Sinatra Recipes

Community contributed recipes and techniques

Fork me on GitHub

Select a topic

Views ¶ ↑

Sinatra is built upon a very powerful templating engine, Tilt.

This enables you to do all sorts of fun and exciting things to display your views. In this section we will cover some of them, along with the book.

Also, be sure to read up the readme first to cover the basics of what's possible within Sinatra itself and for a list of supported templates inherited from Tilt.

- Coffee Script
- Markdown
- Rss Feed With Builder

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

CoffeeScript ¶ ↑

To render CoffeeScript templates you first need the `coffee-script` gem and `therubyracer`, or access to the `coffee` binary.

Here's an example of using CoffeeScript with Sinatra's template rendering engine Tilt:

```
## You'll need to require coffee-script in your app
require 'coffee-script'

get '/application.js' do
  coffee :application
end
```

Renders `./views/application.coffee`.

This works great if you have access to nodejs or `therubyracer` gem on your platform of choice and hosting environment. If that's not the case, but you'd still like to use CoffeeScript, you can precompile your scripts using the `coffee` binary:

```
coffee -c -o public/javascripts/ src/
```

Or you can use this example rake task to compile them for you with the `coffee-script` gem, which can use either `therubyracer` gem or the `coffee` binary:

```
require 'coffee-script'

namespace :js do
  desc "compile coffee-scripts from ./src to ./public/javascripts"
  task :compile do
```

```
    source = "#{File.dirname(__FILE__)}/src/"
    javascripts = "#{File.dirname(__FILE__)}/public/javascripts/"

    Dir.foreach(source) do |cf|
      unless cf == '.' || cf == '..'
        js = CoffeeScript.compile File.read("#{source}#{cf}")
        open "#{javascripts}#{cf.gsub('.coffee', '.js')}", 'w' do |f|
          f.puts js
        end
      end
    end
  end
end
```

Now, with this rake task you can compile your coffee-scripts to `public/javascripts` by using the `rake js:compile` command.

Resources

If you get stuck or want to look into other ways of implementing CoffeeScript in your application, these are a great place to start:

- coffee-script
- therubyracer
- ruby-coffee-script

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

Fork me on GitHub

Select a topic

Chapters

Markdown Templates

The rdiscount gem/library is required to render Markdown templates:

```
## You'll need to require rdiscount in your app
require "rdiscount"

get '/' do
  markdown :index
end
```

Renders `./views/index.markdown` (`md` and `mkd` are also valid file extensions).

It is not possible to call methods from markdown, nor to pass locals to it. You therefore will usually use it in combination with another rendering engine:

```
erb :overview, :locals => { :text => markdown(:introduction) }
```

Note that you may also call the markdown method from within other templates:

```
%h1 Hello From Haml!
%p= markdown(:greetings)
```

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

RSS Feed

The builder gem used for creating XML markup is required in this recipe.

Let's first write our basic app to begin with:

```
require 'sinatra'
require 'builder' # dont for get this!

get '/rss' do
  @posts = # ... find posts
  builder :rss
end
```

Then, assuming that your site url is `http://liftoff.msfc.nasa.gov/`, we will set up our view at `views/rss.builder`:

```
xml.instruct! :xml, :version => '1.0'
xml.rss :version => "2.0" do
  xml.channel do
    xml.title "Liftoff News"
    xml.description "Liftoff to Space Exploration."
    xml.link "http://liftoff.msfc.nasa.gov/"

    @posts.each do |post|
      xml.item do
        xml.title post.title
        xml.link "http://liftoff.msfc.nasa.gov/posts/#{post.id}"
        xml.description post.body
        xml.pubDate Time.parse(post.created_at.to_s).rfc822()
```

```
          xml.guid "http://liftoff.msfc.nasa.gov/posts/#{post.id}"
        end
      end
    end
  end
```

This will render the RSS inline, directly from the handler.

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