NAME

Log::Message::Simple - Simplified interface to Log::Message

SYNOPSIS

```
use Log::Message::Simple qw[msg error debug
carp croak cluck confess];

use Log::Message::Simple qw[:STD :CARP];

### standard reporting functionality
msg( "Connecting to database", $verbose );
error( "Database connection failed: $@", $verbose );
debug( "Connection arguments were: $args", $debug );

### standard carp functionality
carp( "Wrong arguments passed: @_" );
croak( "Fatal: wrong arguments passed: @_" );
cluck( "Wrong arguments passed -- including stacktrace: @_" );
confess("Fatal: wrong arguments passed -- including stacktrace: @_" );

### retrieve individual message
my @stack = Log::Message::Simple->stack;
my @stack = Log::Message::Simple->flush;

### retrieve the entire stack in printable form
my $msgs = Log::Message::Simple->stack_as_string;
my $trace = Log::Message::Simple->stack_as_string(1);

### redirect output
local $Log::Message::Simple::MSG_FH = \*STDERR;
local $Log::Message::Simple::ERROR_FH = \*STDERR;
local $Log::Message::Simple::DEBUG_FH = \*STDERR;

### force a stacktrace on error
local $Log::Message::Simple::STACKTRACE_ON_ERROR = 1
```

DESCRIPTION

This module provides standardized logging facilities using the `Log::Message` module.

FUNCTIONS

msg(“message string” [,VERBOSE])

Records a message on the stack, and prints it to `STDOUT` (or actually `$MSG_FH`, see the `GLOBAL VARIABLES` section below), if the `VERBOSE` option is true. The `VERBOSE` option defaults to false.

Exported by default, or using the `:STD` tag.

debug(“message string” [,VERBOSE])

Records a debug message on the stack, and prints it to `STDOUT` (or actually `$DEBUG_FH`, see the `GLOBAL VARIABLES`section belo w), if the `VERBOSE` option is true. The `VERBOSE` option defaults to false.

Exported by default, or using the `:STD` tag.

error(“error string” [,VERBOSE])

Records an error on the stack, and prints it to `STDERR` (or actually `$ERROR_FH`, see the `GLOBAL VARIABLES` sections below), if the `VERBOSE` option is true. The `VERBOSE` options defaults to true.

Exported by default, or using the `:STD` tag.

carp();
Provides functionality equal to `Carp::carp()` while still logging to the stack.

Exported by using the `:CARP` tag.

croak();
Provides functionality equal to `Carp::croak()` while still logging to the stack.

Exported by using the `:CARP` tag.

confess();
Provides functionality equal to `Carp::confess()` while still logging to the stack.

Exported by using the `:CARP` tag.

cluck();
Provides functionality equal to `Carp::cluck()` while still logging to the stack.

Exported by using the `:CARP` tag.

CLASS METHODS

Log::Message::Simple–>*stack()*
Retrieves all the items on the stack. Since `Log::Message::Simple` is implemented using `Log::Message` consult its manpage for the function `retrieve` to see what is returned and how to use the items.

Log::Message::Simple–>stack_as_string([TRACE])
Returns the whole stack as a printable string. If the `TRACE` option is true all items are returned with `Carp::longmess` output, rather than just the message. `TRACE` defaults to false.

Log::Message::Simple–>*flush()*
Removes all the items from the stack and returns them. Since `Log::Message::Simple` is implemented using `Log::Message` consult its manpage for the function `retrieve` to see what is returned and how to use the items.

GLOBAL VARIABLES

`$ERROR_FH`
This is the filehandle all the messages sent to `error()` are being printed. This defaults to `*STDERR`.

`$MSG_FH`
This is the filehandle all the messages sent to `msg()` are being printed. This default to `*STDOUT`.

`$DEBUG_FH`
This is the filehandle all the messages sent to `debug()` are being printed. This default to `*STDOUT`.

`$STACKTRACE_ON_ERROR`
If this option is set to `true`, every call to `error()` will generate a stacktrace using `Carp::shortmess()`. Defaults to`false`