

Controllers

```
require 'rails_helper'

describe ApplicationController do
  before(:each) do
    @articles = Article.create([
      {
        title: Faker::Lorem.sentence(
          rand(3)+2),
        body: Faker::Lorem.paragraph(
          rand(30)+30),
      }
    ])
  end

  describe 'GET search' do
    it 'renders search' do
      get :search
      expect(response.status).to eq(200)
      expect(response).to render_template(:index)
    end

    it 'assigns instance variables' do
      get :search, :params => {:query => @articles.first.title}

      expect(response.status).to eq(200)
      expect(assigns(:query)).to eq(@articles.first.title)
      expect(assigns(:show_search_results)).to be(true)
    end

    it 'returns search results' do
      search_result = Article.create({
        title: "Find Me",
        body: "One Result ",
        published: true,
        author_id: @author.id
      })
      get :search, :params => {:query => 'One Result'}
      expect(response.status).to eq(200)
```

Controllers (cont)

```
> expect(assigns(:articles).length).to eq(1)
expect(assigns(:articles).first.title).to eq("Find Me")
#expect(response).to redirect_to(location)
end
end
end
```

Models

```
RSpec.describe MyModel, type: :model do

  let(:string) { "foo bar bazz" }
  it 'checks something' do

    expect(my_model).to be_valid
    expect(actual).to eq(expected)
    expect(actual).to be_nil
    expect(actual).to be true
    expect(actual).to be > expected
    expect(actual).to be >= expected
    expect(actual).to be <= expected
    expect(actual).to be < expected
    expect(actual).to be_between(min,
max).inclusive
    expect(actual).to be_between(min,
max).exclusive
    expect(actual).to match(/expression/)
    expect(actual).to be_within(delta).of(expected)
    expect(actual).to start_with expected
    expect(actual).to end_with expected
    expect(actual).to start_with("foo").and end_with("bazz")
    expect(result).not_to be_empty
    expect(actual).to be_instance_of(expected)
    expect(actual).to be_kind_of(expected)
    expect(actual).to respond_to(expected)
    expect(arr).to be_an(Array)
    expect{...}.to raise_error
```

Models (cont)

```
> expect { ... }.to raise_error(ErrorClass)
expect { ... }.to raise_error("message")
expect { ... }.to raise_error(ErrorClass, "message")
expect { ... }.to throw_symbol
expect { ... }.to throw_symbol(:symbol)
expect { ... }.to throw_symbol(:symbol, 'value')

end

end
```

Using `not_to` instead of `to` reverses the condition.

Factory Girl

```
#Defining factories
FactoryGirl.define do
  factory :user do
    first_name 'John'
    last_name 'Doe'
    birthdate { 21.years.ago }
    admin false
    sequence(:username) { |n| "user#{n}" }
  end
end

# Using
FactoryGirl.build(:user, name: 'John') # not saved
create(:user) # saved
attribute_for(:user) # hash
```

