Package ‘fauxpas’

April 13, 2020

Title  HTTP Error Helpers

Description  HTTP error helpers. Methods included for general purpose HTTP error handling, as well as individual methods for every HTTP status code, both via status code numbers as well as their descriptive names. Supports ability to adjust behavior to stop, message or warning. Includes ability to use custom whisker template to have any configuration of status code, short description, and verbose message. Currently supports integration with 'crul', 'curl', and 'httr'.

Version  0.5.0

License  MIT + file LICENSE

URL  https://docs.ropensci.org/fauxpas,
     https://github.com/ropensci/fauxpas

BugReports  https://github.com/ropensci/fauxpas/issues

VignetteBuilder  knitr

Encoding  UTF-8

Language  en-US

Imports  R6 (>= 2.1.2), httpcode (>= 0.3.0), whisker

Suggests  crul (>= 0.5.0), curl (>= 2.2), httr (>= 1.2.0), testthat, knitr, rmarkdown

RoxygenNote  7.1.0

X-schema.org-applicationCategory  Web

X-schema.org-keywords  http, https, API, web-services, curl, errors, error

X-schema.org-isPartOf  https://ropensci.org

NeedsCompilation  no

Author  Scott Chamberlain [aut, cre] (<https://orcid.org/0000-0003-1444-9135>)

Maintainer  Scott Chamberlain <myrmecocystus@gmail.com>

Repository  CRAN

Date/Publication  2020-04-13 16:10:02 UTC
**R topics documented:**

- fauxpas-package .................................................. 2
- Error ................................................................. 2
- Error-Classes ....................................................... 5
- find_error_class ................................................... 6
- http100 ............................................................. 7

### Description

HTTP Error Helpers

### Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

### Error

#### Error class

### Description

Error class

### Arguments

- **behavior**
  
  Behavior of the error. default: auto. See Details

- **message_template**
  
  A message template. optional. use whisker templating. names to use are: reason and status. use in template like \{reason\} and \{status\}. Note that \{message\} that is used in message_template_verbose will be ignored here.

- **call**
  
  (logical) indicating if the call should become part of the error message. Default: FALSE

- **message_template_verbose**
  
  A verbose message template. optional. use whisker templating. names to use are: reason, status, message. use in template like \{reason\}, \{status\}, and \{message\}. Note that this is ignored here, but is used in the HTTP* methods (e.g. HTTPBadRequest)

- **muffle**
  
  (logical) whether to not respond when status codes in 1xx-3xx series. Default: FALSE
**Error**

Details

Methods

- **do(response, mssg)**
  Execute condition, whether it be message, warning, or error.
  - response: is any response from **curl**, **curl**, or **httr**
  - mssg: character string message to include in call. ignored if template does not have a message entry

- **set_behavior(behavior)**
  Set behavior, same as setting behavior on initializing with $new()

behavior parameter options

- stop - use stop
- warning - use warning
- message - use message
- auto - toggle between stop and message depending on the HTTP status code series. Defaults will be:
  - 1xx: message
  - 2xx: message
  - 3xx: message
  - 4xx: stop
  - 5xx: stop

Of course, you can always override the defaults.

See Also

- http, Error-Classes

Examples

```r
Error$new()
# reset behavior
(z <- Error$new())
z$set_behavior("warning")
z

if (requireNamespace("curl")) {
  library("curl")
  res <- HttpClient$new("https://httpbin.org/status/418")$get()

  # stop
  (x <- Error$new(behavior = "stop"))
  ## Not run: x$do(res)
```
# warn
(x <- Error$new(behavior = "warning"))
x$do(res)

# do vs. do_verbose
x <- HTTPRequestURITooLong$new(behavior = "stop")
res <- HttpClient$new("https://httpbin.org/status/414")$get()
## Not run:
http414(res)
## with template
http414(res, message_template = "{{status}}\n --> {{reason}}")
x$do(res)
x$doVerbose(res)

