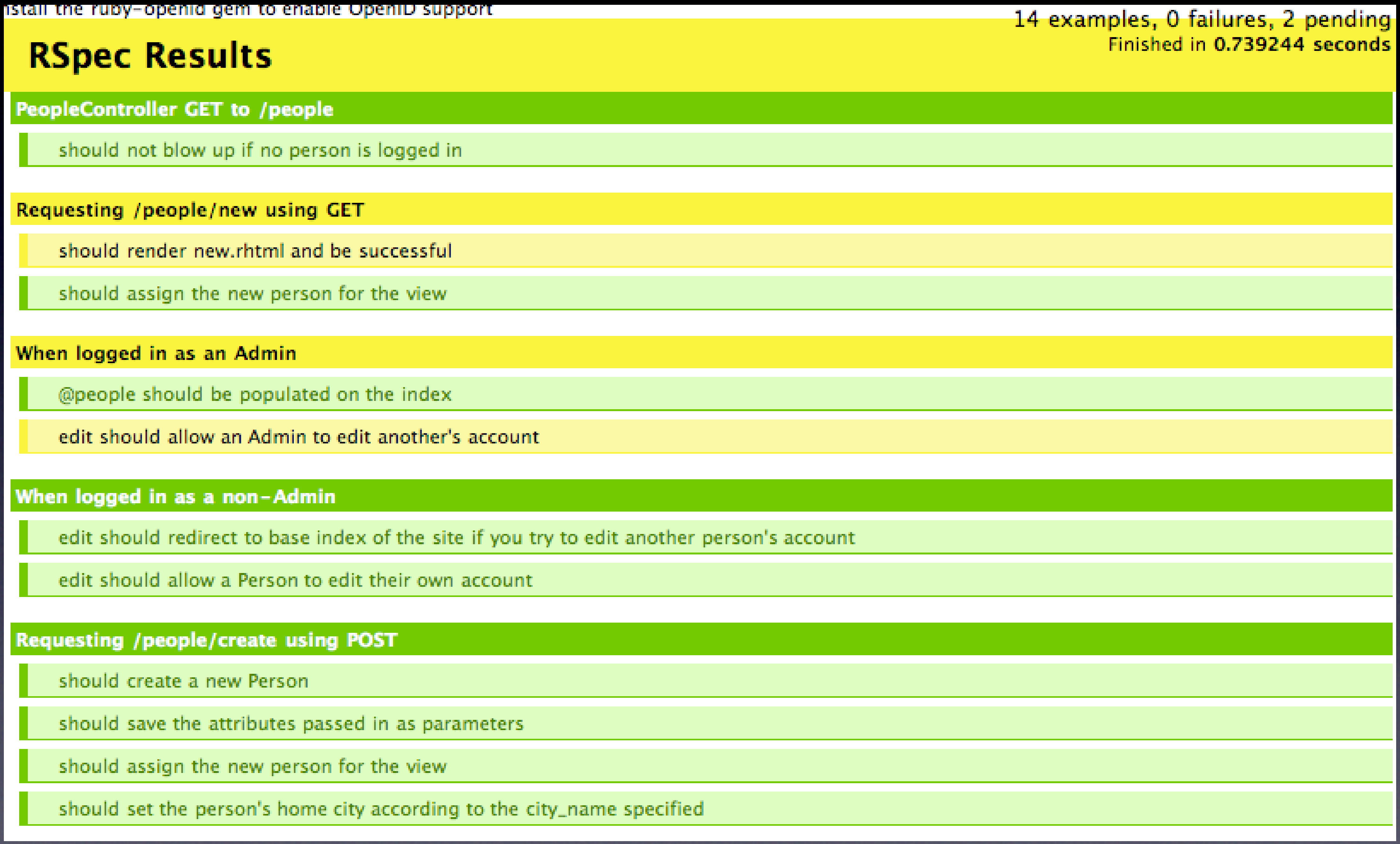


```
RSpec Results                                              14 examples, 0 failures
                                                  Finished in 0.885034 seconds
PeopleController GET to /people
    should not blow up if no person is logged in

Requesting /people/new using GET
    should render new.rhtml and be successful
    should assign the new person for the view

When logged in as an Admin
    @people should be populated on the index
    edit should allow an Admin to edit another's account

When logged in as a non-Admin
    edit should redirect to base index of the site if you try to edit another person's account
    edit should allow a Person to edit their own account

Requesting /people/create using POST
    should create a new Person
    should save the attributes passed in as parameters
    should assign the new person for the view
    should set the person's home city according to the city_name specified

Requesting /activate using GET
    should activate when using a matching activation key
    should NOT activate when using a non-matching activation key
    should redirect to account settings page if person is not valid
```

Pending

- Lets you define the examples without implementing them.
  - Lets you let business people get involved
  - You can give yourself to-do’s

