Rocky Enterprise Linux 9.2 Manual Pages on command 'Data::Dump::Trace.3pm'

C:\>man Data::Dump::Trace.3pm

Data::Dump::Trace(3pm) User Contributed Perl Documentation Data::Dump::Trace(3pm)

NAME

Data::Dump::Trace - Helpers to trace function and method calls

SYNOPSIS

```
use Data::Dump::Trace qw(autowrap mcall);

autowrap("LWP::UserAgent" => "ua", "HTTP::Response" => "res");

use LWP::UserAgent;
$ua = mcall(LWP::UserAgent => "new");      # instead of LWP::UserAgent->new;
$ua->get("http://www.example.com")->dump;
```

DESCRIPTION

The following functions are provided:

autowrap($class)

autowrap($class => $prefix)

autowrap($class1 => $prefix1, $class2 => $prefix2, ...)

autowrap($class1 => \%info1, $class2 => \%info2, ...)

Register classes whose objects are automatically wrapped when returned by one

of the call functions below. If $prefix is provided it will be used as to name the objects.

Alternative is to pass an %info hash for each class. The recognized keys are:

prefix => $string

The prefix string used to name objects of this type.

proto => \%hash

A hash of prototypes to use for the methods when an object is wrapped.

wrap(name => $str, func => \&func, proto => $proto)

wrap(name => $str, obj => $obj, proto => \%hash)

Returns a wrapped function or object. When a wrapped function is invoked then a trace is printed after the underlying function has returned. When a method on a wrapped object is invoked then a trace is printed after the methods on the underlying objects has returned.

See "Prototypes" for description of the "proto" argument.

call($name, \&func, $proto, @ARGS)

Calls the given function with the given arguments. The trace will use $name as the name of the function.

See "Prototypes" for description of the $proto argument.

mcall($class, $method, $proto, @ARGS)

mcall($object, $method, $proto, @ARGS)

Calls the given method with the given arguments.

See "Prototypes" for description of the $proto argument.

trace($symbol, $prototype)

Replaces the function given by $symbol with a wrapped function.

Prototypes

Note: The prototype string syntax described here is experimental and likely to change in revisions of this interface.

The $proto argument to call() and mcall() can optionally provide a prototype for the function call. This give the tracer hints about how to best format the argument lists and if there are in/out or out arguments. The general form for the prototype string is:

<arguments> = <return_value>

The default prototype is "@ = @"; list of values as input and list of values as output.

The value '%' can be used for both arguments and return value to say that key/value pair style lists are used.

Alternatively, individual positional arguments can be listed each represented by a letter:

"i" input argument

"o" output argument

"O" both input and output argument

If the return value prototype has "!" appended, then it signals that this function sets errno ($!) when it returns a false value. The trace will display the current value of errno in that case.

If the return value prototype looks like a variable name (with "$" prefix), and the

function returns a blessed object, then the variable name will be used as prefix and the returned object automatically traced.

SEE ALSO

Data::Dump

AUTHOR