## End(Not run)

# service unavailable
x <- HTTPServiceUnavailable$new(behavior = "stop")
res <- HttpClient$new("https://httpbin.org/status/503")$get()
## Not run:
x$do(res)
x$doVerbose(res)

## End(Not run)

# message template
y <- Error$new(message_template = "{{reason}} ............ {{status}}")
res <- HttpClient$new("https://httpbin.org/status/418")$get()
## Not run:
y$do(res)

## End(Not run)

yy <- Error$new(message_template = "{{status}}\n --> {{reason}}")
yy$message_template
## Not run:
yy$do(res)

## End(Not run)

## with verbose message
library(cru)
res <- HttpClient$new("https://httpbin.org/status/401")$get()
yy <- HTTPUnauthorized$new()
zz <- HTTPUnauthorized$new(
  message_template = "HTTP({{status}}): {{message}}"
)
yy$message_template; zz$message_template
## Not run:
yy$do(res)
zz$do(res)
yy$doVerbose(res)
zz$doVerbose(res)
## End(Not run)

yy <- Error$new(
  message_template = "HTTP({{status}}): {{reason}}
  {{message}}"
)

yy$message_template

## Not run: yy$do(res)

# muffle responses
(x <- Error$new(muffle = TRUE))
res <- crul::HttpClient$new("https://httpbin.org/status/226")$get()

z <- x$do(res)
z

---

### Error-Classes

#### Individual error classes

#### Description

These error classes are for each HTTP error, and inherit from the `Error` class in this package.

#### Details

In addition to what’s available in `Error`, these classes have a single variable `mssg` that is the very verbose complete message describing the HTTP condition in detail. You can include that message in your condition by using `do_verbose` (see below).

#### Methods

In addition to the methods documented in `Error`, these methods also have:

- `do_verbose(response, template)`

  Execute condition, whether it be message, warning, or error.

  - `response`: is any response from `crul`, `curl`, or `httr` Execute condition, whether it be message, warning, error, or your own custom function. This method uses `message_template_verbose`, and uses it’s default value.

  - `template`: a template to use for the verbose message, see `Error` for details

#### See Also

`Error`, `http`

#### Examples

```r
if (requireNamespace("crul")) {

  library("crul")
  res <- HttpClient$new("https://httpbin.org/status/414")$get()

  # Example usage
  x$do(res)
  x$do_verbose(res, "Verbose message")
}
```
```r
x <- HTTPRequestURITooLong$new()
x
## Not run:
x$do(res)
x$do_verbose(res)
## End(Not run)

# behavior
x <- HTTPRequestURITooLong$new(behavior = "warning")
## Not run:
x$do(res)
x$do_verbose(res)
## End(Not run)

# with message template
(x <- HTTPRequestURITooLong$new(
  message_template = "{{{reason}}} ............ {{status}}",
  message_template_verbose = "{{reason}} .>.>.>.>.> {{status}}\n {{message}}"
))
## Not run:
x$do(res)
x$do_verbose(res)
## End(Not run)
```

---

**find_error_class**

Find error classes

### Description

Find error classes

### Usage

```r
find_error_class(status_code)
```

### Arguments

- `status_code` (numeric, integer) A status code
Value

an object of class R6ClassGenerator. call $new() to initialize a new instance

See Also

Error, Error-Classes

Examples

find_error_class(414)
find_error_class(418)
find_error_class(505)

