Package ‘fauxpas’

March 1, 2018

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  0.2.0

License  MIT + file LICENSE

URL  https://github.com/ropenscilabs/fauxpas

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

VignetteBuilder  knitr

Imports  R6 (>= 2.1.2), httpcode (>= 0.2.0), whisker

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

RoxygenNote  6.0.1

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  2018-03-01 12:54:53 UTC
**R topics documented:**

- fauxpas-package .......................................................... 2
- Error ............................................................................. 2
- Error-Classes ............................................................... 5
- find_error_class ........................................................... 9
- http100 ................................................................. 10

**Index**

<table>
<thead>
<tr>
<th>fauxpas-package</th>
<th>HTTP Error Helpers</th>
</tr>
</thead>
</table>

**Description**

HTTP Error Helpers

**Author(s)**

Scott Chamberlain <myrmecocystus@gmail.com>

**Error**

**Error class**

**Description**

Error class

**Arguments**

- **behavior**
  
  Behavior of the error. 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)
Details

Methods

• do(response, mssg)
  Execute condition, whether it be message, warning, or error.
  – response: is any response from **curl, 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.
  – 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() 

behave parameter options

• stop - stop on error
• warning - warning on error
• message - message on error

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")$get()
  # Not run:
  http414(res)
  # with template
```
http414(res, message_template = "\n\n  --> {{reason}}")
x$do(res)
x$do_verbose(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$do_verbose(res)

## End(Not run)

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

## End(Not run)

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

yy$message_template; zz$message_template
## Not run:

yy$do(res)
zz$do(res)

yy$do_verbose(res)
zz$do_verbose(res)

## End(Not run)

yy <- Error$new(

  message_template = "HTTP({{status}}): {{reason}}\n  {{message}}"
)

yy$message_template
## Not run: yy$do(res)
Error-Classes

<table>
<thead>
<tr>
<th>Error-Classes</th>
<th>Individual error classes</th>
</tr>
</thead>
</table>

**Description**

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

**Usage**

- HTTPContinue
- HTTPSwitchProtocol
- HTTPProcessing
- HTTPOK
- HTTPCreated
- HTTPAccepted
- HTTPNonAuthoritativeInformation
- HTTPNoContent
- HTTPResetContent
- HTTPPartialContent
- HTTPMultiStatus
- HTTPAlreadyReported
- HTTPImUsed
- HTTPMultipleChoices
- HTTPMovedPermanently
- HTTPFound
- HTTPSeeOther
- HTTPNotModified
- HTTPUseProxy
HTTPSwitchProxy
HTTPTemporaryRedirect
HTTPPermanentRedirect
HTTPBadRequest
HTTPUnauthorized
HTTPPaymentRequired
HTTPForbidden
HTTPNotFound
HTTPMethodNotAllowed
HTTPNotAcceptable
HTTPProxyAuthenticationRequired
HTTPRequestTimeout
HTTPConflict
HTTPGone
HTTPLengthRequired
HTTPPreconditionFailed
HTTPRequestEntityTooLarge
HTTPRequestURITooLong
HTTPUnsupportedMediaType
HTTPRequestRangeNotSatisfiable
HTTPExpectationFailed
HTTPTeapot
HTTPAuthenticationTimeout
HTTPMethodFailure
Error-Classes

HTTPUnprocessableEntity
HTTPLocked
HTTPFailedDependency
HTTPUnorderedCollection
HTTPUpgradeRequired
HTTPPreconditionRequired
HTTPTooManyRequests
HTTPRequestHeaderFieldsTooLarge
HTTPLoginTimeout
HTTPNoResponse
HTTPRetryWith
HTTPBlockedByWindowsParentalControls
HTTPUnavailableForLegalReasons
HTTPRequestHeaderValueTooLarge
HTTPCertError
HTTPNoCert
HTTPHTTPToHTTPS
HTTPTokenExpiredInvalid
HTTPClientClosedRequest
HTTPInternalServerError
HTTPNotImplemented
HTTPBadGateway
HTTPServiceUnavailable
HTTPGatewayTimeout
HTTPHTTPVersionNotSupported
HTTPVariantAlsoNegotiates
HTTPInsufficientStorage
HTTPLoopDetected
HTTPBandwidthLimitExceeded
HTTPNotExtended
HTTPNetworkAuthenticationRequired
HTTPNetworkReadTimeoutError
HTTPNetworkConnectTimeoutError

Format
An object of class R6ClassGenerator of length 24.

Details
In addition to what’s available in Error, these classes have a single variable msg 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 http Execute condition, whether it be mes-
    sage, 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
if (requireNamespace("crul")) {

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

## Not run:
```r
dxdo(res)
dxdo_verbose(res)
```

## End(Not run)

