Package 'httpcache'

August 29, 2016

Type Package

Title Query Cache for HTTP Clients

Description In order to improve performance for HTTP API clients, 'httpcache' provides simple tools for caching and invalidating cache. It includes the HTTP verb functions GET, PUT, PATCH, POST, and DELETE, which are drop-in replacements for those in the 'httr' package. These functions are cache-aware and provide default settings for cache invalidation suitable for RESTful APIs; the package also enables custom cache-management strategies. Finally, 'httpcache' includes a basic logging framework to facilitate the measurement of HTTP request time and cache performance.

Version 0.1.8

Date 2016-08-22

Author Neal Richardson [aut, cre]

Maintainer Neal Richardson <neal.p.richardson@gmail.com>

URL https://github.com/nealrichardson/httpcache

BugReports https://github.com/nealrichardson/httpcache/issues

License MIT + file LICENSE

Depends R (>= 3.0.0)

Imports httr (>= 1.0.0), digest

Suggests knitr, testthat

RoxygenNote 5.0.1

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2016-08-29 20:48:53

R topics documented:

`cache-management` *Manage the HTTP cache*

Description

These functions turn the cache on and off and clear the contents of the query cache.

Usage

```
cacheOn()

cacheOff()

clearCache()
```

Value

Nothing. Functions are run for their side effects.

`cached-http-verbs` *Cache-aware versions of httr verbs*

Description

These functions set, read from, and bust the HTTP query cache. They wrap the similarly named functions in the httr package and can be used as drop-in replacements for them.

Usage

```
GET(url, ...)

PUT(url, ..., drop = dropCache(url))

POST(url, ..., drop = dropOnly(url))

PATCH(url, ..., drop = dropCache(url))

DELETE(url, ..., drop = dropCache(url))
```

Arguments

url	character URL of the request
...	additional arguments passed to the httr functions
drop	For PUT, PATCH, POST, and DELETE, code to be executed after the request. This is intended to be for supplying cache-invalidation logic. By default, POST drops cache only for the specified url (i.e. dropOnly), while the other verbs drop cache for the request URL and for any URLs nested below it (i.e. dropCache).

Details

GET checks the cache before making an HTTP request, and if there is a cache miss, it sets the response from the request into the cache for future requests. The other verbs, assuming a more or less RESTful API, would be assumed to modify server state, and thus they should trigger cache invalidation. They have default cache-invalidation strategies, but you can override them as desired.

Value

The corresponding httr response object, potentially read from cache

See Also

dropCache

---

cacheLogSummary *Summarize cache performance from a log*

Description

Summarize cache performance from a log

Usage

```
cacheLogSummary(logdf)
```

Arguments

logdf	A logging data.frame, as loaded by link{loadLogfile}.

Value

A list containing counts of cache hit/set/drop events, plus a cache hit rate.

---

dropCache *Invalidate cache*

Description

These functions let you control cache invalidation. dropOnly invalidates cache only for the specified URL. dropPattern uses regular expression matching to invalidate cache. dropCache is a convenience wrapper around dropPattern that invalidates cache for any resources that start with the given URL.

Usage

```
dropCache(x)

dropOnly(x)

dropPattern(x)
```

Arguments

x	character URL or regular expression

Value

Nothing. Functions are run for their side effects.

---

halt *Stop, log, and no call*

Description

Wrapper around stop that logs the error message and then stops with call.=FALSE by default.

Usage

```
halt(..., call. = FALSE)
```

Arguments

...	arguments passed to stop
call.	logical: print the call? Default is FALSE, unlike stop

Value

Nothing. Raises an error.

---

loadLogfile *Read in a httpcache log file*

Description

Read in a httpcache log file

Usage

```
loadLogfile(filename, scope = c("CACHE", "HTTP"))
```

Arguments

filename	character name of the log file, passed to read.delim
scope	character optional means of selecting only certain log messages. By default, only "CACHE" and "HTTP" log messages are kept. Other logged messages, such as "ERROR" messages from halt, will be dropped from the resulting data.frame.

Value

A data.frame of log results.

---

logMessage *Log a message*

Description

Log a message

Usage

```
logMessage(...)
```

Arguments

...	Strings to pass to cat

Value

Nothing

`requestLogSummary` *Summarize HTTP requests from a log*

Description

Summarize HTTP requests from a log

Usage

```
requestLogSummary(logdf)
```

Arguments

`logdf`	A logging data.frame, as loaded by `link{loadLogfile}`.

Value

A list containing counts of HTTP requests by verb, as well as summaries of time spent waiting on HTTP requests.

`startLog` *Enable logging*

Description

Enable logging

Usage

```
startLog(filename = "", append = FALSE)
```

Arguments

`filename`	character: a filename/path where the log can be written out. If `""`, messages will print to stdout (the screen). See `cat`.
`append`	logical: if the file already exists, append to it? Default is `FALSE`, and if not in append mode, if the `filename` exists, it will be deleted.

Value

Nothing.

uncached *Context manager to temporarily turn cache off if it is on*

Description

If you don't want to store the response of a GET request in the cache, wrap it in `uncached()`. It will neither read from nor write to cache. However, `uncached` will not invalidate cache records, if present.

Usage

```
uncached(...)
```

Arguments

...	Things to evaluate with caching off

Value

Whatever ... returns.

Examples

```
uncached(GET("http://httpbin.org/get"))
```

Index