Building web framework with Rack

Marcin Kulik

EuRuKo, 2010/05/30

I am:

- Senior developer @ Lunar Logic Polska - agile Ruby on Rails development services
- Working with web for 10 years
- Using Ruby, Python, Java and others - in love with Ruby since 2006

Open Source contributor:

- CodeRack.org - Rack middleware repository
- Open File Fast - Netbeans and JEdit plugin
- racksh - console for Rack apps
- and many more (check github.com/sickill)

Why would you need another framework?

"Because world needs yet another framework ;-)" - Tomash

No, you probably don't need it actually :)

- Several mature frameworks
- Tens of custom/experimental ones
- "Don't reinvent the wheel", right?
- But...

But it's so easy that you should at least try

- Rack provides everything you'll need, is extremely simple but extremely powerful
- It will help you to better understand HTTP
- It will make you better developer
- Your custom framework will be the fastest one *
- It's fun! A lot of fun :)

What is Rack?

- Ruby web applications interface
- library

Simplest Rack application

```
run lambda do |env|
  [200, { "Content-type" => "text/plain" }, ["Hello"]]
end
```

Simplest Rack middleware

```
class EurukoMiddleware
  def initialize(app)
    @app = app
  end

  def call(env)
    env['euruko'] = 2010
    @app.call
  end
end
```

Let's transform it into framework!

How does typical web framework look like?

- Rails
- Merb
- Pylons
- Django
- Rango

Looks like MVC, more or less

What features we'd like to have?

- dependencies management
- RESTful routing
- controllers (session, flash messages)
- views (layouts, templates, partials)
- ORM
- authentication
- testing
- console

Available Rack middleware and tools we can use

(1/8) Gem dependency management

bundler

"A gem to bundle gems"

github.com/carlhuda/bundler

```
# Gemfile

source "http://gemcutter.org"
gem "rack"

# config.ru

require "bundler"
Bundler.setup
Bundler.require
```

(2/8) Routing

Usher

"Pure ruby general purpose router with interfaces for rails, rack, email or choose your own adventure"

github.com/joshbuddy/usher

```
# Gemfile

gem "usher"

# config.ru

require APP_ROOT / "config" / "router.rb"

run Foobar::Router
```

```
# config/router.rb

module Foobar
  Router = Usher::Interface.for(:rack) do
    get('/').to(HomeController.action(:welcome)).name(:root)
    add('/login').to(SessionController.action(:login)).name(:
    get('/logout').to(SessionController.action(:logout)).name
    ...
    default ExceptionsController.action(:not_found) # 404
  end
end
```

(3/8) Controller

Let's build our base controller

- every action is valid Rack endpoint
- value returned from action becomes body of the response

```ruby
# lib/base_controller.rb

module Foobar
  class BaseController
    def call(env)
      @request = Rack::Request.new(env)
      @response = Rack::Response.new
      resp_text = self.send(env['x-rack.action-name'])
      @response.write(resp_text)
      @response.finish
    end

    def self.action(name)
      lambda do |env|
        env['x-rack.action-name'] = name
        self.new.call(env)
      end
    end
  end
end
```

```
# config.ru

require APP_ROOT / "lib" / "base_controller.rb"
Dir[APP_ROOT / "app" / "controllers" / "*.rb"].each do |f|
  require f
end
```

Now we can create UsersController

```
# app/controllers/users_controller.rb

class UsersController < Foobar::BaseController
  def index
    "Hello there!"
  end
end
```

Controllers also need following:

- session access
- setting flash messages
- setting HTTP headers
- redirects
- url generation

rack-contrib

"Contributed Rack Middleware and Utilities"

github.com/rack/rack-contrib

rack-flash

"Simple flash hash implementation for Rack apps"

nakajima.github.com/rack-flash

```
# Gemfile

gem "rack-flash"
gem "rack-contrib", :require => 'rack/contrib'

# config.ru

use Rack::Flash
use Rack::Session::Cookie
use Rack::MethodOverride
use Rack::NestedParams
```

```ruby
# lib/base_controller.rb

module Foobar
  class BaseController
    def status=(code); @response.status = code; end

    def headers; @response.header; end

    def session; @request.env['rack.session']; end

    def flash; @request.env['x-rack.flash']; end

    def url(name, opts={}); Router.generate(name, opts); end

    def redirect_to(url)
      self.status = 302
      headers["Location"] = url
      "You're being redirected"
    end
  end
end
```

