Smoke

Smoke

Ben Schwarz

INTRODUCTION

Smoke is a DSL written in Ruby. Its primary goal is to be a very simple interface to query many online sources, chop them up into usable pieces and use them in ways that the source provider may have never imagined.

This document is aimed to serve as a guide for usage to get you started.

The concept

The concept comes from using Yahoo Pipes to make web based mash ups: Get a list of tv shows for my torrent client, compile a recipe book or make tools to give me a list of albums that artists in my music library are about to be released.

DIVING STRAIGHT IN

Say you wanted to get your twitter stream, doesn't sound too hard? It isn't!

The first thing that you'll need to do is discover your source. Smoke supports YQL, XML, JSON and RSS / Atom.

```
require 'smoke'
Smoke.data(:twitter) do
  url "http://twitter.com/users/show.json?screen_name=benschwarz"
end
```

Now that the source has been defined you can use it like this:

```
Smoke.twitter.output
```

This will give you an array of hashes, each hash will be a tweet.

Right now you're probably sitting there smugly thinking to yourself "yeah, I could do that myself pretty easily."

Well, you smug bastard, there is some more we can do. Perhaps you'd like to open an interface to pull twitter feeds for multiple users? Maybe you'd like to strip out tweets that were directed at someone?

```
Smoke.data(:twitter) do
  prepare do
    url "http://twitter.com/users/show.json?screen_name=#{username}"
  end
  discard :text, /@\w/
end
```

Now that you've suggested that a username is required for the query. You need to send through a username.

```
Smoke.twitter.username(:benschwarz).output
```

The smoke DSL is pretty easy to pickup. All methods are well documented using rdoc, so exploring the API should be easy.

COOL STUFF SMOKE DOES

Output formats

Not only does Smoke query a web service and automagically translate Json or XML to an array of ruby hashes, it will also output in Json or Yaml.

```
Smoke.twitter.output(:yaml)
```

Smart requests

All requests ask the web service on the other end to gzip its data before it delivers it back to smoke. This means that less data is transferred - You can compress plain text pretty successfully.

Not all services support / implement gzip, but for those who do, its a clear win.

CONFIGUATION

Smoke can be configured to a couple of things differently, at the time of writing the major functional differences are:

- *logging*
- *caching*
- *setting the user agent*

If you wanted to enable logging and masquerade as something other than Smoke (say, a browser - you cheeky bastard), you could add something like this before your Smoke definitions:

```
Smoke.configure do |c|
  c[:enable_logging] = true
  c[:user_agent] = "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0)" # har, har
end
```

TRANSPARENT REQUEST CACHING

Now that you've pulled your sources together, you launch your web site and go out to dinner, you might even brag to your nerdy company about how quickly you pulled some new cool shit together. Then you check your twitter mentions.

Everyone is saying that the site you just linked to is 500ing. Damn. That happened to me, I rescued nothing, doh!

The responsible party was twitters api request limit (per hour), we need to make less requests. Enter, caching.

Caching is disabled by default, however, its really easy to turn it on:

```
Smoke.configure do |c|
  c[:cache][:enabled] = true
  c[:cache][:store] = :memory
end
```

The above example will use a ruby hash to store the contents of your requests in memory. In some cases, this might be okay, however, there are more options.

Caching is implemented using wycats moneta, it provides a common interface to many hash storage types:

- *file*
- *memcache*
- *tokyo cabinet*
- *a ruby hash (memory)*

- *mongodb*

There are more, but if you want to know more about them, I suggest you read a bit about moneta

If you wanted to use Smoke caching with Heroku, you could setup something like this (you'll need memcached enabled on your heroku instance):

```
Smoke.configure do |c|
  c[:cache][:enabled] = true
  c[:cache][:store] = :memcache
  c[:cache][:options] = {
    :server => ENV['MEMCACHE_SERVERS'].split(','),
    :namespace => ENV['MEMCACHE_NAMESPACE']
  }
end
```

Expiry

Smoke defaults to 1800 seconds for expiry (thats 30 minutes), you can change the expiry time by adding another option to the configuration hash.

```
Smoke.configure do |c|
  c[:cache][:enabled] = true
  c[:cache][:store] = :memory
  c[:cache][:expiry] = 3600 # 1 Hour
end
```