

Merror

1.3

Generated by Doxygen 1.5.8

Mon Mar 8 17:18:20 2010



# Contents

<b>1</b>	<b>Merror</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Author . . . . .	1
1.3	Bugtracking . . . . .	1
1.4	Installation . . . . .	1
1.5	Example . . . . .	2
1.6	Errormapping file syntax . . . . .	3
<b>2</b>	<b>Class Index</b>	<b>5</b>
2.1	Class List . . . . .	5
<b>3</b>	<b>Class Documentation</b>	<b>7</b>
3.1	Merror Class Reference . . . . .	7
3.1.1	Detailed Description . . . . .	8
3.1.2	Member Function Documentation . . . . .	8
3.1.2.1	ec . . . . .	8
3.1.2.2	error . . . . .	8
3.1.2.3	et . . . . .	9
3.1.2.4	mapErrorCode . . . . .	9
3.1.2.5	mappedError . . . . .	9
3.1.2.6	merror_copy . . . . .	9
3.1.2.7	new . . . . .	10



# Chapter 1

## Merror

### 1.1 Introduction

[Merror](#) is a class implementing a mechanism to handle errors OOP style which means that it gives you the ability to easily set or get error codes, error descriptions and gives you the ability to print out a stacktrace.

### 1.2 Author

Markus Mazurczak <[coding@markus-mazurczak.de](mailto:coding@markus-mazurczak.de)>

### 1.3 Bugtracking

If you find any bugs or you want to have some features included register at <http://mantis.markus-mazurczak.de>

### 1.4 Installation

Installation must be done as with every other standard perl module installed manually:

- perl Makefile.PL
- make
- make test
- make install

If you want to install that module in some other than your perl standard paths specify PREFIX=, e.g.:

- perl Makefile.PL PREFIX=/home/foouser/perl/lib

## 1.5 Example

Lets consider we created the following errormapping file named "errorcodes" in /home/foobar/test:

- 0: Test error
- 1: Another test error
- 2: Yet another one
- #3: This will never be printed out

Lets create the following test perl module:

```
package test:
BEGIN {
    use strict;
    use warnings;
    use File::Spec;
    use File::Basename;
}

use Merror;

sub new {
    my $invocant = shift;
    my $class = ref($invocant) || $invocant;
    my $self = {
        MERROR => Merror->new(stackdepth => 16, errorfile => "/home/foobar/test/errorcodes", ramusage => 1);
        TEXT => "Test messages";
    };

    bless $self, $class;
    return($self);
}

sub tellme {
    my $self = shift;
    my $errorcode = shift;
    #indicate that an error happened
    $self->{MERROR}->error(1);
    #set an error code and the mapped description
    $self->{MERROR}->mappedError($errorcode);
}

sub tellmethesecond {
    my $self = shift;
    #indicate that an error happened
    $self->{MERROR}->error(1);
    #set an error code
    $self->{MERROR}->ec($errorcode);
    #set the error description
    $self->{MERROR}->et("This message will be the error description");
}

1;
```

Now lets create a file foo.pl in /home/foobar/test

```
#!/usr/bin/perl
BEGIN {
    use strict;
    use warnings;
```

```

    use Merror;
}

use test;

my $foobar = Test->new();

$foobar->tellmethesecond();
if($foobar->{MERROR}->error()) {
    #print out the error code
    print("Errorcode: " . $foobar->{MERROR}->ec . "\n");
    #print out the error description
    print("Errordescription: " . $foobar->{MERROR}->et . "\n");
    #print out a stacktrace
    print("Stacktrace: \n");
    $foobar->{MERROR}->stacktrace;
}

$foobar->tellme();
if($foobar->{MERROR}->error(1)) {
    #print out the error code
    print("Errorcode: " . $foobar->{MERROR}->ec . "\n");
    #print out the error description
    print("Errordescription: " . $foobar->{MERROR}->et . "\n");
    #print out a stacktrace
    print("Stacktrace: \n");
    $foobar->{MERROR}->stacktrace;
}

```

## 1.6 Errormapping file syntax

Every line of the errorfile consists out of 3 parts.

