Rocky Enterprise Linux 9.2 Manual Pages on command 'HTTP::Headers::Util.3pm'

C:\>man HTTP::Headers::Util.3pm

HTTP::Headers::Util(3pm) User Contributed Perl Documentation HTTP::Headers::Util(3pm)

NAME

HTTP::Headers::Util - Header value parsing utility functions

VERSION

version 6.22

SYNOPSIS

```
use HTTP::Headers::Util qw(split_header_words);
@values = split_header_words($h->header("Content-Type"));
```

DESCRIPTION

This module provides a few functions that helps parsing and construction of valid HTTP header values. None of the functions are exported by default.

The following functions are available:

split_header_words(@header_values)

This function will parse the header values given as argument into a list of anonymous arrays containing key/value pairs. The function knows how to deal with ",", ";" and "=" as well as quoted values after "=". A list of space

separated tokens are parsed as if they were separated by ";".

If the @header_values passed as argument contains multiple values, then they are treated as if they were a single value separated by comma ",".

This means that this function is useful for parsing header fields that follow this syntax (BNF as from the HTTP/1.1 specification, but we relax the requirement for tokens).

```
headers          = #header
header           = (token | parameter) *( [";"] (token | parameter))

token            = 1*<any CHAR except CTLs or separators>
separators       = "(" | ")" | "<" | ">" | "@"
                 | "," | ";" | ":" | "\" | <">
                 | "/" | "[" | "]" | "?" | "="
                 | "{" | "}" | SP | HT

quoted-string    = ( <"> *(qdtext | quoted-pair ) <"> )
qdtext           = <any TEXT except <">>
quoted-pair      = "\" CHAR

parameter        = attribute "=" value
attribute        = token
value            = token | quoted-string
```

Each header is represented by an anonymous array of key/value pairs. The keys will be all be forced to lower case. The value for a simple token (not part of a parameter) is "undef". Syntactically incorrect headers will not necessarily be parsed as you would want.

This is easier to describe with some examples:

```
split_header_words('foo="bar"; port="80,81"; DISCARD, BAR=baz');
split_header_words('text/html; charset="iso-8859-1"');
split_header_words('Basic realm="\\"foo\\\\bar\\""');
```

will return

```
[foo=>'bar', port=>'80,81', discard=> undef], [bar=>'baz' ]
['text/html' => undef, charset => 'iso-8859-1']
[basic => undef, realm => "\"foo\\bar\""]
```

If you don't want the function to convert tokens and attribute keys to lower case you can call it as "_split_header_words" instead (with a leading underscore).

join_header_words(@arrays)

This will do the opposite of the conversion done by split_header_words(). It takes a list of anonymous arrays as arguments (or a list of key/value pairs) and produces a single header value. Attribute values are quoted if needed.

Example:

```
join_header_words(["text/plain" => undef, charset => "iso-8859/1"]);
join_header_words("text/plain" => undef, charset => "iso-8859/1");
```

will both return the string:

```
text/plain; charset="iso-8859/1"
```

AUTHOR

Gisle Aas <gisle@activestate.com>

COPYRIGHT AND LICENSE

This is free software; you can redistribute it and/or modify it under the same terms as the Perl 5 programming language system itself.