# initialize the class
find_error_class(418)$new()

# not found
## Not run: find_error_class(999)

http100

higher level error wrappers

Description

higher level error wrappers

Usage

http100(response, behavior = "auto", message_template, muffle = FALSE)
http101(response, behavior = "auto", message_template, muffle = FALSE)
http102(response, behavior = "auto", message_template, muffle = FALSE)
http200(response, behavior = "auto", message_template, muffle = FALSE)
http201(response, behavior = "auto", message_template, muffle = FALSE)
http202(response, behavior = "auto", message_template, muffle = FALSE)
http203(response, behavior = "auto", message_template, muffle = FALSE)
http204(response, behavior = "auto", message_template, muffle = FALSE)
http205(response, behavior = "auto", message_template, muffle = FALSE)
http206(response, behavior = "auto", message_template, muffle = FALSE)
http207(response, behavior = "auto", message_template, muffle = FALSE)
http208(response, behavior = "auto", message_template, muffle = FALSE)
http226(response, behavior = "auto", message_template, muffle = FALSE)
http300(response, behavior = "auto", message_template, muffle = FALSE)
http301(response, behavior = "auto", message_template, muffle = FALSE)
http302(response, behavior = "auto", message_template, muffle = FALSE)
http303(response, behavior = "auto", message_template, muffle = FALSE)
http304(response, behavior = "auto", message_template, muffle = FALSE)
http305(response, behavior = "auto", message_template, muffle = FALSE)
http306(response, behavior = "auto", message_template, muffle = FALSE)
http307(response, behavior = "auto", message_template, muffle = FALSE)
http308(response, behavior = "auto", message_template, muffle = FALSE)
http400(response, behavior = "auto", message_template, muffle = FALSE)
http401(response, behavior = "auto", message_template, muffle = FALSE)
http402(response, behavior = "auto", message_template, muffle = FALSE)
http403(response, behavior = "auto", message_template, muffle = FALSE)
http404(response, behavior = "auto", message_template, muffle = FALSE)
http405(response, behavior = "auto", message_template, muffle = FALSE)
http406(response, behavior = "auto", message_template, muffle = FALSE)
http407(response, behavior = "auto", message_template, muffle = FALSE)
http408(response, behavior = "auto", message_template, muffle = FALSE)
http409(response, behavior = "auto", message_template, muffle = FALSE)
http410(response, behavior = "auto", message_template, muffle = FALSE)
http411(response, behavior = "auto", message_template, muffle = FALSE)
http100

http412(response, behavior = "auto", message_template, muffle = FALSE)
http413(response, behavior = "auto", message_template, muffle = FALSE)
http414(response, behavior = "auto", message_template, muffle = FALSE)
http415(response, behavior = "auto", message_template, muffle = FALSE)
http416(response, behavior = "auto", message_template, muffle = FALSE)
http417(response, behavior = "auto", message_template, muffle = FALSE)
http418(response, behavior = "auto", message_template, muffle = FALSE)
http419(response, behavior = "auto", message_template, muffle = FALSE)
http420(response, behavior = "auto", message_template, muffle = FALSE)
http422(response, behavior = "auto", message_template, muffle = FALSE)
http423(response, behavior = "auto", message_template, muffle = FALSE)
http424(response, behavior = "auto", message_template, muffle = FALSE)
http425(response, behavior = "auto", message_template, muffle = FALSE)
http426(response, behavior = "auto", message_template, muffle = FALSE)
http428(response, behavior = "auto", message_template, muffle = FALSE)
http429(response, behavior = "auto", message_template, muffle = FALSE)
http431(response, behavior = "auto", message_template, muffle = FALSE)
http440(response, behavior = "auto", message_template, muffle = FALSE)
http444(response, behavior = "auto", message_template, muffle = FALSE)
http449(response, behavior = "auto", message_template, muffle = FALSE)
http450(response, behavior = "auto", message_template, muffle = FALSE)
http451(response, behavior = "auto", message_template, muffle = FALSE)
http494(response, behavior = "auto", message_template, muffle = FALSE)
http495(response, behavior = "auto", message_template, muffle = FALSE)
http496(response, behavior = "auto", message_template, muffle = FALSE)
http497(response, behavior = "auto", message_template, muffle = FALSE)
http498(response, behavior = "auto", message_template, muffle = FALSE)
http499(response, behavior = "auto", message_template, muffle = FALSE)
http500(response, behavior = "auto", message_template, muffle = FALSE)
http501(response, behavior = "auto", message_template, muffle = FALSE)
http502(response, behavior = "auto", message_template, muffle = FALSE)
http503(response, behavior = "auto", message_template, muffle = FALSE)
http504(response, behavior = "auto", message_template, muffle = FALSE)
http505(response, behavior = "auto", message_template, muffle = FALSE)
http506(response, behavior = "auto", message_template, muffle = FALSE)
http507(response, behavior = "auto", message_template, muffle = FALSE)
http508(response, behavior = "auto", message_template, muffle = FALSE)
http509(response, behavior = "auto", message_template, muffle = FALSE)
http510(response, behavior = "auto", message_template, muffle = FALSE)
http511(response, behavior = "auto", message_template, muffle = FALSE)
http598(response, behavior = "auto", message_template, muffle = FALSE)
http599(response, behavior = "auto", message_template, muffle = FALSE)
http(response, behavior = "auto", message_template, muffle = FALSE)