```r
# behavior
x <- HTTPRequestURIToolong$new(behavior = "warning")
## Not run:
```
```r
dxdo(res)
dxdo_verbose(res)
```

## End(Not run)

```r
x <- HTTPRequestURIToolong$new(behavior = "message")
## Not run:
```
```r
dxdo(res)
dxdo_verbose(res)
```

## End(Not run)

```r
# with message template
(x <- HTTPRequestURIToolong$new(
message_template = "{{reason}} ........... {{status}}",
message_template_verbose = "{{reason}} .>{{status}}\n{{message}}",
))
```
```r
## Not run:
```
```r
dxdo(res)
dxdo_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 = "stop", message_template)
http101(response, behavior = "stop", message_template)
http102(response, behavior = "stop", message_template)
http200(response, behavior = "stop", message_template)
http201(response, behavior = "stop", message_template)
http202(response, behavior = "stop", message_template)
http203(response, behavior = "stop", message_template)
http204(response, behavior = "stop", message_template)
http205(response, behavior = "stop", message_template)
http206(response, behavior = "stop", message_template)
http100

http207(response, behavior = "stop", message_template)
http208(response, behavior = "stop", message_template)
http226(response, behavior = "stop", message_template)
http300(response, behavior = "stop", message_template)
http301(response, behavior = "stop", message_template)
http302(response, behavior = "stop", message_template)
http303(response, behavior = "stop", message_template)
http304(response, behavior = "stop", message_template)
http305(response, behavior = "stop", message_template)
http306(response, behavior = "stop", message_template)
http307(response, behavior = "stop", message_template)
http308(response, behavior = "stop", message_template)
http400(response, behavior = "stop", message_template)
http401(response, behavior = "stop", message_template)
http402(response, behavior = "stop", message_template)
http403(response, behavior = "stop", message_template)
http404(response, behavior = "stop", message_template)
http405(response, behavior = "stop", message_template)
http406(response, behavior = "stop", message_template)
http407(response, behavior = "stop", message_template)
http408(response, behavior = "stop", message_template)
http409(response, behavior = "stop", message_template)
http410(response, behavior = "stop", message_template)
http411(response, behavior = "stop", message_template)
http412(response, behavior = "stop", message_template)
http413(response, behavior = "stop", message_template)
http414(response, behavior = "stop", message_template)
http415(response, behavior = "stop", message_template)
http416(response, behavior = "stop", message_template)
http417(response, behavior = "stop", message_template)
http418(response, behavior = "stop", message_template)
http419(response, behavior = "stop", message_template)
http420(response, behavior = "stop", message_template)
http422(response, behavior = "stop", message_template)
http423(response, behavior = "stop", message_template)
http424(response, behavior = "stop", message_template)
http425(response, behavior = "stop", message_template)
http426(response, behavior = "stop", message_template)
http428(response, behavior = "stop", message_template)
http429(response, behavior = "stop", message_template)
http431(response, behavior = "stop", message_template)
http440(response, behavior = "stop", message_template)
http444(response, behavior = "stop", message_template)
http449(response, behavior = "stop", message_template)
http450(response, behavior = "stop", message_template)
http451(response, behavior = "stop", message_template)
http494(response, behavior = "stop", message_template)
http495(response, behavior = "stop", message_template)
Arguments

response  The result of a call via `curl`, `curl`, or `httr`
behavior  Behavior of the error. 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.
behavior parameter options