- Errornumber
- :
- Errordescription

It does not matter if there are any whitespaces between the parts, they will be substituted. The perl regexp looks like the following:

```
/^\s{1,}\s{0,}:\s{0,}(.*)/
```



# Chapter 2

## Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">Merror</a> .....	7
------------------------------	---



# Chapter 3

## Class Documentation

### 3.1 Merror Class Reference

#### Public Functions

- public [Merror](#) **new** (int stackdepth=> 64, string errorfile=> undef, boolean ramusage=> 0)  
*Creates a new [Merror](#) object.*
- public int **error** (boolean flag=0)  
*If called with an value != undef the internal state will be considered as an error. If called without a parameter, it will return the internal state where a positive value indicates an error. If called with a value != undef a stacktrace will be created.*
- public int **ec** (int errorCode=0)  
*Returns or sets a error code.*
- public string **et** (string errorDescription="")  
*Returns or sets an error description.*
- public void **mappedError** (int errorcode)  
*Sets the errorcode to the given one and sets the error description from the error mapping file.*
- public void **stacktrace** ()  
*Prints out a captured stacktrace in case of error.*
- public void **merror\_copy** ([Merror](#) to)  
*You could treat this function as a copy operator for [Merror](#) objects It copies the complete internal state into a new object.*

#### Private Functions

- private void **fillstack** ()  
*Captures the stacktrace with a max depth of  $\$self->\{STACKDEPTH\}$ .*

- private void [parseErrorFile](#) ()

*If user wants to parse the complete error mapping file and save it into ram than this function is called.*

- private string [mapErrorCode](#) (int errorcode)

*Returns the error description of the given errorcode. If errorcode is undef than an undefined error will be returned.*

### 3.1.1 Detailed Description

Implementing a OOP style error handling class with an ability of printing out a stacktrace Revision:

#### Rev

4

Last changed:

#### Date

2010-02-17 23:32:19 +0100 (Wed, 17 Feb 2010)

By:

#### Author

nexus

### 3.1.2 Member Function Documentation

#### 3.1.2.1 public int Merror::ec (int *errorCode* = 0)

Returns or sets a error code.

#### Parameters:

*errorCode* If a value != undef is given, than this value will be treated as an errorcode If this function is called without a parameter it will return the last set errorCode

#### Returns:

scalar An error code

#### 3.1.2.2 public int Merror::error (boolean *flag* = 0)

If called with an value != undef the internal state will be considered as an error. If called without a parameter, it will return the internal state where a positive value indicates an error. If called with a value != undef a stacktrace will be created.

#### Parameters:

*flag* The parameter is optional and should only be set in classes using [Merror](#) not in interfaces implementing something

**Returns:**

scalar If called without a parameter it will return a 1 indicating that an error happended or a 0 to indicate that everything is fine

**3.1.2.3 public string Merror::et (string *errorDescription* = " ")**

Returns or sets an error description.

**Parameters:**

*errorDescription* If a value != undef is given this value will be treated as a string and this string will be the error description. If no value is given than the last set error description will be given back

**Returns:**

scalar Error description

**3.1.2.4 private string Merror::mapErrorCode (int *errorcode*)**

Returns the error description of the given errorcode. If errorcode is undef than an undefined error will be returned.

**Parameters:**

*errorcode* The errorcode

**Returns:**

string

**3.1.2.5 public void Merror::mappedError (int *errorcode*)**

Sets the errorcode to the given one and sets the error description from the error mapping file.

**Parameters:**

*errorcode* The errorcode

**3.1.2.6 public void Merror::merror\_copy (Merror *to*)**

You could treat this function as a copy operator for [Merror](#) objects It copies the complete internal state into a new object.

**Parameters:**

*to* The destination hash structure

### 3.1.2.7 public Merror Merror::new (int *stackdepth*, 64, string *errorfile*, undef, boolean *ramusage*, 0)

Creates a new [Merror](#) object.

#### Parameters:

*stackdepth* Describes how deep the stacktrace should trace back the calls. Default = 64

*errorfile* Path to the file that contains the errorcode to errordescription mapping. Default = undef

*ramusage* Describes if set to 1 that the complete errormapping will be held in ram. Default = 0

#### Returns:

[Merror](#) object

The documentation for this class was generated from the following file:

- lib/Merror.pm

# Index

- ec
  - [Merror, 8](#)
- error
  - [Merror, 8](#)
- et
  - [Merror, 9](#)
- mapErrorCode
  - [Merror, 9](#)
- mappedError
  - [Merror, 9](#)
- Merror, [7](#)
  - [ec, 8](#)
  - [error, 8](#)
  - [et, 9](#)
  - [mapErrorCode, 9](#)
  - [mappedError, 9](#)
  - [merror\\_copy, 9](#)
  - [new, 9](#)
- merror\_copy
  - [Merror, 9](#)
- new
  - [Merror, 9](#)