Typhoeus

Bill Doughty

Austin.rb
May 19, 2011

Typhoeus

What?

- Ruby GEM that optimizes concurrent HTTP client requests

How?

- Leverages the ubiquitous, powerful, native *libcurl* library using Ruby C bindings

Why?

- Efficient. Robust. Idiomatic. Cool name.
- Active and well documented **open-source** project

Typhoeus

Who?

- Created by Paul Dix (author of *feedzirra*)
- Currently maintained on Github by *dbalatero*

Where?

- https://github.com/dbalatero/typhoeus

When should you consider Typhoeus?

- You have a batch of HTTP client requests that can be processed simultaneously
- Processing order is not important
- Minimum execution time is important
- Detailed exception handling is a requirement
- Need fine-grained control over timeouts
- Want an easy way to stub external HTTP services during development and testing

How does Typhoeus work?

- Uses cURL to perform actual HTTP client *request*s
  - *libcurl* is a native library and fast; can utilize multiple kernel threads behind the scenes
  - *libcurl* "multi" API efficiently handles multiple concurrent requests in a non-blocking way

- *Response* handlers are defined using Ruby blocks
  - Blocks maintain programatic context for each request allowing responses to be handled in a non-deterministic order

- Batches of requests are processed using a "*hydra*"
  - Hydras dispatch requests to *libcurl* in batches and coordinate delivery of each response to the appropriate handler

Using Typhoeus

class Typhoeus::Request

- Instantiated by your application to define an individual request to be processed
- All necessary data to perform the request is stored in the attributes of the instance
- You supply a block as a callback that will get executed when your request has been processed

class Typhoeus::Response

- Instantiated by Typhoeus and passed to your request's callback block when processing of the request is complete
- Contains all response data pertinent to an individual request

Using Typhoeus

class Typhoeus::Hydra

- Instantiated by your application to handle concurrent processing of multiple requests
- Abstracts underlying cURL integration with a few simple methods
- Allows your application to create arbitrarily large batches of requests and transparent divides them into smaller batches to be handled by cURL.
- Manages other nifty features like memoization, caching, and stubbing.

Example

```
# the request object request = Typhoeus::Request.new("http://www.pauldix.net", :body => "this is a request body", :method => :post, :headers => {:Accept => "text/html"}, :timeout => 100, # milliseconds :cache_timeout => 60, # seconds :params => {:field1 => "a field"}) # we can see from this that the first argument is the url. the second is a set of options. # the options are all optional. The default for :method is :get. Timeout is measured in milliseconds. # cache_timeout is measured in seconds. # Run the request via Hydra. hydra = Typhoeus::Hydra.new hydra.queue (request) hydra.run # the response object will be set after the request is run response = request.response response.code # http status code response.time # time in seconds the request took response.headers # the http headers response.headers_hash # http headers put into a hash response.body # the response body
```

Diagram

Example

Testing Support

Typhoeus has built-in stubbing support for remote services.

- Greatly simplifies and speeds up testing your request handling logic!

```
hydra = Typhoeus::Hydra.new response = Response.new(:code => 200, :headers => "", :body => "{'name' : 'paul'}", :time => 0.3) hydra.stub(:get, "http://localhost:3000/users/1").and_return (response) request = Typhoeus::Request.new("http://localhost: 3000/users/1") request.on_complete do |response| JSON.parse (response.body) end hydra.queue request hydra.run
```

Additional Features

- Request Memoization
  - Persistent only within the context of one Hydra run, a lot like ActiveRecord's built in *find* caching.
- Response Caching
  - Persistent across Hydra runs according to underlying cache's expiration strategy.
  - Supports *memcached* API out of the box.
  - Other cache frameworks easily supported by overriding cache API read/write methods.
- Supports SSL, basic authentication, and other authentication methods supported by *libcurl*

Links

These slides

- http://bit.ly/typho_slides

Github Repo

- https://github.com/dbalatero/typhoeus

Discussion Group

- http://groups.google.com/group/typhoeus

QUESTIONS

Some Personal Philosophy

Documentation is a great way to contribute to an Open Source project!

- Like grease is to a physical machine; reduces friction
- Eases understanding and therefore promotes adoption
- Little clarifications or corrections can make a big difference
- Easier when you are busy but still want to contribute something