Sinatra Recipes

Community contributed recipes and techniques

Fork me on GitHub

Select a topic

Testing ¶ ↑

Testing is an important part of building and designing applications. Sinatra is no exception, and there are more than a few ways to test your application.

This section will cover most of these, along with the book and readme.

- Capybara With Steak
- Minitest
- Rspec
- Test-unit
- Webrat With Cucumber

Did we miss something?

It's very possible we've left something out, that's why we need your help!This is a community driven project after all. Feel free to fork the project and send us a pull request to get your recipe or tutorial included in the book.

See the README for more details.

Top

Sinatra Recipes

Community contributed recipes and techniques

Select a topic

Chapters

Using Capybara ¶ ↑

Steak ¶ ↑

```
# spec/acceptance/acceptance_helper.rb

ENV['RACK_ENV'] = 'test'

require 'rubygems'
require 'steak'
require 'rack/test'
require 'capybara/dsl'

RSpec.configure do |config|
  config.include Capybara
end

require File.expand_path '../../../my-app.rb', __FILE__

Capybara.app = Sinatra::Application
```

My Page Acceptance Spec

```
require File.expand_path '../acceptance_helper.rb', __FILE__

feature "My Page" do
```

```
  scenario "greets the visitor" do
    visit "/"
    page.should have_content "Welcome to my page!"
  end

end
```

Steak Resources

- Source on GitHub
- Documentation
- Timeless: BDD with RSpec and Steak
- More Steak Resources

The Future of Steak

Capybara's acceptance testing DSL was recently added. They just basically consumed Steak, since it was always "a gist that happened to be distributed as a Ruby gem." It won't be included until the next release of Capybara, but after that, Steak won't be needed, unless you're using an older version of Capybara for some reason.

Did we miss something?

It's very possible we've left something out, that's why we need your help!This is a community driven project after all. Feel free to fork the project and send us a pull request to get your recipe or tutorial included in the book.

See the README for more details.

Top

Sinatra Recipes

Community contributed recipes and techniques

Select a topic

Chapters

Minitest ¶ ↑

Since Ruby 1.9, Minitest is shipped with the standard library. If you want to use it on 1.8, it is still installable via Rubygems.

After installing Minitest, setting it up works similar to `Test::Unit`:

If you have multiple test files, you could create a test helper file and do all the setup in there:

```
# test_helper.rb
ENV['RACK_ENV'] = 'test'
require 'minitest/autorun'
require 'rack/test'

require File.expand_path '../my-app.rb', __FILE__
```

In your test files you only have to require that helper:

```
# test.rb
require File.expand_path '../test_helper.rb', __FILE__

class MyTest < MiniTest::Unit::TestCase

  include Rack::Test::Methods

  def app
    Sinatra::Application
  end
```

```
  def test_hello_world
    get '/'
    assert last_response.ok?
    assert_equal "Hello, World!", last_response.body
  end
end
```

If your app was defined using the modular style, use

```
def app
  MyApp # <- your application class name
end
```

instead of

```
def app
  Sinatra::Application
end
```

Specs and Benchmarks with Minitest

Specs

```
require File.expand_path '../test_helper.rb', __FILE__

include Rack::Test::Methods

def app
  Sinatra::Application
end

describe "my example app" do
  it "should successfully return a greeting" do
    get '/'
    assert_equal 'Welcome to my page!', last_response.body
  end
end
```

Benchmarks

```
require File.expand_path '../test_helper.rb', __FILE__

require 'minitest/benchmark'

include Rack::Test::Methods

def app
  Sinatra::Application
end
```

```
describe "my example app" do
  bench_range { bench_exp 1, 10_000 }
  bench_performance_linear "welcome message", 0.9999 do |n|
    n.times do
      get '/'
      assert_equal 'Welcome to my page!', last_response.body
    end
  end
end
```

Rake

When you're ready to start using MiniTest as an automated testing framework, you'll need to setup a Rake TestTask. Here's one we'll use to run our MiniTest::Specs:

```
require 'rake/testtask'
Rake::TestTask.new do |t|
  t.pattern = "spec/*_spec.rb"
end
```

Now run your MiniSpecs with `rake test`.

More on Rake::TestTask

MiniTest Resources

- Source on github
- Documentation
- Official Blog Archive
- 1.9.2 Stdlib Documentation
- Bootspring MiniTest Blog Post

Did we miss something?

It's very possible we've left something out, that's why we need your help!This is a community driven project after all. Feel free to fork the project and send us a pull request to get your recipe or tutorial included in the book.

