Package: request (via r-universe)

August 27, 2024

Title High Level 'HTTP' Client

Description High level and easy 'HTTP' client for 'R' that makes assumptions that should work in most cases. Provides functions for building 'HTTP' queries, including query parameters, body requests, headers, authentication, and more.

Version 0.1.6.91

License MIT + file LICENSE

URL https://github.com/sckott/request

BugReports https://github.com/sckott/request/issues

Encoding UTF-8

Depends httr (>= 1.2.0)

Imports methods, stats, utils, curl (>= 2.2), jsonlite (>= 1.1), magrittr (>= 1.5), lazyeval (>= 0.2.0), whisker (>= 0.3-2), R6 (>= 2.2.0), tibble (>= 1.2), digest, rappdirs

Suggests testthat, data.table

RoxygenNote 7.1.0

Repository https://sckott.r-universe.dev

RemoteUrl https://github.com/sckott/request

RemoteRef HEAD

RemoteSha ab4ceb85093c3e2bde6be3d1cce61d78440c29a5

Contents

`request-package` *request*

Description

Easy http

Author(s)

Scott Chamberlain <`myrmecocystus@gmail.com`>

Examples

```
## Not run:
## Build API routes
### Works with full or partial URLs
api('https://api.github.com/')
api('http://api.gbif.org/v1')
api('api.gbif.org/v1')

### Works with ports, full or partial
api('http://localhost:9200')
api('localhost:9200')
api(':9200')
api('9200')

## The above are not passed through a pipe, so simply define a URL, but don't
## do a request. To make an http request, you can either pipe a url or
## partial url to e.g., \code{\link{api}}, or call \code{\link{http}}
'https://api.github.com/' %>% api()
### Or
api('https://api.github.com/') %>% http()

# Non-standard evaluation (NSE)
api('https://api.github.com/') %>%
  api_path(repos, ropensci, rgbif, issues) %>%
  peep

# Standard evaluation (SE)
```

```
api('https://api.github.com/') %>%
  api_path_('repos', 'ropensci', 'rgbif', 'issues') %>%
  peep

## Templating
repo_info <- list(username = 'craigcitro', repo = 'r-travis')
api('https://api.github.com/') %>%
  api_template(template = 'repos/{{username}}/{{repo}}/issues', data = repo_info) %>%
  peep

## End(Not run)
```

`api` *API base url and endpoint setup*

Description

API base url and endpoint setup

Usage

```
api(x)
```

Arguments

`x`	A URL

See Also

Other dsl: `api_body()`, `api_config()`, `api_error_handler()`, `api_path()`, `api_query()`, `api_template()`, `auth`

Examples

```
## Not run:
# Set base url
## works with full or partial URLs
api('https://api.github.com/')
api('http://api.gbif.org/v1')
api('api.gbif.org/v1')

## works with ports, full or partial
api('http://localhost:9200')
api('localhost:9200')
api(':9200')
api('9200')
api('9200/stuff')

# set paths
## NSE
```

```
api('https://api.github.com/') %>%
  api_path(repos, ropensci, rgbif, issues)
## SE
api('https://api.github.com/') %>%
  api_path_('repos', 'ropensci', 'rgbif', 'issues')

# template
repo_info <- list(username = 'craigcitro', repo = 'r-travis')
api('https://api.github.com/') %>%
  api_template(template = 'repos/{{username}}/{{repo}}/issues', data = repo_info)

## End(Not run)
```

api_body *Query construction*

Description

Query construction

Usage

```
api_body(.data, ..., body_value = NULL)

api_body_(.data, ..., .dots, body_value = NULL)
```

Arguments

`.data`	Result of a call to `api`
`...`	Comma separated list of unquoted variable names. These are combined into a list and passed to whatever http method is used downstream
`body_value`	one of the following: • FALSE: No body • NULL: An empty body • "": A length 0 body • upload_file("path/"): The contents of a file. The mime type will be guessed from the extension, or can be supplied explicitly as the second argument to `upload_file()` • A character or raw vector: sent as is in body. Use content_type to tell the server what sort of data you are sending.
`.dots`	Used to work around non-standard evaluation

