Ruby master - Feature #5064

HTTP user-agent class

07/21/2011 01:10 PM - drbrain (Eric Hodel)

Status:	Assigned	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:		

Description

Currently there are some problems with Net::HTTP:

- Too many ways to use (user confusion)
- No automatic support for HTTPS (must conditionally set use_ssl)
- No automatic support for HTTPS peer verification (must be manually set)
- Single-connection oriented
- No support for redirect-following
- No support for HTTP/1.1 persistent connection retry (RFC 2616 8.1.4)
- No automatic support for HTTP proxies
- No automatic support for authentication (must be set per-request)

Additionally the style of the API of Net::HTTP makes it difficult to take advantage of persistent connections. The user has to store the created connection and manually handle restarting the connection if it has timed out or is closed by the server.

RFC 2616 8.1.1 has a large section explaining the benefits of persistent connections, but while Net::HTTP implements persistent connections they could be easier for users to implement with next work.

I've implemented support for many of these additional features of Net::HTTP in various projects and I'd like Ruby to have the features required to make a useful HTTP user-agent built-in.

The agent should have the following responsibilities:

- Make or reuse connections based on [host, port, SSL enabled]
- Automatically enable SSL for https URIs
- Automatically enable SSL peer verification for SSL connections
- Limit number of persistent connections per host
- Follow redirects
- Retry when a persistent connection fails
- Automatically configure proxies
- Automatically use authentication
- Callbacks for various options connect

The agent may add the following responsibilities:

- Default headers for all requests
- HTTP cookies
- Tracking history
- Logging

I don't think any of these features are critical as they are implementable by users via callbacks.

The agent would have the following configurable items:

- Number of connections per host
- Depth of redirects followed
- Persistent connection retries (none, HTTP/1.1 (default), always)
- Proxy host, port, user, password

I think the class should be called Net::HTTP::Agent.

Basic use would look something like this:

uris = [
URI('http://example/1'),

URI('http://example/2'),
URI('https://secure.example'),
]

agent = Net::HTTP::Agent.new

uris.map do |uri|
agent.get uri # Returns Net::HTTPResponse
end

For special requests a Net::HTTPRequest could be constructed:

req = Net::HTTP::Get.new uri.request_uri
# do something special with req

agent.request req

The agent should support GET, POST, etc. directly through API methods. I think the API should look something like this:

def get uri_or_string, query = nil, headers = nil
Same for other requests with no body
#
query may be a Hash or String
How query param vs query string in URI is used is undecided

def post uri_or_string, data, headers = nil
same for other requests with a body
#
data may be a String, IO or Hash
How data format is chosen is undecided

SSL options, proxy options, timeouts and similar options should exist on Net::HTTP::Agent and be set on new connections as they are made.

I've implemented most of these features in mechanize as Mechanize::HTTP::Agent. The Agent class in mechanize is bigger than is necessary and would need to be cut-down for inclusion in Ruby as Net::HTTP::Agent

https://github.com/tenderlove/mechanize/blob/master/lib/mechanize/http/agent.rb

Mechanize depends on net-http-persistent to provide HTTP/1.1 retry support and connection management:

https://github.com/drbrain/net-http-persistent/blob/master/lib/net/http/persistent.rb

Portions of net-http-persistent should be patches of Net::HTTP, for example #idempotent? #can_retry?, #reset and portions of #request. Other parts (connection management) should be moved to Net::HTTP::Agent.

net-http-persistent provides a separate connection list per thread. I would like Net::HTTP::Agent to be multi-thread friendly but implementing this in another way would be fine.

As an addendum, open-uri and mechanize should be written to take advantage of Net::HTTP::Agent on order to guide useful implementation.

Related issues:

Related to Ruby master - Feature #5461: Add pipelining to Net::HTTP	**Assigned**	
Related to Ruby master - Feature #6482: Add URI requested to Net::HTTP reques...	**Closed**	**05/23/2012**

History

#1 - 07/25/2011 04:13 AM - naruse (Yui NARUSE)

Interesting proposal.

I also sometimes make such agent so I agree the concept.
(there is still a question: why stdlib)

NOTE: [ruby-core:38158] may be related.

#2 - 07/25/2011 06:00 PM - naruse (Yui NARUSE)

=begin
NOTE: #2567 can be implemented as a part of this proposal.
=end

#3 - 07/26/2011 04:12 AM - drbrain (Eric Hodel)

I have had discussions where people say "Net::HTTP's API is not very good" and people seem to want a better way to use Net::HTTP without trying out one of the many Net::HTTP::Agent-like extensions to the base API.

Having a recommended way to perform HTTP access in ruby will be useful to the community since they don't have to do any additional research. Also users will be able perform basic HTTP requests for multiple servers and will gain the efficiencies of persistent connections without extra work.

The biggest issue people have is that they need to pass the host, port and SSL params for every HTTP connection made. The nature of the API means that most people make a new connection for every request. Since this is confusing for new users and annoying for experienced users I think it should be addressed in stdlib.

