# Sinatra Recipes

*Community contributed recipes and techniques*

Fork me on GitHub

Select a topic

## Embedded Applications

Although Sinatra is mainly used for web-applications and quick prototyping, one of it's popular usage is as an API end-point for various applications built on Rails and other frameworks like EventMachine. For example, it is used for the web-portion of the iTunes Store on MacOSX. This section introduces some methods on embedding Sinatra in another framework.

- Event-machine

## 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



---

# Embedding Sinatra within EventMachine

EventMachine is a very useful tool and sometimes you need to add a web-interface on top of it. Yes, EM does support this out of the box, but it can be ugly and hard to work with. Why not use something that everyone already knows and loves like Sinatra?

Below is a (working) code-sample for running a simple HelloWorld Sinatra app within EventMachine. I've also provided a simple example of deferring tasks within your Sinatra call.

```
require 'eventmachine'
require 'sinatra/base'
require 'thin'


# This example shows you how to embed Sinatra into your EventMachine
# application. This is very useful if you're application needs some
# sort of API interface and you don't want to use EM's provided
# web-server.

def run(opts)

  # Start he reactor
  EM.run do

    # define some defaults for our app
    server  = opts[:server] || 'thin'
    host    = opts[:host]   || '0.0.0.0'
    port    = opts[:port]   || '8181'
    web_app = opts[:app]
```

```
    # create a base-mapping that our application will set at. If I
    # have the following routes:
    #
    #   get '/hello' do
    #     'hello!'
    #   end
    #
    #   get '/goodbye' do
    #     'see ya later!'
    #   end
    #
    # Then I will get the following:
    #
    #   mapping: '/'
    #   routes:
    #     /hello
    #     /goodbye
    #
    #   mapping: '/api'
    #   routes:
    #     /api/hello
    #     /api/goodbye
    dispatch = Rack::Builder.app do
      map '/' do
        run web_app
      end
    end

    # NOTE that we have to use an EM-compatible web-server. There
    # might be more, but these are some that are currently available.
    unless ['thin', 'hatetepe', 'goliath'].include? server
      raise "Need an EM webserver, but #{server} isn't"
    end

    # Start the web server. Note that you are free to run other tasks
    # within your EM instance.
    Rack::Server.start({
      app:    dispatch,
      server: server,
      Host:   host,
      Port:   port
    })
  end
end

# Our simple hello-world app
class HelloApp < Sinatra::Base
  # threaded - False: Will take requests on the reactor thread
  #            True:  Will queue request for background thread
  configure do
    set :threaded, false
```

```
    end

    # Request runs on the reactor thread (with threaded set to false)
    get '/hello' do
      'Hello World'
    end

    # Request runs on the reactor thread (with threaded set to false)
    # and returns immediately. The deferred task does not delay the
    # response from the web-service.
    get '/delayed-hello' do
      EM.defer do
        sleep 5
      end
      'I\'m doing work in the background, but I am still free to take requests'
    end
  end

  # start the application
  run app: HelloApp.new
```

You can run this simply with the command:

```
ruby em-sinatra-test.rb   # em-sinatra-test.rb is the filename of the above-code
```

You should also be able to test that it is working correctly with the following `ab` command:

```
ab -c 10 -n 100 http://localhost:8181/delayed-hello
```

If this finishes in “zero point something” seconds, then you have successfully setup Sinatra to run within EM and you are taking requests on the event-loop and deferring tasks to the background. If it takes any longer than that, then you are most likely taking requests in the background which means when the EM queue fills up, you can't process your sinatra requests (not a good thing!). Make sure that you have threaded set to false and then try again.

---

## 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