

Dismiss

Join GitHub today

GitHub is home to over 20 million developers working together to host and review code, manage projects, and build software together.

Sign up

A cURL wrapper

95 commits

1 branch

16 releases

8 contributors

MIT

Branch: master

New pull request

Find file

Clone or download

chriso Merge pull request #55 from katerberg/master ... Latest commit 550e6a2 on 6 Mar

LICENSE	Update the license	2 years ago
README.md	Updates docs for fail option	8 months ago
errors.js	Determine error from exit code & pipe stderr to the main process	6 years ago
index.js	remove insecure options and docs	a year ago
package.json	1.0.1	a year ago
spawn.js	Limit the amount of spawns per tick	6 years ago
useragents.js	Fix UA export	2 years ago

README.md

curlrequest is a node wrapper for the command line **curl(1)**

```
$ npm install curlrequest
```

Why?

Curl has significantly more features than any of the node.js request libraries. A lot of the logic that you would build on top of a node.js request library (rate limiting, URL globbing, uploading/downloading file, better proxy support) already exists in curl

- Curl is mature, stable, and fast
- Separate requests/processes take advantage of all CPUs
- Spawning processes is relatively cheap and non-blocking
- Better control over connect, request, and retry timeouts. If a request has hung just kill the process

Note: don't use this if you need to stream the response - use [mikeal/request](#) instead

Usage

Make a request with curl - callback receives (err, stdout) on request completion

```
var curl = require('curlrequest');

curl.request(options, callback);
```

Note that you can also call `curl.request(url, callback)` which is shorthand for `curl.request({ url: url }, callback)`.

To setup default options and return a function that can be used later

```
var request = curl.request(default_options);

request([options ,] callback);
```

Options

`url`

The request url.

`method` - *default: GET*

The request method.

`encoding` - *default: utf8*

Encode the response body as either `utf` or `ascii`. Set to `null` return a buffer.

`headers` - *default: {}*

Set request headers, e.g. `headers: { accept: 'text/*' }`

`data` - *default: false*

An object containing data to urlencode and then POST.

`useragent` - *default:*

Set the request user-agent.

`location` - *default: true*

Whether to follow 30x redirects or not.

`redirects` - *default: 3*

The maximum amount of redirects to follow before failing with error "retries".

`retries` - *default: 0*

How many times to retry the request in the case of failure.

`timeout` - *default: false*

The maximum amount of seconds the request can take before failing with error "timeout".

`scope` - *default: {}*

The scope to call the callback in.

`require` - *default: null*

Pass a string or regular expression to search for in the response body. If there's no match, fail the request with "required string not found". You can also pass an array of strings / regexps to search for where only one has to match.

`require_not`

The inverse of `require` - fail if the response contains a string.

`process` - *default: false*

Pass in a function which modifies the response body before sending it to the callback. Useful if you need to modify the response in some way before a higher level library has the chance to modify it.

`file` - *default: false*

Open a file and process it like a request response, useful if using temporary files.

`stderr` - *default: false*

Pipe the stderr of each curl process to the main process. Set this to a string to write stderr to a file.

`pretend` - *default: false*

Useful if you want to see what curl command is to be executed without actually making the request.

`fail` - *default: false*

When set to true, a failing response body will be returned as the first parameter of the callback.

```
curl.request({ url: 'http://google.com', pretend: true }, function (err, stdout, meta) {  
  console.log('%s %s', meta.cmd, meta.args.join(' '));  
});
```

`curl_path` - *default: 'curl'*

Use this to specify an alternative path for curl.

Passing options directly to curl

Any additional options are sent as command line options to curl. See `man curl` or `curl --manual` for a detailed description of options and usage.

Example 1. Include response headers in the output

```
var options = { url: 'google.com', include: true };  
  
curl.request(options, function (err, parts) {  
  parts = parts.split('\r\n');  
  var data = parts.pop()  
    , head = parts.pop();  
});
```

Example 2. Limit the download speed of a transfer

```
var options = {  
  url: 'example.com/some/large/file.zip'  
  , 'limit-rate': '500k'  
  , encoding: null  
};  
  
curl.request(options, function (err, file) {  
  //file is a Buffer  
});
```

Example 3. See what's going on under the hood

```
var options = {  
  url: 'google.com'  
  , verbose: true  
  , stderr: true  
};  
  
curl.request(options, function (err, data) {  
  //..  
});
```

License

Copyright (c) 2016 Chris O'Hara cohara87@gmail.com

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.