NAME

Mail::Reporter − base−class and error reporter for Mail::Box

INHERITANCE

```
 Mail::Reporter is extended by
   Mail::Box
   Mail::Box::Collection
   Mail::Box::Identity
   Mail::Box::Locker
   Mail::Box::MH::Index
   Mail::Box::MH::Labels
   Mail::Box::Manager
   Mail::Box::Parser
   Mail::Box::Search
   Mail::Box::Thread::Manager
   Mail::Box::Thread::Node
   Mail::Message
   Mail::Message::Body
   Mail::Message::Body::Delayed
   Mail::Message::Convert
   Mail::Message::Field
   Mail::Message::Field::Attribute
   Mail::Message::Head
   Mail::Message::Head::FieldGroup
   Mail::Message::TransferEnc
   Mail::Server
   Mail::Transport
```

SYNOPSIS

```
 $folder−>log(WARNING => 'go away');
 print $folder−>trace;        # current level
 $folder−>trace('PROGRESS');  # set level
 print $folder−>errors;
 print $folder−>report('PROGRESS');
```

DESCRIPTION

The `Mail::Reporter` class is the base class for all classes, except Mail::Message::Field::Fast because it would become slow... This base class is used during initiation of the objects, and for configuring and logging error messages.

METHODS

The `Mail::Reporter` class is the base for nearly all other objects. It can store and report problems, and contains the general constructor **new()**.

Constructors

Mail::Reporter−>**new**(%options)

This error container is also the base constructor for all modules, (as long as there is no need for another base object) The constructor always accepts the following `%options` related to error reports.

```
 −Option−−Default
  log     'WARNINGS'
  trace   'WARNINGS'
```

log => LEVEL

Log messages which have a priority higher or equal to the specified level are stored internally and can be retrieved later. The global default for this option can be changed with **defaultTrace()**.

Known levels are `INTERNAL`, `ERRORS`, `WARNINGS`, `PROGRESS`, `NOTICES` `DEBUG`, and `NONE`. The `PROGRESS` level relates to the reading and writing of folders. `NONE` will cause only

`INTERNAL` errors to be logged. By the way: `ERROR` is an alias for `ERRORS`, as `WARNING` is an alias for `WARNINGS`, and `NOTICE` for `NOTICES`.

trace => LEVEL
: Trace messages which have a level higher or equal to the specified level are directly printed using warn. The global default for this option can be changed with **defaultTrace()**.

Error handling

`$obj`−>**AUTOLOAD**()
: By default, produce a nice warning if the sub-classes cannot resolve a method.

`$obj`−>**addReport**($object)
: Add the report from other `$object` to the report of this object. This is useful when complex actions use temporary objects which are not returned to the main application but where the main application would like to know about any problems.

`$obj`−>**defaultTrace**([$level]|[$loglevel, `$tracelevel`]|[$level, `$callback`])
Mail::Reporter−>**defaultTrace**([$level]|[$loglevel, `$tracelevel`]|[$level, `$callback`])
: Reports the default log and trace level which is used for object as list of two elements. When not explicitly set, both are set to `WARNINGS`.

This method has three different uses. When one argument is specified, that `$level` is set for both loglevel as tracelevel.

With two arguments, the second determines which configuration you like. If the second argument is a CODE reference, you install a `$callback`. The loglevel will be set to NONE, and all warnings produced in your program will get passed to the `$callback` function. That function will get the problem level, the object or class which reports the problem, and the problem text passed as arguments.

In any case two values are returned: the first is the log level, the second represents the trace level. Both are special variables: in numeric context they deliver a value (the internally used value), and in string context the string name. Be warned that the string is always in singular form!

