NAME

Mail::Message::Body::Construct − adds functionality to Mail::Message::Body

SYNOPSIS

DESCRIPTION

This package adds complex functionality to the Mail::Message::Body class. This functions less often used, so many programs will not compile this package.

METHODS

Constructing a body

`$obj`−>**attach**($messages, `%options`)

Make a multipart containing this body and the specified `$messages`. The options are passed to the constructor of the multi-part body. If you need more control, create the multi-part body yourself. At least take a look at Mail::Message::Body::Multipart.

The message-parts will be coerced into a Mail::Message::Part, so you may attach Mail::Internet or MIME::Entity objects if you want −−see **Mail::Message::coerce()**. A new body with attached messages is returned.

example:

```
my $pgpkey = Mail::Message::Body::File->new(file => 'a.pgp');
my $msg    = Mail::Message->buildFromBody(
   $message->decoded->attach($pgpkey));

# The last message of the $multi multiparted body becomes a coerced $entity.
my $entity  = MIME::Entity->new;
my $multi   = $msg->body->attach($entity);

# Now create a new message
my $msg     = Mail::Message->new(head => ..., body => $multi);
```

`$obj`−>**concatenate**($components)

Concatenate a list of elements into one new body.

Specify a list of text `$components`. Each component can be a message (Mail::Message, the body of the message is used), a plain body (Mail::Message::Body), `undef` (which will be skipped), a scalar (which is split into lines), or an array of scalars (each providing one line).

example:

```
# all arguments are Mail::Message::Body's.
my $sum = $body->concatenate($preamble, $body, $epilogue, "-- \n" , $sig);
```

`$obj`−>**foreachLine**(CODE)

Create a new body by performing an action on each of its lines. If none of the lines change, the current body will be returned, otherwise a new body is created of the same type as the current.

The CODE refers to a subroutine which is called, where `$_` contains body's original line. DO NOT CHANGE `$_`!!! The result of the routine is taken as new line. When the routine returns `undef`, the line will be skipped.

example:

```
my $content  = $msg->decoded;
my $reply    = $content->foreachLine( sub { '> '.$_ } );
my $rev      = $content->foreachLine( sub {reverse} );

sub filled() { length $_ > 1 ? $_ : undef }
my $nonempty = $content->foreachLine( \&filled );
```

```
 my $wrong    = $content->foreachLine( sub {s/a/A/} );  # WRONG!!!
 my $right    = $content->foreachLine(
        sub {(my $x=$_) =~ s/a/A/; $x} );
```

$obj->**stripSignature**(%options)

Strip the signature from the body. The body must already be decoded otherwise the wrong lines may get stripped. Returned is the stripped version body, and in list context also the signature, encapsulated in its own body object. The signature separator is the first line of the returned signature body.

The signature is added by the sender to tell about him- or herself. It is superfluous in some situations, for instance if you want to create a reply to the person's message you do not need to include that signature.

If the body had no signature, the original body object is returned, and `undef` for the signature body.

```
 -Option     --Default
  max_lines    10
  pattern      qr/^--\s?$/
  result_type  <same as current>
```

max_lines => INTEGER|undef

The maximum number of lines which can be the length of a signature. Specify `undef` to remove the limit.

pattern => REGEX|STRING|CODE

Which pattern defines the line which indicates the separator between the message and the signature. In case of a STRING, this is matched to the beginning of the line, and REGEX is a full regular expression.

In case of CODE, each line (from last to front) is passed to the specified subroutine as first argument. The subroutine must return TRUE when the separator is found.

result_type => CLASS

The type of body to be created for the stripped body (and maybe also to contain the stripped signature)

example:

```
 my $start = $message->decoded;
 my $start = $body->decoded;

 my $stripped = $start->stripSignature;

 my ($stripped, $sign) = $start->stripSignature
     (max_lines => 5, pattern => '-*-*-');
```

SEE ALSO

This module is part of Mail-Message distribution version 3.012, built on February 11, 2022. Website: *http://perl.overmeer.net/CPAN/*

LICENSE