See Also

Other dsl: `api_config()`, `api_error_handler()`, `api_path()`, `api_query()`, `api_template()`, `api()`, `auth`

Examples

```
## Not run:
## NSE
dd <- api("https://httpbin.org/post")
dd %>% api_body(body_value = NULL) %>% http("POST")
dd %>% api_body(body_value = "") %>% http("POST")

## other named parameters are passed as form values
dd %>% api_body(x = hello) %>% http("POST")

# upload a file
file <- "~/some_test.txt"
cat("hello, world", file = file)
dd %>% api_body(x = upload_file("~/some_test.txt")) %>% http("POST")

# A named list
dd %>% api_body(x = hello, y = stuff) %>% http("POST")

## SE
dd %>% api_body_(x = "hello", y = "stuff") %>% http("POST")

## End(Not run)
```

`api_cache` *Caching helper*

Description

Caching helper

Usage

```
api_cache(.data, dir = NULL, ...)
```

Arguments

`.data`	Result of a call to `api`
`dir`	(character) Directory to cache in. Uses `rappdirs::user_cache_dir()` by default
`...`	ignored

Examples

```
## Not run:
# cache
## first call is slower
api('http://localhost:5000') %>%
  api_path(get) %>%
  api_query(foo = "bar") %>%
  api_cache()

## second call is faster, pulling from cache
api('http://localhost:5000') %>%
  api_path(get) %>%
  api_query(foo = "bar") %>%
  api_cache()

# other egs
x <- api('api.crossref.org') %>%
  api_path(works) %>%
  api_query(rows = 1000) %>%
  api_cache()

## End(Not run)
```

---

api_config *Curl settings*

Description

Curl settings

Usage

```
api_config(.data, ...)
```

Arguments

.data	Result of a call to api
...	Comma separated list of unquoted variable names

See Also

Other dsl: api_body(), api_error_handler(), api_path(), api_query(), api_template(), api(), auth

Examples

```
## Not run:
# Config handler
api('http://api.crossref.org/works') %>%
 api_config(verbose(), progress()) %>%
 peep()

xx <- api('http://api.crossref.org') %>%
  api_path(works, 10.3897/zookeys.515.9459) %>%
  api_config(verbose())

## End(Not run)
```

api_error_handler *Error handler*

Description

Error handler

Usage

```
api_error_handler(.data, fun)
```

Arguments

.data	Result of a call to api
fun	A function, either defined in the session, or a function available in loaded or name-spaced packges

See Also

Other dsl: api_body(), api_config(), api_path(), api_query(), api_template(), api(), auth

Examples

```
## Not run:
# Use functions from httr
api('http://httpbin.org/status/503') %>%
 api_error_handler(stop_for_status)

api('http://httpbin.org/status/404') %>%
 api_error_handler(warn_for_status)

# Custom error handling functions
my_stop <- function(x) {
  if (x$status > 200) {
     warning("nope, try again", call. = FALSE)
  }
}
api("http://httpbin.org/status/404") %>%
 api_error_handler(my_stop)

## End(Not run)
```

api_headers *Headers*

Description

Headers

Usage

```
api_headers(.data, ..., .dots)

api_headers_(.data, ..., .dots)
```

Arguments

`.data`	Result of a call to `api`
`...`	Key value pairs of headers
`.dots`	Used to work around non-standard evaluation

Examples

```
## Not run:
api('https://api.github.com/') %>%
   api_config(verbose()) %>%
   api_headers(`X-FARGO-SEASON` = 3) %>%
   peep

api('http://httpbin.org/headers') %>%
   api_config(verbose()) %>%
   api_headers(`X-FARGO-SEASON` = 3, `X-NARCOS-SEASON` = 5)

## End(Not run)
```

api_path *API paths*

Description

API paths

Usage

```
api_path(.data, ..., .dots)

api_path_(.data, ..., .dots)
```

Arguments

`.data`	Result of a call to `api`
`...`	Comma separated list of unquoted variable names
`.dots`	Used to work around non-standard evaluation

