CGI: Ruby's Bare Metal

Jan 5, 2015

How simple can you make a web request?

Happy 2015 everyone! For 2015, I wanted to spend some time documenting and automating my business as much as possible. Ezra Z's and James Golick's recent passing was a reminder to myself about life: *hope for the best but plan for the worst*.

My biggest technical task was then to automate the onboarding of a new Sidekiq Pro (http://sidekiq.org/) customer. If I pass away next week, I want people to still be able to purchase and renew their subscription so my wife and child have recurring income they can count on for the next few years. Essentially I want to automate day-to-day operations.

My Sidekiq Pro server is as simple as humanly possible: it's running only Apache. Perfect for serving static files but how do I handle an arbitrary request? That's when I asked myself: **How simple can you make a web request?** The requirements are straightforward: Stripe will call my server with a subscription event when someone starts or stops their Sidekiq Pro subscription. I need a script to perform the magic to grant/revoke access and send the customer an email with access details. This call will only happen a few times a day, max.

This is a perfect case for going down to the bare metal (http://www.boblee.com/images/bear_guitar.jpg) and using the oldest web technology: CGI (http://www.ruby-doc.org/stdlib-2.2.0/libdoc/cgi/rdoc/CGI.html).

Common Gateway Interface

CGI was the first standard for tying Unix and the Web together. The Unix programming model says a process should take input on STDIN and output on STDOUT. CGI allows a webserver like Apache to call an external script

with the details of a web request as STDIN. The script then outputs the HTTP response back as STDOUT. Ruby's `cgi` library will parse the request coming from STDIN and provides some response output helpers your code can use to generate HTML responses.

In my case, Stripe POSTs a blob of JSON in the request body. Since I'm responding to the Stripe robot, it only needs to see a 200 OK response — no fancy view rendering layer required.

```
#!/usr/bin/env ruby
require 'json'
require 'cgi'

cgi = CGI.new
# CGI tries to parse the request body as form parameters so a
# blob of JSON awkwardly ends up as the one and only parameter key.
stripe_event = JSON.parse(cgi.params.keys.first)

do_the_magic(stripe_event) # magic happens right here

cgi.out("status" => "OK", "type" => "text/plain", "connection" => "close") do
  "Success"
end
```

I configured Apache to know to execute my CGI script by adding this inside the vhost configuration:

```
ScriptAlias /stripe/ /opt/stripe/

<Directory /opt/stripe/>
  Require all granted
</Directory>
```

Now if I request `http://server/stripe/event.rb` (`http://server/stripe/event.rb`), Apache will call `/opt/stripe/event.rb`.

Look at what I'm not running: puma or unicorn, rails or sinatra, redis or memcached, postgres or mysql, bundler, capistrano, etc. The real thing is using 3-4 gems. That's it. The script runs in a few seconds and then exits. Nothing to keep running 24/7 and nothing to monitor. Deployment means using `scp` to copy the .rb file to the server. I don't even have to restart anything upon deploy because nothing was running in the first place!

Reality Check

CGI certainly isn't the right solution for every problem: each request starts a new Ruby process so there's a small bit of overhead but for systems which expect little traffic but require maximum reliability, it's something worth considering. There's a higher performance variant of CGI called FastCGI (http://www.fastcgi.com/) which solves the performance overhead by keeping a process running 24/7.

Ultimately plain old CGI solved my requirements: only Apache is running 24/7 and new Sidekiq Pro customers now get their license information within seconds of purchase, making everyone happy!

Mike Perham

Ruby, OSS and the Internet
mperham@gmail.com (mailto:mperham@gmail.com)

mperham (https://github.com/mperham)
mperham (https://twitter.com/mperham)