Now we can use #session, #flash and #redirect_to

```
# app/controllers/users_controller.rb

class UsersController < Foobar::BaseController
  def openid
    if session["openid.url"]
      flash[:notice] = "Cool!"
      redirect_to "/cool"
    else
      render
    end
  end
end
```

(4/8) Views

Tilt

"Generic interface to multiple Ruby template engines"

github.com/rtomayko/tilt

```
# Gemfile

gem "tilt"
```

```
# lib/base_controller.rb

module Foobar
  class BaseController
    def render(template=nil)
      template ||= @request.env['x-rack.action-name']
      views_path = "#{APP_ROOT}/app/views"
      template_path =
        "#{views_path}/#{self.class.to_s.underscore}/" +
          "#{template}.html.erb"
      layout_path =
        "#{views_path}/layouts/application.html.erb"
      Tilt.new(layout_path).render(self) do
        Tilt.new(template_path).render(self)
      end
    end
  end
end
```

(5/8) ORM

DataMapper

"DataMapper is a Object Relational Mapper written in Ruby. The goal is to create an ORM which is fast, thread-safe and feature rich."

datamapper.org

```
# Gemfile

gem "dm-core"
gem "dm-..."

# app/models/user.rb

class User
  include DataMapper::Resource

  property :id, Serial
  property :login, String, :required => true
  property :password, String, :required => true
end

# config.ru

Dir[APP_ROOT / "app" / "models" / "*.rb"].each do |f|
  require f
end
```

(6/8) Authentication

Warden

"General Rack Authentication Framework"

github.com/hassox/warden

```
# Gemfile

gem "warden"

# config.ru

use Warden::Manager do |manager|
  manager.default_strategies :password
  manager.failure_app =
    ExceptionsController.action(:unauthenticated)
end

require "#{APP_ROOT}/lib/warden.rb"
```

```
# lib/warden.rb

Warden::Manager.serialize_into_session do |user|
  user.id
end

Warden::Manager.serialize_from_session do |key|
  User.get(key)
end

Warden::Strategies.add(:password) do
  def authenticate!
    u = User.authenticate(
          params["username"],
          params["password"]
        )
    u.nil? ? fail!("Could not log in") : success!(u)
  end
end
```

```
# lib/base_controller.rb

module Foobar
  class BaseController
    def authenticate!
      @request.env['warden'].authenticate!
    end

    def logout!(scope=nil)
      @request.env['warden'].logout(scope)
    end

    def current_user
      @request.env['warden'].user
    end
  end
end
```

Now we can guard our action:

```
# app/controllers/users_controller.rb

class UsersController < Foobar::BaseController
  def index
    authenticate!
    @users = User.all(:id.not => current_user.id)
    render
  end
end
```

(7/8) Testing

rack-test

*"**Rack::Test** is a small, simple testing API for Rack apps. It can be used on its own or as a reusable starting point for Web frameworks and testing libraries to build on."*

github.com/brynary/rack-test

```
# Gemfile

gem "rack-test"
```

```
require "rack/test"

class UsersControllerTest < Test::Unit::TestCase
  include Rack::Test::Methods

  def app
    Foobar::Router.new
  end

  def test_redirect_from_old_dashboard
    get "/old_dashboard"
    follow_redirect!

    assert_equal "http://example.org/new_dashboard",
                 last_request.url
    assert last_response.ok?
  end
end
```

(8/8) Console

racksh (aka Rack::Shell)

"***racksh*** *is a console for Rack based ruby web applications. It's like Rails* script/console *or Merb's* merb -i, *but for any app built on Rack*"

github.com/sickill/racksh

Installation

```
gem install racksh
```

Example racksh session

```
$ racksh
Rack::Shell v0.9.7 started in development environment.
>> $rack.get "/"
=> #<Rack::MockResponse:0xb68fa7bc @body="<html>...",
   @headers={"Content-Type"=>"text/html", "Content-Length"=>"
   @status=200, ...
>> User.count
=> 123
```

Questions?

That's it!

Example code available at: github.com/sickill/example-rack-framework

email: marcin.kulik at gmail.com / www: ku1ik.com / twitter: @sickill