See Also

Other dsl: `api_body()`, `api_config()`, `api_error_handler()`, `api_query()`, `api_template()`, `api()`, `auth`

Examples

```
## Not run:
# set paths
## NSE
api('https://api.github.com/') %>%
  api_path(repos, ropensci, rgbif, issues)
## SE
api('https://api.github.com/') %>%
  api_path_('repos', 'ropensci', 'rgbif', 'issues')

## End(Not run)
```

`api_query` *Query construction*

Description

Query construction

Usage

```
api_query(.data, ...)

api_query_(.data, ..., .dots)
```

Arguments

`.data`	Result of a call to `api`
`...`	Comma separated list of unquoted variable names
`.dots`	Used to work around non-standard evaluation

See Also

Other dsl: `api_body()`, `api_config()`, `api_error_handler()`, `api_path()`, `api_template()`, `api()`, `auth`

Examples

```
## Not run:
## NSE
api("http://api.plos.org/search") %>%
  api_query(q = ecology, wt = json, fl = 'id,journal') %>%
  peep

api("http://api.plos.org/search") %>%
  api_query(q = ecology, wt = json, fl = id, fl = journal) %>%
  peep

## SE
api("http://api.plos.org/search") %>%
  api_query_(q = "ecology", wt = "json", fl = 'id', fl = 'journal') %>%
  peep

## NSE
api("http://api.plos.org/search") %>%
  api_query(q = ecology, wt = json, fl = 'id,journal')
## SE
api("http://api.plos.org/search") %>%
  api_query_(q = "ecology", wt = "json", fl = 'id', fl = 'journal')

## End(Not run)
```

`api_template` *API path template*

Description

API path template

Usage

```
api_template(.data, template, data)
```

Arguments

`.data`	Result of a call to `api`
`template`	Template to contstruct API route
`data`	Data to pass to the template parameter

See Also

Other dsl: `api_body()`, `api_config()`, `api_error_handler()`, `api_path()`, `api_query()`, `api()`, `auth`

Examples

```
## Not run:
repo_info <- list(username = 'craigcitro', repo = 'r-travis')
api('https://api.github.com/') %>%
  api_template(template = 'repos/{{username}}/{{repo}}/issues', data = repo_info)

## End(Not run)
```

---

api_write *Write helper*

Description

Write helper

Usage

```
api_write(.data, file, overwrite = FALSE, ...)
```

Arguments

`.data`	Result of a call to `api`
`file`	(character) Full file path to write to
`overwrite`	(logical) Will only overwrite existing path if `TRUE`
`...`	ignored for now

Examples

```
## Not run:
## write to disk
ff <- tempfile(fileext = ".json")
api('https://api.github.com/') %>%
  api_path(repos, ropensci, rgbif, issues) %>%
  api_write(ff)
jsonlite::fromJSON(ff)

## End(Not run)
```

as_df *Attempt to convert list to a data.frame*

Description

Attempt to convert list to a data.frame

Usage

```
as_df(x, clean = TRUE)
```

Arguments

x	input, a list
clean	(logical) clean 0 length elements. Default: TRUE

Examples

```
mtlist <- apply(mtcars, 1, as.list)
as_df(mtlist)

mtlist <- apply(mtcars, 1, as.list)
mtlist[[1]]$mpg <- list()
as_df(mtlist)
```

auth *Authentication configuration/setup*

Description

Authentication configuration/setup

Usage

```
api_simple_auth(.data, user, pwd, type = "basic")

api_oauth2(
  .data,
  token = NULL,
  app_name = NULL,
  key = NULL,
  secret = NULL,
  base_url = NULL,
  authorize = NULL,
  access = NULL
)
```

```
api_oauth1(
  .data,
  token = NULL,
  app_name = NULL,
  key = NULL,
  secret = NULL,
  base_url = NULL,
  request = NULL,
  authorize = NULL,
  access = NULL
)
```

Arguments

`.data`	Result of a call to `api`
`user`	user name
`pwd`	password
`type`	type of HTTP authentication. Should be one of the following types supported by Curl: basic, digest, digest_ie, gssnegotiate, ntlm, ntlm_vn, any. Default: "basic" (the most common type)
`token`	An OAuth token
`app_name`	An OAuth application name
`key`	An OAuth key
`secret`	An OAuth secret key
`base_url`	option url to use as base for request, authorize and access urls.
`authorize`	url to send client to for authorisation
`access`	url used to exchange unauthenticated for authenticated token.
`request`	url used to request initial (unauthenticated) token. If using OAuth2.0, leave as NULL.