example: setting loglevels

```
my ($loglevel, $tracelevel) = Mail::Reporter−>defaultTrace;
Mail::Reporter−>defaultTrace('NOTICES');

my ($l, $t) = Mail::Reporter−>defaultTrace('WARNINGS', 'DEBUG');
print $l;     # prints "WARNING"  (no S!)
print $l+0;   # prints "4"
print "Auch" if $l >= $self−>logPriority('ERROR');

Mail::Reporter−>defaultTrace('NONE');  # silence all reports

$folder−>defaultTrace('DEBUG');   # Still set as global default!
$folder−>trace('DEBUG');          # local default
```

example: installing a callback

```
Mail::Reporter−>defaultTrace
```

`$obj`−>**errors**()
: Equivalent to

```
$folder−>report('ERRORS')
```

`$obj`−>**log**([$level, [$strings]])
Mail::Reporter−>**log**([$level, [$strings]])
: As instance method, this function has three different purposes. Without any argument, it returns one scalar containing the number which is internally used to represent the current log level, and the textual

representation of the string at the same time. See Scalar::Util method `dualvar` for an explanation.

With one argument, a new level of logging detail is set (specify a number of one of the predefined strings). With more arguments, it is a report which may need to be logged or traced.

As class method, only a message can be passed. The global configuration value set with **defaultTrace()** is used to decide whether the message is shown or ignored.

Each log-entry has a `$level` and a text string which will be constructed by joining the `$strings`. If there is no newline, it will be added.

example:

```
print $message->log;      # may print "NOTICE"
print $message->log +0;   # may print "3"
$message->log('ERRORS');  # sets a new level, returns the numeric value

$message->log(WARNING => "This message is too large.");
$folder ->log(NOTICE  => "Cannot read from file $filename.");
$manager->log(DEBUG   => "Hi there!", reverse sort @l);

Mail::Message->log(ERROR => 'Unknown');
```

`$obj`->**logPriority**($level)
Mail::Reporter->**logPriority**($level)

One error level (log or trace) has more than one representation: a numeric value and one or more strings. For instance, `4`, `'WARNING'`, and `'WARNINGS'` are all the same. You can specify any of these, and in return you get a dualvar (see Scalar::Util method `dualvar`) back, which contains the number and the singular form.

The higher the number, the more important the message. Only messages about `INTERNAL` problems are more important than `NONE`.

example:

```
my $r = Mail::Reporter->logPriority('WARNINGS');
my $r = Mail::Reporter->logPriority('WARNING');    # same
my $r = Mail::Reporter->logPriority(4);            # same, deprecated
print $r;      # prints 'WARNING'  (no S!)
print $r + 0;  # prints 4
if($r < Mail::Reporter->logPriority('ERROR')) {..} # true
```

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

Returns a list of `(key =` value)> pairs which can be used to initiate a new object with the same log-settings as this one.

example:

```
$head->new($folder->logSettings);
```

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

A special case of **log()**, which logs a `INTERNAL`−error and then croaks. This is used by extension writers.

`$obj`->**report**([$level])

Get logged reports, as list of strings. If a `$level` is specified, the log for that level is returned.

In case no `$level` is specified, you get all messages each as reference to a tuple with level and message.

example:

```
 my @warns = $message->report('WARNINGS');
   # previous indirectly callable with
   my @warns = $msg->warnings;

 print $folder->report('ERRORS');

 if($folder->report('DEBUG')) {...}

 my @reports = $folder->report;
 foreach (@reports) {
    my ($level, $text) = @$_;
    print "$level report: $text";
 }
```

`$obj->`**reportAll**([$level])

Report all messages which were produced by this object and all the objects which are maintained by this object. This will return a list of triplets, each containing a reference to the object which caught the report, the level of the report, and the message.

example:

```
 my $folder = Mail::Box::Manager->new->open(folder => 'inbox');
 my @reports = $folder->reportAll;
 foreach (@reports) {
    my ($object, $level, $text) = @$_;

    if($object->isa('Mail::Box')) {
       print "Folder $object: $level: $message";
    } elsif($object->isa('Mail::Message') {
       print "Message ".$object->seqnr.": $level: $message";
    }
 }
```

`$obj->`**trace**([$level])

Change the trace `$level` of the object. When no arguments are specified, the current level is returned only. It will be returned in one scalar which contains both the number which is internally used to represent the level, and the string which represents it. See **logPriority()**.

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

Equivalent to

```
 $folder->report('WARNINGS')
```

Cleanup

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

Cleanup the object.

DIAGNOSTICS

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.

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