NAME

Mail::Header − manipulate MIME headers

SYNOPSIS

```
use Mail::Header;

my $head = Mail::Header->new;
my $head = Mail::Header->new( \*STDIN );
my $head = Mail::Header->new( [<>], Modify => 0);
```

DESCRIPTION

Read, write, create, and manipulate MIME headers, the leading part of each modern e−mail message, but also used in other protocols like HTTP. The fields are kept in Mail::Field objects.

Be aware that the header fields each have a name part, which shall be treated case-insensitive, and a content part, which may be folded over multiple lines.

Mail::Header does not always follow the RFCs strict enough, does not help you with character encodings. It does not use weak references where it could (because those did not exist when the module was written) which costs some performance and make the implementation a little more complicated. The Mail::Message::Head implementation is much newer and therefore better.

METHODS

Constructors

`$obj->`**dup**()

Create a duplicate of the current object.

`$obj->`**new**([$source], [%options])

Mail::Header**->new**([$source], [%options])

The `$source` may be either a file descriptor (reference to a GLOB) or a reference to an array. If given the new object will be initialized with headers either from the array of read from the file descriptor.

`%options` is a list of options given in the form of key-value pairs, just like a hash table. Valid options are

```
-Option     --Default
 FoldLength  79
 MailFrom    'KEEP'
 Modify      false
```

FoldLength => INTEGER

The default length of line to be used when folding header lines. See **fold_length()**.

MailFrom => 'IGNORE'|'COERCE'|'KEEP'|'ERROR'

See method **mail_from()**.

Modify => BOOLEAN

If this value is *true* then the headers will be re-formatted, otherwise the format of the header lines will remain unchanged.

"Fake" constructors

Be warned that the next constructors all require an already created header object, of which the original content will be destroyed.

`$obj->`**empty**()

Empty an existing `Mail::Header` object of all lines.

`$obj->`**extract**(ARRAY)

Extract a header from the given array into an existing Mail::Header object. `extract` **will modify** this array. Returns the object that the method was called on.

`$obj->`**header**([ARRAY])
: `header` does multiple operations. First it will extract a header from the ARRAY, if given. It will then reformat the header (if reformatting is permitted), and finally return a reference to an array which contains the header in a printable form.

`$obj->`**header_hashref**([HASH])
: As **header()**, but it will eventually set headers from a hash reference, and it will return the headers as a hash reference.

  example:

```
$fields->{From} = 'Tobias Brox <tobix@cpan.org>';
$fields->{To}   = ['you@somewhere', 'me@localhost'];
$head->header_hashref($fields);
```

`$obj->`**read**($fh)
: Read a header from the given file descriptor into an existing Mail::Header object.

Accessors

`$obj->`**fold_length**([$tag], [$length])
: Set the default fold length for all tags or just one. With no arguments the default fold length is returned. With two arguments it sets the fold length for the given tag and returns the previous value. If only `$length` is given it sets the default fold length for the current object.

  In the two argument form `fold_length` may be called as a static method, setting default fold lengths for tags that will be used by **all** `Mail::Header` objects. See the `fold` method for a description on how `Mail::Header` uses these values.

`$obj->`**mail_from**('IGNORE'|'COERCE'|'KEEP'|'ERROR')
: This specifies what to do when a `` `From ' `` line is encountered. Valid values are `IGNORE` – ignore and discard the header, `ERROR` – invoke an error (call die), `COERCE` – rename them as Mail-From and `KEEP` – keep them.

`$obj->`**modify**([$value])
: If `$value` is *false* then `Mail::Header` will not do any automatic reformatting of the headers, other than to ensure that the line starts with the tags given.

Processing

`$obj->`**add**(`$tag`, `$line` [, `$index`])
: Add a new line to the header. If `$tag` is `undef` the tag will be extracted from the beginning of the given line. If `$index` is given, the new line will be inserted into the header at the given point, otherwise the new line will be appended to the end of the header.

`$obj->`**as_string**()
: Returns the header as a single string.

`$obj->`**cleanup**()
: Remove any header line that, other than the tag, only contains whitespace

`$obj->`**combine**(`$tag` [, `$with`])
: Combine all instances of `$tag` into one. The lines will be joined together `$with`, or a single space if not given. The new item will be positioned in the header where the first instance was, all other instances of `$tag` will be removed.

`$obj->`**count**($tag)
: Returns the number of times the given atg appears in the header

`$obj->`**delete**(`$tag` [, `$index`])
: Delete a tag from the header. If an `$index` id is given, then the Nth instance of the tag will be removed. If no `$index` is given, then all instances of tag will be removed.

`$obj->`**fold**([$length])
Fold the header. If `$length` is not given, then `Mail::Header` uses the following rules to determine what length to fold a line.

`$obj->`**get**(`$tag` [, `$index`])
Get the text from a line. If an `$index` is given, then the text of the Nth instance will be returned. If it is not given the return value depends on the context in which `get` was called. In an array context a list of all the text from all the instances of the `$tag` will be returned. In a scalar context the text for the first instance will be returned.

The lines are unfolded, but still terminated with a new-line (see `chomp`)

`$obj->`**print**([$fh])
Print the header to the given file descriptor, or `STDOUT` if no file descriptor is given.

`$obj->`**replace**(`$tag`, `$line` [, `$index`])
Replace a line in the header. If `$tag` is `undef` the tag will be extracted from the beginning of the given line. If `$index` is given the new line will replace the Nth instance of that tag, otherwise the first instance of the tag is replaced. If the tag does not appear in the header then a new line will be appended to the header.

`$obj->`**tags**()
Returns an array of all the tags that exist in the header. Each tag will only appear in the list once. The order of the tags is not specified.

`$obj->`**unfold**([$tag])
Unfold all instances of the given tag so that they do not spread across multiple lines. If `$tag` is not given then all lines are unfolded.

The unfolding process is wrong but (for compatibility reasons) will not be repaired: only one blank at the start of the line should be removed, not all of them.

SEE ALSO

This module is part of the MailTools distribution, *http://perl.overmeer.net/mailtools/*.

AUTHORS

The MailTools bundle was developed by Graham Barr. Later, Mark Overmeer took over maintenance without commitment to further development.

Mail::Cap by Gisle Aas <aas@oslonett.no>. Mail::Field::AddrList by Peter Orbaek <poe@cit.dk>. Mail::Mailer and Mail::Send by Tim Bunce <Tim.Bunce@ig.co.uk>. For other contributors see ChangeLog.

LICENSE