# connect-history-api-fallback

Middleware to proxy requests through a specified index page, useful for Single Page Applications that utilise the HTML5 History API.

## Table of Contents

- [Introduction](#introduction) - [Usage](#usage) - [Options](#options) - [index](#index) - [rewrites](#rewrites) - [verbose](#verbose) - [htmlAcceptHeaders](#htmlacceptheaders) - [disableDotRule](#disabledotrule) ## Introduction Single Page Applications (SPA) typically only utilise one index file that is accessible by web browsers: usually `index.html`. Navigation in the application is then commonly handled using JavaScript with the help of the [HTML5 History API](http://www.w3.org/html/wg/drafts/html/master/single-page.html#the-history-interface). This results in issues when the user hits the refresh button or is directly accessing a page other than the landing page, e.g. `/help` or `/help/online` as the web server bypasses the index file to locate the file at this location. As your application is a SPA, the web server will fail trying to retrieve the file and return a \*404 - Not Found\* message to the user. This tiny middleware addresses some of the issues. Specifically, it will change the requested location to the index you specify (default being `/index.html`) whenever there is a request which fulfills the following criteria: 1. The request is a `GET` or `HEAD` request 2. which accepts `text/html`, 3. is not a direct file request, i.e. the requested path does not contain a `.` (DOT) character and 4. does not match a pattern provided in options.rewrites (see options below) ## Usage The middleware is available through NPM and can easily be added. ``` npm install --save connect-history-api-fallback ``` Import the library ```javascript var history = require('connect-history-api-fallback'); ``` Now you only need to add the middleware to your application like so ```javascript var connect = require('connect'); var app = connect() .use(history()) .listen(3000); ``` Of course you can also use this piece of middleware with express: ```javascript var express = require('express'); var app = express(); app.use(history()); ``` ## Options You can optionally pass options to the library when obtaining the middleware ```javascript var middleware = history({}); ``` ### index Override the index (default `/index.html`). This is the request path that will be used when the middleware identifies that the request path needs to be rewritten. This is not the path to a file on disk. Instead it is the HTTP request path. Downstream connect/express middleware is responsible to turn this rewritten HTTP request path into actual responses, e.g. by reading a file from disk. ```javascript history({ index: '/default.html' }); ``` ### rewrites Override the index when the request url matches a regex pattern. You can either rewrite to a static string or use a function to transform the incoming request. The following will rewrite a request that matches the `/\/soccer/` pattern to `/soccer.html`. ```javascript history({ rewrites: [ { from: /\/soccer/, to: '/soccer.html'} ] }); ``` Alternatively functions can be used to have more control over the rewrite process. For instance, the following listing shows how requests to `/libs/jquery/jquery.1.12.0.min.js` and the like can be routed to `./bower\_components/libs/jquery/jquery.1.12.0.min.js`. You can also make use of this if you have an API version in the URL path. ```javascript history({ rewrites: [ { from: /^\/libs\/.\*$/, to: function(context) { return '/bower\_components' + context.parsedUrl.pathname; } } ] }); ``` The function will always be called with a context object that has the following properties: - \*\*parsedUrl\*\*: Information about the URL as provided by the [URL module's](https://nodejs.org/api/url.html#url\_url\_parse\_urlstr\_parsequerystring\_slashesdenotehost) `url.parse`. - \*\*match\*\*: An Array of matched results as provided by `String.match(...)`. - \*\*request\*\*: The HTTP request object. ### verbose This middleware does not log any information by default. If you wish to activate logging, then you can do so via the `verbose` option or by specifying a logger function. ```javascript history({ verbose: true }); ``` Alternatively use your own logger ```javascript history({ logger: console.log.bind(console) }); ``` ### htmlAcceptHeaders Override the default `Accepts:` headers that are queried when matching HTML content requests (Default: `['text/html', '\*/\*']`). ```javascript history({ htmlAcceptHeaders: ['text/html', 'application/xhtml+xml'] }) ``` ### disableDotRule Disables the dot rule mentioned above: > […] is not a direct file request, i.e. the requested path does not contain a `.` (DOT) character […] ```javascript history({ disableDotRule: true }) ```