NAME

Mail::Message::Body::Multipart − body of a message with attachments

INHERITANCE

```
 Mail::Message::Body::Multipart
   is a Mail::Message::Body
   is a Mail::Reporter
```

SYNOPSIS

```
 See Mail::Message::Body

 if($body->isMultipart) {
    my @attachments = $body->parts;
    my $attachment3 = $body->part(2);
    my $before      = $body->preamble;
    my $after       = $body->epilogue;
    $body->part(1)->delete;
 }
```

DESCRIPTION

The body (content) of a message can be stored in various ways. In this manual-page you find the description of extra functionality you have when a message contains attachments (parts).

Extends "DESCRIPTION" in Mail::Message::Body.

OVERLOADED

Extends "OVERLOADED" in Mail::Message::Body.

overload: **""**
Inherited, see "OVERLOADED" in Mail::Message::Body

overload: **'==' and '!='**
Inherited, see "OVERLOADED" in Mail::Message::Body

overload: **@{}**
Inherited, see "OVERLOADED" in Mail::Message::Body

overload: **bool**
Inherited, see "OVERLOADED" in Mail::Message::Body

METHODS

Extends "METHODS" in Mail::Message::Body.

Constructors

Extends "Constructors" in Mail::Message::Body.

`$obj->`**clone**()
Inherited, see "Constructors" in Mail::Message::Body

Mail::Message::Body::Multipart−>**new**(%options)

```
 −Option           −−Defined in         −−Default
  based_on           Mail::Message::Body  undef
  boundary                                undef
  charset            Mail::Message::Body  'PERL' or <undef>
  checked            Mail::Message::Body  <false>
  content_id         Mail::Message::Body  undef
  data               Mail::Message::Body  undef
  description        Mail::Message::Body  undef
  disposition        Mail::Message::Body  undef
  eol                Mail::Message::Body  'NATIVE'
  epilogue                                ''
  file               Mail::Message::Body  undef
  filename           Mail::Message::Body  undef
  log                Mail::Reporter       'WARNINGS'
  message            Mail::Message::Body  undef
  mime_type          Mail::Message::Body  'multipart/mixed'
  modified           Mail::Message::Body  <false>
  parts                                   undef
  preamble                                undef
  trace              Mail::Reporter       'WARNINGS'
  transfer_encoding  Mail::Message::Body  'none'
```

based_on => BODY
boundary => STRING
: Separator to be used between parts of the message. This separator must be unique in case the message contains nested multiparts (which are not unusual). If `undef`, a nice unique boundary will be generated.

charset => CHARSET|'PERL'
checked => BOOLEAN
content_id => STRING
data => ARRAY-OF-LINES | STRING
description => STRING|FIELD
disposition => STRING|FIELD
eol => 'CR'|'LF'|'CRLF'|'NATIVE'
epilogue => BODY|STRING
: The text which is included in the main body after the final boundary. This is usually empty, and has no meaning.

: Provide a BODY object or a STRING which will automatically translated into a `text/plain` body.

file => FILENAME|FILEHANDLE|IOHANDLE
filename => FILENAME
log => LEVEL
message => MESSAGE
mime_type => STRING|FIELD|MIME
modified => BOOLEAN
parts => ARRAY−OF−(MESSAGES|BODIES)
: Specifies an initial list of parts in this body. These may be full MESSAGES, or BODIES which transformed into messages before use. Each message is coerced into a Mail::Message::Part object.

: MIME::Entity and Mail::Internet objects are acceptable in the list, because they are coercible into Mail::Message::Part's. Values of `undef` will be skipped silently.

preamble => BODY|STRING
: The text which is included in the body before the first part. It is common use to include a text to warn the user that the message is a multipart. However, this was useful in earlier days: most mail agents are very capable in warning the user themselves.

: Provide a BODY object or a STRING which will automatically translated into a text/plain body.

trace => LEVEL
transfer_encoding => STRING|FIELD

example:

```
 my $intro = Mail::Message::Body->new(data => ['part one']);
 my $pgp   = Mail::Message::Body->new(data => ['part three']);

 my $body  = Mail::Message::Body::Multipart->new
   ( boundary => time . '--it-s-mine'
   , preamble => "This is a multi-part message in MIME format.\n\n"
```

```
      , parts    => [ $intro, $folder->message(3)->decoded, $pgp ]
      );
```

Constructing a body

Extends “Constructing a body” in Mail::Message::Body.

`$obj->`**attach**($messages|$bodies)

Attach a list of `$messages` to this multipart. A new body is returned. When you specify `$bodies`, they will first be translated into real messages. MIME::Entity and Mail::Internet objects may be specified too. In any case, the parts will be coerced into Mail::Message::Part’s.

