**RSPEC**

➜ Add the following gems to your gemfile

group :development, :test do

gem 'rspec-rails'

~~gem ‘rspec-rails-mocha’ Mocha conflicts with email-spec gem~~

gem 'faker'

gem 'populator'

gem 'annotate'

gem 'hirb'

gem 'shoulda-matchers' # shoulda-context causes the error http://stackoverflow.com/q/9523931/238880

gem 'factory\_girl\_rails' # Used for seeding data also

gem 'rb-fsevent', require: false

gem 'guard-rspec'

gem 'guard-spork'

gem ‘guard-annotate’

gem ‘guard-bundler’

end

group :test do

gem 'email\_spec' # Mocha conflicts with email\_spec gem

gem 'cucumber-rails', require: false

gem 'capybara'

gem 'database\_cleaner'

gem 'launchy'

gem 'spork-rails'

end

➜ Run rspec install command

$ rails generate rspec:install

create .rspec

exist spec

create spec/spec\_helper.rb

➜ Delete Test Unit directory

$ rm -rf test

➜ Skip unit tests for helpers and views. Modify the file **config/application.rb** to include:

# don't generate RSpec tests for views and helpers  
config.generators do |g|  
 g.view\_specs false  
 g.helper\_specs false  
end

**CUCUMBER**

➜ Run cucumber generator

$ rails generate cucumber:install --capybara --rspec

The -–capybara option specifies Capybara instead of the default Webrat for acceptance testing. The -–rspec option enables RSpec matchers for your step definitions.

The cucumber-rails generator creates the directories:

* **features/step\_definitions**
* **features/support**

The cucumber-rails generator creates the files:

* **config/cucumber.yml**
* **features/support/env.rb**
* **lib/tasks/cucumber.rake**
* **script/cucumber**

➜ To save some typing, you can eliminate the need for --require features by changing the **config/cucumber.yml** file:

std\_opts = "-r features/support/ -r features/step\_definitions --format #{ENV['CUCUMBER\_FORMAT'] || 'pretty'} --strict --tags ~@wip"

#### ➜ **Database Cleaner for Cucumber**

To reset your application database to a pristine state during testing, Cucumber makes use of [Database Cleaner](http://github.com/bmabey/database_cleaner). The file **features/support/env.rb** is already set up to use Database Cleaner:

begin  
 DatabaseCleaner.strategy = :transaction  
rescue NameError  
 raise "You need to add database\_cleaner to your Gemfile (in the :test group) if you wish to use it."  
end

#### ➜ **Email Spec**

**RSpec**

Modify the file **spec/spec\_helper.rb** to enable the email\_spec library:

# This file is copied to spec/ when you run 'rails generate rspec:install'  
ENV["RAILS\_ENV"] ||= 'test'  
require File.expand\_path("../../config/environment", \_\_FILE\_\_)  
require 'rspec/rails'  
require 'email\_spec'  
require 'rspec/autorun'  
require 'rspec/rails'

~~require 'rspec/rails/~~**~~mocha~~**~~'~~

.  
.  
.  
RSpec.configure do |config|  
 config.include(EmailSpec::Helpers)  
 config.include(EmailSpec::Matchers)

### Mock Framework

config.mock\_with ~~:mocha~~ :rspec  
.  
.  
end

→ **Factory Girl Helpers**

Add the following to spec\_helper.rb

config.include FactoryGirl::Syntax::Methods

**Cucumber**

The email\_spec gem provides a collection of Cucumber steps for testing email.

Add a file **features/support/email\_spec.rb** to enable the email\_spec library:

require 'email\_spec/cucumber'

Create a set of Cucumber step definitions for testing email:

$ rails generate email\_spec:steps

This creates a file **features/step\_definitions/email\_steps.rb**.

#### ➜ **Guard**

$ bundle exec guard init

19:56:17 - INFO - Writing new Guardfile to /Users/pkhadloya/Personal/railsforcharity/Guardfile

19:56:17 - INFO - rspec guard added to Guardfile, feel free to edit it

$ bundle exec guard init rspec

$ bundle exec guard init annotate

$ bundle exec guard init bundler

$ bundle exec guard start

#### ➜ **Spork**

$ spork rspec --bootstrap

$ bundle exec guard init spork

\* Copy the spork config in Guardfile to the top

# More info at <https://github.com/guard/guard#readme>

**DEVISE TESTING**

➜ **Add Devise Test Helpers**

Using Devise, your controllers will often include before\_filter :authenticate\_user! to limit access to signed-in users. Your tests will fail unless a default user is created and logs in before each test runs. Devise provides test helpers to make it simple to create and log in a default user.