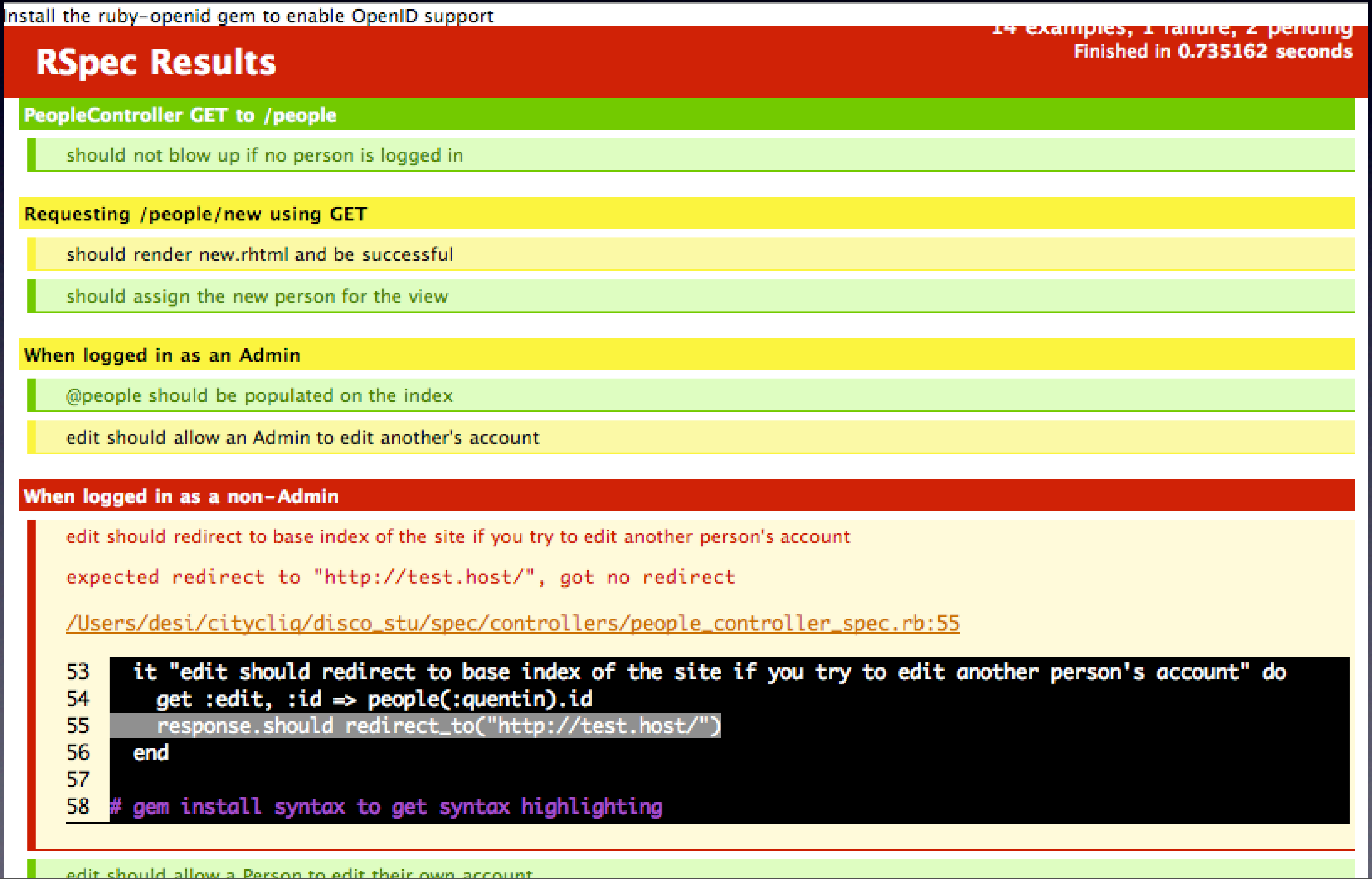
Prreeettyy



nstall the ruby-openid gem to enable OpenID support

```
RSpec Results                                    14 examples, 0 failures, 2 pending
                                                   Finished in 0.739244 seconds

PeopleController GET to /people
    should not blow up if no person is logged in

Requesting /people/new using GET
    should render new.rhtml and be successful
    should assign the new person for the view

When logged in as an Admin
    @people should be populated on the index
    edit should allow an Admin to edit another's account

When logged in as a non-Admin
    edit should redirect to base index of the site if you try to edit another person's account
    edit should allow a Person to edit their own account

Requesting /people/create using POST
    should create a new Person
    should save the attributes passed in as parameters
    should assign the new person for the view
    should set the person's home city according to the city_name specified
```

For Good Measure Lets see some Red



Install the ruby-openid gem to enable OpenID support

RSpec Results

14 examples, 1 failure, 2 pending
Finished in 0.735162 seconds

PeopleController GET to /people

should not blow up if no person is logged in

Requesting /people/new using GET

should render new.rhtml and be successful

should assign the new person for the view

When logged in as an Admin

@people should be populated on the index

edit should allow an Admin to edit another's account

When logged in as a non-Admin

edit should redirect to base index of the site if you try to edit another person's account

```
expected redirect to "http://test.host/", got no redirect
```

/Users/desi/citycliq/disco_stu/spec/controllers/people_controller_spec.rb:55

```
53  it "edit should redirect to base index of the site if you try to edit another person's account" do
54    get :edit, :id => people(:quentin).id
55    response.should redirect_to("http://test.host/")
56  end
57
58 # gem install syntax to get syntax highlighting
```

edit should allow a Person to edit their own account

Resources

- David Chemlisky's - blog.davidchelimsky.net
- The Rails Way by Obie Fernandez
- PeepCode ScreenCasts
- RSpec documentation - rspec.rubyforge.org
- Lots and lots of other blogs

Contact Info

- Desi McAdam - desi.mcadam@gmail.com
- Blog - www.devchix.com