I think #2567 should be implemented in Net::HTTP itself. After my tickets for the 1.9.3 release are finished I can port some code from mechanize to fix this.

#4 - 07/26/2011 10:44 AM - jrochkind (jonathan rochkind)

I think this is a good API which will avoid the need for a third party library for API convenience. (You didn't go into detail about 'timeout' configuration, but a SINGLE timeout param should exist, possibly along with separate read/open timeouts if someone really wants different values for each.).

However, I think there may, at least in the past, be other performance-related reasons people needed to get third party libraries, especially when the app involves threading -- but NOT related to the the hypothetical Agent itself and it's concurrency model. Not sure if I have this right, or if these are still issues in 1.9.3?

1. Huge performance problem of the way timeouts are implemented.

2. global-interpreter-blocking nature of DNS lookups, instead of a non-blocking select DNS lookup.

Can anyone confirm whether I have this right or not, and if it's still an issue in 1.9.3? If either of these are still issues, then a new standard library Agent addition ought to solve them too to really provide a solid library obviating need for third party libraries for standard use cases.

#5 - 07/26/2011 04:09 PM - drbrain (Eric Hodel)

There is no longer a problem with timeouts in Net::HTTP as timeout is no longer used for reads. If you are connecting so fast that timeout is a greater problem than the three-way TCP handshakes and slow-start I think there may be a problem with your program's design.

If you are having problems with DNS blocking then require 'resolv/replace' to get non-blocking DNS lookups. Non-blocking DNS in ruby has been available back to 1.6.

I don't think either of these points are relevant to the addition of a user-agent library. Net::HTTP should remain the same other than minor changes to ease the implementation of the user-agent library. If you find they are still issues for your workload please file separate issues.

#6 - 07/26/2011 04:29 PM - akr (Akira Tanaka)

2011/7/26 Eric Hodel drbrain@segment7.net:

> If you are having problems with DNS blocking then require 'resolv/replace' to get non-blocking DNS lookups. Non-blocking DNS in ruby has been available back to 1.6.

Ruby invokes getaddrinfo() without GVL since 1.9.2.
This means, even without resolv/replace, DNS lookup doesn't block other
threads if the platform have getaddrinfo().
--
Tanaka Akira

#7 - 07/27/2011 12:19 AM - steveklabnik (Steve Klabnik)

I would love a simpler, more use-case focused HTTP library in the Ruby standard library. We should make the simple case very simple, and let people fall back to the full Net::HTTP if they need all the advanced and/or unusual stuff.

#8 - 07/27/2011 06:53 AM - normalperson (Eric Wong)

jonathan rochkind jonathan@dnil.net wrote:

> 1. Huge performance problem of the way timeouts are implemented.

I would like to split the HTTP open_timeout into two components:
connect_timeout and dns_timeout

connect_timeout would be trivial to implement

1. global-interpreter-blocking nature of DNS lookups, instead of a non-blocking select DNS lookup.

Like akr said, it's not an issue now. The performance issue is that getaddrinfo() in standard C libraries doesn't provide configurable timeouts so we have to use the nasty timeout library that spawns a thread every time we need to do a DNS lookup...

--
Eric Wong

#9 - 03/25/2012 05:17 PM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

#10 - 03/25/2012 09:22 PM - duerst (Martin Dürst)

Comment, mostly to drbrain (Eric Hodel):
At the developers' meeting today in Akihabare, we assigned this issue to Matz to decide on the main direction, but we thought that it might help Matz to get a shorter summary of the proposal.

#11 - 03/27/2012 09:56 AM - drbrain (Eric Hodel)

=begin
I hope this shortened version is simple enough:

New class named Net::HTTP::Agent with this API:

agent = Net::HTTP::Agent.new

Built in ways to make HTTP requests similar to Net::HTTP:

agent.head "http://example/some/page"
agent.get "http://example/some/page"
agent.post "http://example/search", "q" => "some search"

including put, patch, etc. For all request methods, redirects will be followed automatically.

HTTPS requests for public Internet hosts must work with no extra user configuration.

Requests for multiple hosts must be allowed without creating new instances of the agent.

Requests to the same host and port should re-use an existing connection if possible.

For special requests, a user can use #request like Net::HTTP:

uri = URI "http://example/search"
post = Net::HTTP::Post.new uri.request_uri
post.set_form_data "q" => "some search"
post["X-Custom-Header"] = "some value"

agent.request uri, post

To enable a proxy:

agent.proxy = :ENV # get proxy from the environment
agent.proxy = "http://proxy.example:8000" # specific proxy

Net::HTTP::Agent should use Net::HTTP.

=end

#12 - 11/24/2012 09:11 AM - mame (Yusuke Endoh)

- Target version changed from 1.9.4 to 2.6

#13 - 12/25/2017 06:15 PM - naruse (Yui NARUSE)

- *Target version deleted (2.6)*