Arguments

response  The result of a call via curl, curl, or httr
behavior Behavior of the error. default: auto. See Details
message_template A message template. optional. use whisker templating. names to use are: reason and status. use in template like {{reason}} and {{status}}. Note that {{message}} that is used in message_template_verbose will be ignored here.
muffle (logical) whether to not respond when status codes in 1xx-3xx series. Default: FALSE
behavior parameter options

- stop - use stop
- warning - use warning
- message - use message
- auto - toggle between stop and message depending on the HTTP status code series. Defaults will be:
  - 1xx: message
  - 2xx: message
  - 3xx: message
  - 4xx: stop
  - 5xx: stop

Of course, you can always override the defaults.

using package curl

curl responses are simple lists, so we have little to go on to make sure it’s a response from the `curl` package. We check for list names internally but of course you could pass in a list with the right named elements, while the values are complete nonsense, in which case we’ll probably fail badly. There’s not much we can do.

Note

These `http*` methods only use `$do` and not `$do_verbose`.

See Also

`Error, Error-Classes`

Examples

```r
if (requireNamespace("crul")) {
  library("crul")
  res <- HttpClient$new("https://httpbin.org/status/418")$get()
  ## Not run: http(res)
  http(res, behavior = "warning")
  http(res, behavior = "message")

  res <- HttpClient$new("https://httpbin.org/status/414")$get()
  ## Not run: http414(res)
  http(res, behavior = "warning")
  http(res, behavior = "message")

  res <- HttpClient$new("https://httpbin.org/asdfafadsf")$get()
  ## Not run: http404(res)
  http(res, behavior = "warning")
  http(res, behavior = "message")
}
```
if (requireNamespace("curl")) {
    library("curl")
    h <- curl::new_handle()
    curl::handle_setopt(h)
    res <- curl::curl_fetch_memory("https://httpbin.org/status/418", h)
    ## Not run: http(res)
    http(res, behavior = "warning")
    http(res, behavior = "message")
}

if (requireNamespace("httr")) {
    library("httr")
    res <- GET("https://httpbin.org/status/418")
    ## Not run: http(res)
    http(res, behavior = "warning")
    http(res, behavior = "message")
}

# muffle responses
if (requireNamespace("crul")) {
    library("crul")
    res201 <- HttpClient$new("https://httpbin.org")$get("status/201")
    res404 <- HttpClient$new("https://httpbin.org")$get("status/404")
    # status codes < 300 CAN be muffled - i.e., return the http response object
    http(res201, muffle = TRUE)
    # status codes > 300 CAN NOT be muffled - i.e., return the http response object
    ## Not run:
    http(res404, muffle = TRUE)