- stop - stop on error
- warning - warning on error
- message - message on error

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")
}
**Index**

*Topic datasets
  Error, 2
  Error-Classes, 5

*Topic package
  fauxpas-package, 2

Error, 2, 5, 8, 10, 14
Error-Classes, 5

fauxpas (fauxpas-package), 2
fauxpas-package, 2
find_error_class, 9

http, 3, 8
http (http100), 10
http100, 10
http101 (http100), 10
http102 (http100), 10
http200 (http100), 10
http201 (http100), 10
http202 (http100), 10
http203 (http100), 10
http204 (http100), 10
http205 (http100), 10
http206 (http100), 10
http207 (http100), 10
http208 (http100), 10
http226 (http100), 10
http300 (http100), 10
http301 (http100), 10
http302 (http100), 10
http303 (http100), 10
http304 (http100), 10
http305 (http100), 10
http306 (http100), 10
http307 (http100), 10
http308 (http100), 10
http400 (http100), 10
http401 (http100), 10
http402 (http100), 10
http403 (http100), 10
http404 (http100), 10
http405 (http100), 10
http406 (http100), 10
http407 (http100), 10
http408 (http100), 10
http409 (http100), 10
http410 (http100), 10
http411 (http100), 10
http412 (http100), 10
http413 (http100), 10
http414 (http100), 10
http415 (http100), 10
http416 (http100), 10
http417 (http100), 10
http418 (http100), 10
http419 (http100), 10
http420 (http100), 10
http421 (http100), 10
http422 (http100), 10
http423 (http100), 10
http424 (http100), 10
http425 (http100), 10
http426 (http100), 10
http427 (http100), 10
http428 (http100), 10
http429 (http100), 10
http430 (http100), 10
http431 (http100), 10
http432 (http100), 10
http433 (http100), 10
http434 (http100), 10
http435 (http100), 10
http436 (http100), 10
http437 (http100), 10
http438 (http100), 10
http439 (http100), 10
http440 (http100), 10
http441 (http100), 10
http442 (http100), 10
http443 (http100), 10
http444 (http100), 10
http445 (http100), 10
http446 (http100), 10
http447 (http100), 10
http448 (http100), 10
http449 (http100), 10
http450 (http100), 10
http451 (http100), 10
http452 (http100), 10
http453 (http100), 10
http454 (http100), 10
http455 (http100), 10
http456 (http100), 10
http457 (http100), 10
http458 (http100), 10
http459 (http100), 10
http460 (http100), 10
http461 (http100), 10
http462 (http100), 10
http463 (http100), 10
http464 (http100), 10
http465 (http100), 10
http466 (http100), 10
http467 (http100), 10
http468 (http100), 10
http469 (http100), 10
http470 (http100), 10
http471 (http100), 10
http472 (http100), 10
http473 (http100), 10
http474 (http100), 10
http475 (http100), 10
http476 (http100), 10
http477 (http100), 10
http478 (http100), 10
http479 (http100), 10
http480 (http100), 10
http481 (http100), 10
http482 (http100), 10
http483 (http100), 10
http484 (http100), 10
http485 (http100), 10
http486 (http100), 10
http487 (http100), 10
http488 (http100), 10
http489 (http100), 10
http490 (http100), 10
http491 (http100), 10
http492 (http100), 10
http493 (http100), 10
http494 (http100), 10
http495 (http100), 10
http496 (http100), 10
http497 (http100), 10
http498 (http100), 10
http499 (http100), 10
http500 (http100), 10
http501 (http100), 10
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