See the README for more details.

Top

Sinatra Recipes

Community contributed recipes and techniques

Select a topic

Chapters

RSpec ¶ ↑

[RSpec] is the main competitor to Test::Unit. It is feature rich and pleasant to read, but to heavy for some. Therefore most other frameworks mentioned here (except Minitest, Test::Unit and Cucumber) try to adopt its API without its inner complexity.

In your spec file or your spec helper, you can setup `Rack::Test` like this:

```
# spec/spec_helper.rb
require 'rack/test'

require File.expand_path '../../my-app.rb', __FILE__

module RSpecMixin
  include Rack::Test::Methods
  def app() Sinatra::Application end
end

# For RSpec 2.x
RSpec.configure { |c| c.include RSpecMixin }
# If you use RSpec 1.x you should use this instead:
Spec::Runner.configure { |c| c.include RSpecMixin }
```

If your app was defined using the modular style, use

```
def app() described_class end
```

instead of

```
def app() Sinatra::Application end
```

Now use it in your specs

```
# spec/app_spec.rb
require File.expand_path '../spec_helper.rb', __FILE__

describe "My Sinatra Application" do
  it "should allow accessing the home page" do
    get '/'
    last_response.should be_ok
  end
end
```

RSpec 2.x Resources

- RSpec 2.x Docs
- Source on GitHub
- Resources for RSpec 2.x developers/contributors

RSpec 1.x Resources

- RSpec 1.x Docs
- Source on GitHub
- Resources for RSpec 1.x developers/contributors

Did we miss something?

It's very possible we've left something out, that's why we need your help!This is a community driven project after all. Feel free to fork the project and send us a pull request to get your recipe or tutorial included in the book.

See the README for more details.

Top

Sinatra Recipes

Community contributed recipes and techniques

Select a topic

Chapters

Test::Unit ¶ ↑

One of the advantages of using Test::Unit is that it already ships with Ruby **1.8.7** and you can skip the installation part in some cases.

For modern versions of Ruby, you must install the Test::Unit gem:

```
gem install test-unit
```

Set up rack-test by including `Rack::Test::Methods` into your test class and defining `app`:

```
ENV['RACK_ENV'] = 'test'
require 'test/unit'
require 'rack/test'


require File.expand_path '../my-app.rb', __FILE__

class HomepageTest < Test::Unit::TestCase
  include Rack::Test::Methods
  def app() Sinatra::Application end

  def test_homepage
    get '/'
    assert last_response.ok?
  end
end
```

Shoulda

```
require File.expand_path '../test_helper.rb', __FILE__

class ExampleUnitTest < Test::Unit::TestCase

  include Rack::Test::Methods

  def app() Sinatra::Application end

  context "view my page" do
    setup do
      get '/'
    end

    should "greet the visitor" do
      assert last_response.ok?
      assert_equal 'Welcome to my page!', last_response.body
    end

  end
end
```

Did we miss something?

It's very possible we've left something out, that's why we need your help!This is a community driven project after all. Feel free to fork the project and send us a pull request to get your recipe or tutorial included in the book.

See the README for more details.

Top

Sinatra Recipes

Community contributed recipes and techniques

Select a topic

Chapters

Features from Cucumber and Webrat

A Feature Example

```
Feature: View my page
  In order for visitors to feel welcome
  We must go out of our way
  With a kind greeting

  Scenario: My page
    Given I am viewing my page
    Then I should see "Welcome to my page!"
```

Step to it

```
Given /^I am viewing my page$/ do
  visit('/')
end

Then /^I should see "([^"]*)"$/ do |text|
  last_response.body.should match(/#{text}/m)
end
```

```
# features/support/env.rb

ENV['RACK_ENV'] = 'test'

require 'rubygems'
require 'rack/test'
```

```
require 'rspec/expectations'
require 'webrat'

require File.expand_path '../../../my-app.rb', __FILE__

Webrat.configure do |config|
  config.mode = :rack
end

class WebratMixinExample
  include Rack::Test::Methods
  include Webrat::Methods
  include Webrat::Matchers

  Webrat::Methods.delegate_to_session :response_code, :response_body

  def app
    Sinatra::Application
  end
end

World{WebratMixinExample.new}
```

Cucumber Resources

- Cucumber Homepage
- Source on GitHub
- Documentation

Did we miss something?

It's very possible we've left something out, that's why we need your help!This is a community driven project after all. Feel free to fork the project and send us a pull request to get your recipe or tutorial included in the book.

See the README for more details.

Top