See Also

Other dsl: `api_body()`, `api_config()`, `api_error_handler()`, `api_path()`, `api_query()`, `api_template()`, `api()`

Examples

```
## Not run:
# simple authentication (user/password)
api('https://httpbin.org/basic-auth/user/passwd') %>%
  api_simple_auth(user = "user", pwd = "passwd")
## different auth type
api('https://httpbin.org/basic-auth/user/passwd') %>%
  api_simple_auth(user = "user", pwd = "passwd", type = "gssnegotiate")
```

```
# OAuth setup
## using a token
### fill in your own token
# api('https://api.github.com/') %>%
#   api_path(orgs, ropensci, events) %>%
#   api_oauth2(token = "<your token>") %>%
#   api_config(verbose())

# OAuth2
## using a app name, key, and secret combination
### uses a OAuth app set up by Hadley Wickham, which you'll auth against
# api('https://api.github.com/') %>%
#   api_oauth2(app_name = "github", key = "56b637a5baffac62cad9",
#       secret = "8e107541ae1791259e9987d544ca568633da2ebf",
#       base_url = "https://github.com/login/oauth",
#       authorize = "authorize", access = "access_token")

# OAuth1
# api('https://api.twitter.com/1.1/statuses/home_timeline.json') %>%
#  api_oauth1(app_name = "twitter", key = "afdafasdfs",
#      secret = "asdfasdfasfasfsdf",
#      base_url = "https://api.twitter.com/oauth/",
#      request = "request_token", authorize = "authenticate", access = "access_token")

# Request some data with oauth2 via Github
## put in your username and password
# api('https://api.github.com/') %>%
#  api_simple_auth(user = "<foo>", pwd = "<bar>")

## End(Not run)
```

http *Make a HTTP request*

Description

Make a HTTP request

Usage

```
http(req, method = "GET")

http_client(req)
```

Arguments

req	A req class object
method	(character) Pick which HTTP method to use. Only GET and POST for now. Default: GET

Details

By default, a GET request is made. Will fix this soon to easily allow a different HTTP verb.

The `http` function makes the request and gives back the parsed result. Whereas, the `http_client` function makes the request, but gives back the raw R6 class object, which you can inspect all parts of, modify, etc.

Examples

```
## Not run:
# high level - http()
api('https://api.github.com/') %>%
  api_path(repos, ropensci, rgbif, commits) %>%
  http()

# low level - http_client()
res <- api('https://api.github.com/') %>%
  api_path(repos, ropensci, rgbif, commits) %>%
  http_client()
res$count()
res$body()
res$status()
res$result
res$links
res$parse()

# Specify HTTP verb
## POST
api("https://httpbin.org/post") %>%
   api_body(x = "A simple text string") %>%
   http("POST")

## PUT
api("https://httpbin.org/put") %>%
   api_body(x = "A simple text string") %>%
   http("PUT")

## End(Not run)
```

`peep` *Peek at a query*

Description

Peek at a query

Usage

```
peep(.data)
```

Arguments

`.data`	(list) input, using higher level interface

Examples

```
## Not run:
api('https://api.github.com/') %>% peep
api('https://api.github.com/') %>%
  api_path(repos, ropensci, rgbif, issues) %>%
  peep

repo_info <- list(username = 'craigcitro', repo = 'r-travis')
api('https://api.github.com/') %>%
  api_template(template = 'repos/{{username}}/{{repo}}/issues', data = repo_info) %>%
  peep

api("http://api.plos.org/search") %>%
  api_query(q = ecology, wt = json, fl = id, fl = journal) %>%
  peep

## End(Not run)
```

Index