    ## End(Not run)
}
Index

*Topic package
  fauxpas-package, 2

Error, 2, 5, 7, 11
Error-Classes, 5

fauxpas (fauxpas-package), 2
fauxpas-package, 2
find_error_class, 6

http, 3, 5
http (http100), 7
http100, 7
http101 (http100), 7
http102 (http100), 7
http200 (http100), 7
http201 (http100), 7
http202 (http100), 7
http203 (http100), 7
http204 (http100), 7
http205 (http100), 7
http206 (http100), 7
http207 (http100), 7
http208 (http100), 7
http209 (http100), 7
http210 (http100), 7
http211 (http100), 7
http220 (http100), 7
http221 (http100), 7
http222 (http100), 7
http226 (http100), 7
http227 (http100), 7
http230 (http100), 7
http231 (http100), 7
http232 (http100), 7
http233 (http100), 7
http234 (http100), 7
http235 (http100), 7
http236 (http100), 7
http237 (http100), 7
http238 (http100), 7
http239 (http100), 7
http240 (http100), 7
http241 (http100), 7
http242 (http100), 7
http243 (http100), 7
http244 (http100), 7
http245 (http100), 7
http246 (http100), 7
http247 (http100), 7
http248 (http100), 7
http249 (http100), 7
http250 (http100), 7
http251 (http100), 7
http252 (http100), 7
http253 (http100), 7
http254 (http100), 7
http255 (http100), 7
http256 (http100), 7
http257 (http100), 7
http258 (http100), 7
http259 (http100), 7
http260 (http100), 7
http261 (http100), 7
http262 (http100), 7
http263 (http100), 7
http264 (http100), 7
http265 (http100), 7
http266 (http100), 7
http267 (http100), 7
http268 (http100), 7
http269 (http100), 7
http270 (http100), 7
http271 (http100), 7
http272 (http100), 7
http273 (http100), 7
http274 (http100), 7
http275 (http100), 7
http276 (http100), 7
http277 (http100), 7
http278 (http100), 7
http279 (http100), 7
http280 (http100), 7
http281 (http100), 7
http282 (http100), 7
http283 (http100), 7
http284 (http100), 7
http285 (http100), 7
http286 (http100), 7
http287 (http100), 7
http288 (http100), 7
http289 (http100), 7
http290 (http100), 7
http291 (http100), 7
http292 (http100), 7
http293 (http100), 7
http294 (http100), 7
http295 (http100), 7
http296 (http100), 7
http297 (http100), 7
http298 (http100), 7
http299 (http100), 7
http300 (http100), 7
http301 (http100), 7
http302 (http100), 7
http303 (http100), 7
http304 (http100), 7
http305 (http100), 7
http306 (http100), 7
http307 (http100), 7
http308 (http100), 7
http309 (http100), 7
http310 (http100), 7
http311 (http100), 7
http312 (http100), 7
http313 (http100), 7
http314 (http100), 7
http315 (http100), 7
http316 (http100), 7
http317 (http100), 7
http318 (http100), 7
http319 (http100), 7
http320 (http100), 7
http321 (http100), 7
http322 (http100), 7
http323 (http100), 7
http324 (http100), 7
http325 (http100), 7
http326 (http100), 7
http327 (http100), 7
http328 (http100), 7
http329 (http100), 7
http330 (http100), 7
http331 (http100), 7
http332 (http100), 7
http333 (http100), 7
http334 (http100), 7
http335 (http100), 7
http336 (http100), 7
http337 (http100), 7
http338 (http100), 7
http339 (http100), 7
http340 (http100), 7
http341 (http100), 7
http342 (http100), 7
http343 (http100), 7
http344 (http100), 7
http345 (http100), 7
http346 (http100), 7
http347 (http100), 7
http348 (http100), 7
http349 (http100), 7
http350 (http100), 7
http351 (http100), 7
http352 (http100), 7
http353 (http100), 7
http354 (http100), 7
http355 (http100), 7
http356 (http100), 7
http357 (http100), 7
http358 (http100), 7
http359 (http100), 7
http360 (http100), 7
http361 (http100), 7
http362 (http100), 7
http363 (http100), 7
http364 (http100), 7
http365 (http100), 7
http366 (http100), 7
http367 (http100), 7
http368 (http100), 7
http369 (http100), 7
http370 (http100), 7
http371 (http100), 7
http372 (http100), 7
http373 (http100), 7
http374 (http100), 7
http375 (http100), 7
http376 (http100), 7
http377 (http100), 7
http378 (http100), 7
http379 (http100), 7
http380 (http100), 7
http381 (http100), 7
http382 (http100), 7
http383 (http100), 7
http384 (http100), 7
http385 (http100), 7
http386 (http100), 7
INDEX

