Sinatra Best Practices: Part Two

In part one we saw how to break up your Sinatra application into smaller bite-size pieces. In part two we're going to explore testing, which will also compel us to address environment configuration.

To get started, we need to add rspec and rack-test to our gem file:

```
# Gemfile

source 'https://rubygems.org'

gem 'sinatra'

group :test, :development do

  gem 'rspec'

end

group :test do

  gem 'rack-test'

end
```

Now we can leverage different environments to load only the gems we need. We don't really want RSpec loaded in production, after all, do we?

```
# app.rb

ENV['RACK_ENV'] ||= 'development'

require 'bundler'

Bundler.require :default, ENV['RACK_ENV'].to_sym

require_relative 'helpers'

require_relative 'routes/secrets'

require_relative 'routes/sessions'

class SimpleApp < Sinatra::Base
```

```
  # No changes here - left out for brevity.

end
```

The first line sets the environment to development by default. After that, we use Bundler to load only the gems for the appropriate environment.

We're pretty big believers in having the default rake task run your full test suite, so to that end, our rakefile looks something like this:

```
# rakefile

require 'rspec/core/rake_task'

RSpec::Core::RakeTask.new :specs do |task|

  task.pattern = Dir['spec/**/*_spec.rb']

end

task :default => ['specs']
```

Now, a simple 'bundle exec rake' will run our full test suite, which will yield 0 tests run at this point, so let's fix that by adding some. We'll start with the spec_helper, which is not a requirement, per se, but makes things quite clean when we get to constructing tests, as we'll see in just a moment.

```
# spec/spec_helper.rb

ENV['RACK_ENV'] = 'test'

require_relative File.join('..', 'app')

RSpec.configure do |config|

  include Rack::Test::Methods

  def app

    SimpleApp

  end

end
```

In our spec helper, we force the environment to be test and then load the app, which will load the appropriate gems via bundler.

So let's add a spec:

```
# spec/features/root_spec.rb

require_relative '../spec_helper'

describe 'Root Path' do

  describe 'GET /' do

    before { get '/' }

    it 'is successful' do

      expect(last_response.status).to eq 200

    end

  end

end
```

And there we have it. Our Sinatra app is loading only the gems necessary for the environment and our testing infrastructure is in place.