**Create a file called controller\_macros.rb in spec/support and add the following in it**

module ControllerMacros

def login\_admin

before(:each) do

@request.env["devise.mapping"] = Devise.mappings[:admin]

sign\_in FactoryGirl.create(:admin) # Using factory girl as an example

end

end

def login\_user

before(:each) do

@request.env["devise.mapping"] = Devise.mappings[:user]

user = FactoryGirl.create(:user)

user.confirm! # or set a confirmed\_at inside the factory. Only necessary if you are using the confirmable module

sign\_in user

end

end

end

**In spec/spec\_helper.rb add the following**

config.include Devise::TestHelpers, :type => :controller

config.extend ControllerMacros, :type => :controller

And now you can use the method **login\_user** if you want to get a user to login before running your test.

#### ➜ **Accommodate Cucumber Testing for “Sign Out”**

By default, Devise uses an http DELETE request for sign out requests (destroy\_user\_session\_path). Rails uses Javascript to implement http DELETE requests. Prior to Devise 1.4.1 (27 June 2011), Devise used an http GET request for sign out requests. Jose Valim explained the change: “GET requests should not change the state of the server. When sign out is a GET request, CSRF can be used to sign you out automatically and things that preload links can eventually sign you out by mistake as well.”

However, Cucumber wants to test GET requests not DELETE requests. If you intend to use Cucumber with Devise, you must change the Devise default from DELETE to GET in **/config/initializers/devise.rb** for the Rails test environment. You may see a suggestion elsewhere to tweak the routes.rb file or change the log\_out link to make the fix. It isn’t necessary if you change the **/config/initializers/devise.rb** file.

# The default HTTP method used to sign out a resource. Default is :delete.  
config.sign\_out\_via = Rails.env.test? ? :get : :delete

Since you only use Cucumber during testing, switching the default is only needed for testing.

If you’re not going to use Cucumber, leave Devise’s default (DELETE) in place.

Sample Guard File:

# More info at https://github.com/guard/guard#readme

guard 'spork', :cucumber\_env => { 'RAILS\_ENV' => 'test' }, :rspec\_env => { 'RAILS\_ENV' => 'test' } do

watch('config/application.rb')

watch('config/environment.rb')

watch('config/environments/test.rb')

watch(%r{^config/initializers/.+\.rb$})

watch('Gemfile')

watch('Gemfile.lock')

watch('spec/spec\_helper.rb') { :rspec }

# watch(%r{features/support/}) { :cucumber }

end

guard 'rspec', :cli => "--color --format nested --fail-fast --drb", :all\_on\_start => false, :all\_after\_pass => false do

watch(%r{^spec/.+\_spec\.rb$})

watch(%r{^lib/(.+)\.rb$}) { |m| "spec/lib/#{m[1]}\_spec.rb" }

watch('spec/spec\_helper.rb') { "spec" }

# Rails example

watch(%r{^app/(.+)\.rb$}) { |m| "spec/#{m[1]}\_spec.rb" }

watch(%r{^app/(.\*)(\.erb|\.haml)$}) { |m| "spec/#{m[1]}#{m[2]}\_spec.rb" }

watch(%r{^app/controllers/(.+)\_(controller)\.rb$}) { |m| ["spec/routing/#{m[1]}\_routing\_spec.rb", "spec/#{m[2]}s/#{m[1]}\_#{m[2]}\_spec.rb", "spec/acceptance/#{m[1]}\_spec.rb"] }

watch(%r{^spec/support/(.+)\.rb$}) { "spec" }

watch('config/routes.rb') { "spec/routing" }

watch('app/controllers/application\_controller.rb') { "spec/controllers" }

# Capybara features specs

# watch(%r{^app/views/(.+)/.\*\.(erb|haml)$}) { |m| "spec/features/#{m[1]}\_spec.rb" }

# Turnip features and steps

# watch(%r{^spec/acceptance/(.+)\.feature$})

# watch(%r{^spec/acceptance/steps/(.+)\_steps\.rb$}) { |m| Dir[File.join("\*\*/#{m[1]}.feature")][0] || 'spec/acceptance' }

end