HTTP400 Bad Request
HTTP401 Unauthorized
HTTP402 Payment Required
HTTP403 Forbidden
HTTP404 Not Found
HTTP405 Method Not Allowed
HTTP406 Not Acceptable
HTTP407 Proxy Authentication Required
HTTP408 Request Timeout
HTTP409 Conflict
HTTP410 Gone
HTTP411 Length Required
HTTP412 Precondition Failed
HTTP413 Request Entity Too Large
HTTP414 Request URI Too Long
HTTP415 Unsupported Media Type
HTTP416 Request Range Not Satisfiable
HTTP417 Expectation Failed
HTTP428 Precondition Required
HTTP429 Too Many Requests
HTTP431 Request Header Fields Too Large
HTTP449 Retry With
HTTP451 Unprocessable Entity
HTTP452 Processing Required
HTTP491 Client Certificates Required
HTTP492 HPParseError
HTTP493 HPParseException
HTTP494 HPParseFailure
HTTP495 HPParseTypeError
HTTP496 HPParseValueError
HTTP497 HPParseWarning
HTTP498 HPParseError
HTTP499 HPParseWarning
HTTP500 Internal Server Error
HTTP501 Not Implemented
HTTP502 Bad Gateway
HTTP503 Service Unavailable
HTTP504 Gateway Time Out
HTTP505 HTTP Version Not Supported
HTTP506 Service Unavailable (Temporarily)
HTTP507 Service Unavailable (Temporarily)
HTTP508 Service Unavailable (Temporarily)
HTTP509 Service Unavailable (Temporarily)
HTTP510 Service Unavailable (Temporarily)
HTTP511 Service Unavailable (Temporarily)
HTTP512 Service Unavailable (Temporarily)
HTTP513 Service Unavailable (Temporarily)
HTTP514 Service Unavailable (Temporarily)
HTTP515 Service Unavailable (Temporarily)
HTTP516 Service Unavailable (Temporarily)
HTTP517 Service Unavailable (Temporarily)
HTTP518 Service Unavailable (Temporarily)
HTTP519 Service Unavailable (Temporarily)
HTTP520 Service Unavailable (Temporarily)
HTTP521 Service Unavailable (Temporarily)
HTTP522 Service Unavailable (Temporarily)
HTTP523 Service Unavailable (Temporarily)
HTTP524 Service Unavailable (Temporarily)
HTTP525 Service Unavailable (Temporarily)
HTTP526 Service Unavailable (Temporarily)
HTTP527 Service Unavailable (Temporarily)
HTTP528 Service Unavailable (Temporarily)
HTTP529 Service Unavailable (Temporarily)
HTTP530 Service Unavailable (Temporarily)
HTTP531 Service Unavailable (Temporarily)
HTTP532 Service Unavailable (Temporarily)
HTTP533 Service Unavailable (Temporarily)
HTTP534 Service Unavailable (Temporarily)
HTTP535 Service Unavailable (Temporarily)
HTTP536 Service Unavailable (Temporarily)
HTTP537 Service Unavailable (Temporarily)
HTTP538 Service Unavailable (Temporarily)
HTTP539 Service Unavailable (Temporarily)
HTTP540 Service Unavailable (Temporarily)
HTTP541 Service Unavailable (Temporarily)
HTTP542 Service Unavailable (Temporarily)
HTTP543 Service Unavailable (Temporarily)
HTTP544 Service Unavailable (Temporarily)
HTTP545 Service Unavailable (Temporarily)
HTTP546 Service Unavailable (Temporarily)
HTTP547 Service Unavailable (Temporarily)
HTTP548 Service Unavailable (Temporarily)
HTTP549 Service Unavailable (Temporarily)
HTTP550 Service Unavailable (Temporarily)
