

Package ‘webutils’

April 28, 2020

Type Package

Title Utility Functions for Developing Web Applications

Version 1.1

Author Jeroen Ooms

Maintainer Jeroen Ooms <jeroen@berkeley.edu>

Description Parses http request data in application/json, multipart/form-data, or application/x-www-form-urlencoded format. Includes example of hosting and parsing html form data in R using either 'httpuv' or 'Rhttpd'.

License MIT + file LICENSE

URL <https://github.com/jeroen/webutils>

BugReports <https://github.com/jeroen/webutils/issues>

Imports curl (>= 2.5), jsonlite

Suggests httpuv, testthat

RoxygenNote 7.0.2

Language en-US

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-04-28 21:00:02 UTC

R topics documented:

demo_httpuv	2
demo_rhttpd	2
parse_http	3
parse_multipart	3
parse_query	4

Index	5
--------------	----------

demo_httputv

Demo multipart parser with httputv

Description

Starts the httputv web server and hosts a simple form including a file upload to demo the multipart parser.

Usage

```
demo_httputv(port = 9359)
```

Arguments

port which port number to run the http server

See Also

Other demo: [demo_rhttpd\(\)](#)

demo_rhttpd

Demo multipart parser with rhttpd

Description

Starts the Rhttpd web server and hosts a simple form including a file upload to demo the multipart parser.

Usage

```
demo_rhttpd()
```

See Also

Other demo: [demo_httputv\(\)](#)

parse_http	<i>Parse http request</i>
------------	---------------------------

Description

Parse the body of a http request, based on the Content-Type request header. Currently supports the three most important content types: application/x-www-form-urlencoded ([parse_query](#)), multipart/form-data ([parse_multipart](#)) and application/json ([fromJSON](#)).

Usage

```
parse_http(body, content_type, ...)
```

Arguments

body	request body of the http request
content_type	content-type http request header as specified by the client
...	additional arguments passed to parser function

Examples

```
# Parse json encoded payload:
parse_http('{"foo":123, "bar":true}', 'application/json')

# Parse url-encoded payload
parse_http("foo=1%2B1%3D2&bar=yin%26yang", "application/x-www-form-urlencoded")

## Not run: use demo app to parse multipart/form-data payload
demo_rhttpd()

## End(Not run)
```

parse_multipart	<i>Parse a multipart/form-data request</i>
-----------------	--

Description

Parse a multipart/form-data request, which is usually generated from a HTML form submission. The parameters can include both text values as well as binary files. They can be distinguished from the presence of a filename attribute.

Usage

```
parse_multipart(body, boundary)
```

Arguments

body body of the HTTP request. Must be raw or character vector.
 boundary boundary string as specified in the Content-Type request header.

Details

A multipart/form-data request consists of a single body which contains one or more values plus meta-data, separated using a boundary string. This boundary string is chosen by the client (e.g. the browser) and specified in the Content-Type header of the HTTP request. There is no escaping; it is up to the client to choose a boundary string that does not appear in one of the values.

The parser is written in pure R, but still pretty fast because it uses the regex engine.

Examples

```
## Not run: example form
demo_rhttpd()

## End(Not run)
```

parse_query	<i>Parse query string</i>
-------------	---------------------------

Description

Parse http parameters from a query string. This includes unescaping of url-encoded values.

Usage

```
parse_query(query)
```

Arguments

query a url-encoded query string

Details

For http GET requests, the query string is specified in the URL after the question mark. For http POST or PUT requests, the query string can be used in the request body when the Content-Type header is set to application/x-www-form-urlencoded.

Examples

```
q <- "foo=1%2B1%3D2&bar=yin%26yang"
parse_query(q)
```

Index

`demo_httputv`, [2](#), [2](#)

`demo_rhttpd`, [2](#), [2](#)

`fromJSON`, [3](#)

`parse_http`, [3](#)

`parse_multipart`, [3](#), [3](#)

`parse_query`, [3](#), [4](#)