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

Inherited, see “Constructing a body” in Mail::Message::Body::Encode

`$obj->`**concatenate**($components)

Inherited, see “Constructing a body” in Mail::Message::Body::Construct

`$obj->`**decoded**(%options)

Inherited, see “Constructing a body” in Mail::Message::Body

`$obj->`**encode**(%options)

Inherited, see “Constructing a body” in Mail::Message::Body::Encode

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

Inherited, see “Constructing a body” in Mail::Message::Body::Encode

`$obj->`**eol**([’CR’|’LF’|’CRLF’|’NATIVE’])

Inherited, see “Constructing a body” in Mail::Message::Body

`$obj->`**foreachComponent**(CODE)

Execute the CODE for each component of the message: the preamble, the epilogue, and each of the parts.

Each component is a body and is passed as second argument to the CODE. The first argument is a reference to this multi-parted body. The CODE returns a body object. When any of the returned bodies differs from the body which was passed, then a new multi-part body will be returned. Reference to the not-changed bodies and the changed bodies will be included in that new multi-part.

example:

```
 my $checked = $multi->foreachComponent(sub {$_[1]->check});
```

`$obj->`**foreachLine**((CODE))

It is NOT possible to call some code for each line of a multipart, because that would not only inflict damage to the body of each message part, but also to the headers and the part separators.

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

Removes all parts which contains data usually defined as being signature. The MIME::Type module provides this knowledge. A new multipart is returned, containing the remaining parts. No `%options` are defined yet, although some may be specified, because this method overrules the `stripSignature` method for normal bodies.

```
 -Option     --Defined in                    --Default
  max_lines    Mail::Message::Body::Construct  10
  pattern      Mail::Message::Body::Construct  qr/^--\s?$/
  result_type  Mail::Message::Body::Construct  <same as current>
```

max_lines => INTEGER|undef
pattern => REGEX|STRING|CODE
result_type => CLASS

`$obj->`**unify**($body)

Inherited, see “Constructing a body” in Mail::Message::Body::Encode

The body

Extends "The body" in Mail::Message::Body.

`$obj->`**isDelayed**()
: Inherited, see "The body" in Mail::Message::Body

`$obj->`**isMultipart**()
: Inherited, see "The body" in Mail::Message::Body

`$obj->`**isNested**()
: Inherited, see "The body" in Mail::Message::Body

`$obj->`**message**([$message])
: Inherited, see "The body" in Mail::Message::Body

`$obj->`**partNumberOf**($part)
: Inherited, see "The body" in Mail::Message::Body

About the payload

Extends "About the payload" in Mail::Message::Body.

`$obj->`**charset**()
: Inherited, see "About the payload" in Mail::Message::Body

`$obj->`**checked**([BOOLEAN])
: Inherited, see "About the payload" in Mail::Message::Body

`$obj->`**contentId**([STRING|$field])
: Inherited, see "About the payload" in Mail::Message::Body

`$obj->`**description**([STRING|$field])
: Inherited, see "About the payload" in Mail::Message::Body

`$obj->`**disposition**([STRING|$field])
: Inherited, see "About the payload" in Mail::Message::Body

`$obj->`**dispositionFilename**([$directory])
: Inherited, see "About the payload" in Mail::Message::Body::Encode

`$obj->`**isBinary**()
: Inherited, see "About the payload" in Mail::Message::Body::Encode

`$obj->`**isText**()
: Inherited, see "About the payload" in Mail::Message::Body::Encode

`$obj->`**mimeType**()
: Inherited, see "About the payload" in Mail::Message::Body

`$obj->`**nrLines**()
: Inherited, see "About the payload" in Mail::Message::Body

`$obj->`**size**()
: Inherited, see "About the payload" in Mail::Message::Body

`$obj->`**transferEncoding**([STRING|$field])
: Inherited, see "About the payload" in Mail::Message::Body

`$obj->`**type**([STRING|$field])
: Inherited, see "About the payload" in Mail::Message::Body

Access to the payload

Extends "Access to the payload" in Mail::Message::Body.

`$obj->`**boundary**([STRING])
: Returns the boundary which is used to separate the parts in this body. If none was read from file, then one will be assigned. With STRING you explicitly set the boundary to be used.

$obj->**endsOnNewline**()
Inherited, see "Access to the payload" in Mail::Message::Body

$obj->**epilogue**()
Returns the epilogue; the text after the last message part (after the last real attachment). The epilogue is stored in a BODY object, and its encoding is taken from the general multipart header.

$obj->**file**()
Inherited, see "Access to the payload" in Mail::Message::Body

$obj->**lines**()
Inherited, see "Access to the payload" in Mail::Message::Body

$obj->**part**($index)
Returns only the part with the specified `$index`. You may use a negative value here, which counts from the back in the list. Parts which are flagged to be deleted are included in the count.

example:

```
 $message->body->part(2)->print;
 $body->part(1)->delete;
```

$obj->**parts**([<'ALL'|'ACTIVE'|'DELETED'|'RECURSE'|$filter>])
Return all parts by default, or when ALL is specified. `ACTIVE` returns the parts which are not flagged for deletion, as opposite to `DELETED`. `RECURSE` descents into all nested multiparts to collect all parts.