HTTP551 Service Unavailable (Temporarily)
HTTP552 Service Unavailable (Temporarily)
HTTP553 Service Unavailable (Temporarily)
HTTP554 Service Unavailable (Temporarily)
HTTP555 Service Unavailable (Temporarily)
HTTP556 Service Unavailable (Temporarily)
HTTP557 Service Unavailable (Temporarily)
HTTP558 Service Unavailable (Temporarily)
HTTP559 Service Unavailable (Temporarily)
HTTP560 Service Unavailable (Temporarily)
HTTP561 Service Unavailable (Temporarily)
HTTP562 Service Unavailable (Temporarily)
HTTP563 Service Unavailable (Temporarily)
HTTP564 Service Unavailable (Temporarily)
HTTP565 Service Unavailable (Temporarily)
HTTP566 Service Unavailable (Temporarily)
HTTP567 Service Unavailable (Temporarily)
HTTP568 Service Unavailable (Temporarily)
HTTP569 Service Unavailable (Temporarily)
HTTP570 Service Unavailable (Temporarily)
HTTP571 Service Unavailable (Temporarily)
HTTP572 Service Unavailable (Temporarily)
HTTP573 Service Unavailable (Temporarily)
HTTP574 Service Unavailable (Temporarily)
HTTP575 Service Unavailable (Temporarily)
HTTP576 Service Unavailable (Temporarily)
HTTP577 Service Unavailable (Temporarily)
HTTP578 Service Unavailable (Temporarily)
HTTP579 Service Unavailable (Temporarily)
HTTP580 Service Unavailable (Temporarily)
HTTP581 Service Unavailable (Temporarily)
HTTP582 Service Unavailable (Temporarily)
HTTP583 Service Unavailable (Temporarily)
HTTP584 Service Unavailable (Temporarily)
HTTP585 Service Unavailable (Temporarily)
HTTP586 Service Unavailable (Temporarily)
HTTP587 Service Unavailable (Temporarily)
HTTP588 Service Unavailable (Temporarily)
HTTP589 Service Unavailable (Temporarily)
HTTP590 Service Unavailable (Temporarily)
HTTP591 Service Unavailable (Temporarily)
HTTP592 Service Unavailable (Temporarily)
HTTP593 Service Unavailable (Temporarily)
HTTP594 Service Unavailable (Temporarily)
HTTP595 Service Unavailable (Temporarily)
HTTP596 Service Unavailable (Temporarily)
HTTP597 Service Unavailable (Temporarily)
HTTP598 Service Unavailable (Temporarily)
HTTP599 Service Unavailable (Temporarily)
HTTP600 Service Unavailable (Temporarily)
HTTPRetryWith (Error-Classes), 5
HTTPSeeOther (Error-Classes), 5
HTTPServiceUnavailable (Error-Classes), 5
HTTPSSLHandshakeFailed (Error-Classes), 5
HTTPSwitchProtocol (Error-Classes), 5
HTTPSwitchProxy (Error-Classes), 5
HTTPTeaPot (Error-Classes), 5
HTTPTemporaryRedirect (Error-Classes), 5
HTTPTokenExpiredInvalid (Error-Classes), 5
HTTPTooManyRequests (Error-Classes), 5
HTTPUnauthorized (Error-Classes), 5
HTTPUnavailableForLegalReasons (Error-Classes), 5
HTTPUnorderedCollection (Error-Classes), 5
HTTPUnprocessableEntity (Error-Classes), 5
HTTPUnsupportedMediaType (Error-Classes), 5
HTTPUpgradeRequired (Error-Classes), 5
HTTPUseProxy (Error-Classes), 5
HTTPVariantAlsoNegotiates (Error-Classes), 5
HTTPWebServerIsDown (Error-Classes), 5
HTTPWebServerReturnedUnknownError (Error-Classes), 5