You may also specify a code reference which is called for each nested part. The first argument will be the message part. When the code returns true, the part is incorporated in the return list.

example:

```
 print "Number of attachments: ",
     scalar $message->body->parts('ACTIVE');

 foreach my $part ($message->body->parts) {
     print "Type: ", $part->get('Content-Type');
 }
```

$obj->**preamble**()
Returns the preamble; the text before the first message part (before the first real attachment). The preamble is stored in a BODY object, and its encoding is taken from the multipart header.

$obj->**print**([$fh])
Inherited, see "Access to the payload" in Mail::Message::Body

$obj->**printEscapedFrom**($fh)
Inherited, see "Access to the payload" in Mail::Message::Body

$obj->**string**()
Inherited, see "Access to the payload" in Mail::Message::Body

$obj->**stripTrailingNewline**()
Inherited, see "Access to the payload" in Mail::Message::Body

$obj->**write**(%options)
Inherited, see "Access to the payload" in Mail::Message::Body

Internals

Extends "Internals" in Mail::Message::Body.

$obj->**addTransferEncHandler**(`$name`, <$class|$object>)
Mail::Message::Body::Multipart->**addTransferEncHandler**(`$name`, <$class|$object>)
Inherited, see "Internals" in Mail::Message::Body::Encode

`$obj`−>**contentInfoFrom**($head)
Inherited, see ‘‘Internals’’ in Mail::Message::Body

`$obj`−>**contentInfoTo**($head)
Inherited, see ‘‘Internals’’ in Mail::Message::Body

`$obj`−>**fileLocation**([$begin, `$end`])
Inherited, see ‘‘Internals’’ in Mail::Message::Body

`$obj`−>**getTransferEncHandler**($type)
Inherited, see ‘‘Internals’’ in Mail::Message::Body::Encode

`$obj`−>**isModified**()
Inherited, see ‘‘Internals’’ in Mail::Message::Body

`$obj`−>**load**()
Inherited, see ‘‘Internals’’ in Mail::Message::Body

`$obj`−>**modified**([BOOLEAN])
Inherited, see ‘‘Internals’’ in Mail::Message::Body

`$obj`−>**moveLocation**([$distance])
Inherited, see ‘‘Internals’’ in Mail::Message::Body

`$obj`−>**read**(`$parser`, `$head`, `$bodytype`, [$chars, [$lines]])
Inherited, see ‘‘Internals’’ in Mail::Message::Body

Error handling

Extends ‘‘Error handling’’ in Mail::Message::Body.

`$obj`−>**AUTOLOAD**()
Inherited, see ‘‘Error handling’’ in Mail::Message::Body

`$obj`−>**addReport**($object)
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**defaultTrace**([$level]|[$loglevel, `$tracelevel`]|[$level, `$callback`])
Mail::Message::Body::Multipart−>**defaultTrace**([$level]|[$loglevel, `$tracelevel`]|[$level, `$callback`])
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**errors**()
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**log**([$level, [$strings]])
Mail::Message::Body::Multipart−>**log**([$level, [$strings]])
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**logPriority**($level)
Mail::Message::Body::Multipart−>**logPriority**($level)
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**logSettings**()
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**notImplemented**()
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**report**([$level])
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**reportAll**([$level])
Inherited, see ‘‘Error handling’’ in Mail::Reporter

`$obj`−>**trace**([$level])
: Inherited, see ''Error handling'' in Mail::Reporter

`$obj`−>**warnings**()
: Inherited, see ''Error handling'' in Mail::Reporter

Cleanup

Extends ''Cleanup'' in Mail::Message::Body.

`$obj`−>**DESTROY**()
: Inherited, see ''Cleanup'' in Mail::Reporter

DETAILS

Extends ''DETAILS'' in Mail::Message::Body.

DIAGNOSTICS

Warning: Charset `$name` is not known
: The encoding or decoding of a message body encounters a character set which is not understood by Perl's Encode module.

Error: Data not convertible to a message (type is `$type`)
: An object which is not coercable into a Mail::Message::Part object was passed to the initiation. The data is ignored.

Warning: No decoder defined for transfer encoding `$name`.
: The data (message body) is encoded in a way which is not currently understood, therefore no decoding (or recoding) can take place.

Warning: No encoder defined for transfer encoding `$name`.
: The data (message body) has been decoded, but the required encoding is unknown. The decoded data is returned.

Error: Package `$package` does not implement `$method`.
: Fatal error: the specific package (or one of its superclasses) does not implement this method where it should. This message means that some other related classes do implement this method however the class at hand does not. Probably you should investigate this and probably inform the author of the package.

Error: Unknown criterium `$what` to select parts.
: Valid choices fdr part selections are `ALL`, `ACTIVE`, `DELETED`, `RECURSE` or a code reference. However, some other argument was passed.

Warning: Unknown line terminator `$eol` ignored

Error: You cannot use foreachLine on a multipart
: **foreachLine()** should be used on decoded message bodies only, because it would attempt to modify part-headers and separators as well, which is clearly not acceptable.

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