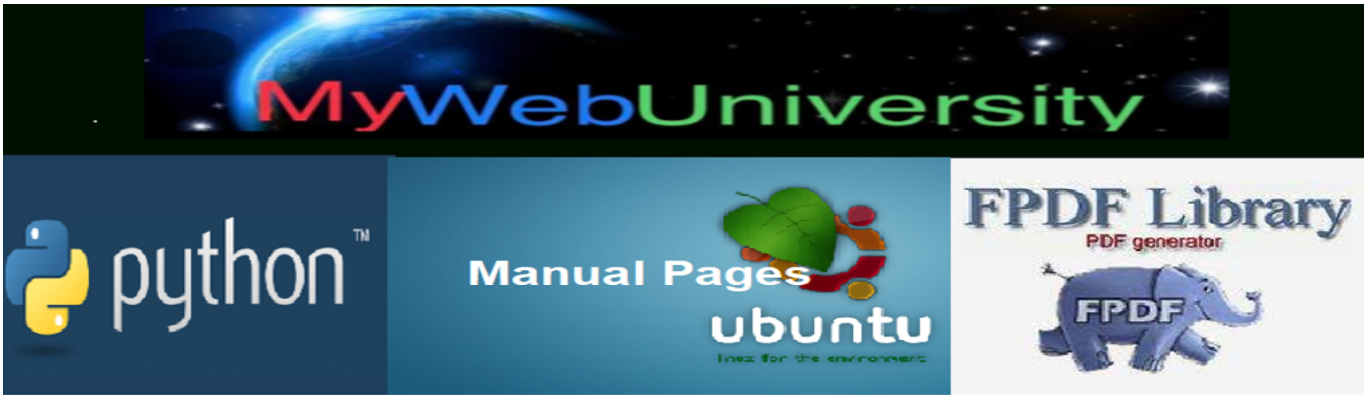

Rocky Enterprise Linux 9.2 Manual Pages on command 'Data::Dump::Filtered.3pm'

$ man Data::Dump::Filtered.3pm

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

NAME

Data::Dump::Filtered - Pretty printing with filtering

DESCRIPTION

The following functions are provided:

add_dump_filter(\&filter)

This registers a filter function to be used by the regular Data::Dump::dump() function. By default no filters are active.

Since registering filters has a global effect is might be more appropriate to use the dump_filtered() function instead.

remove_dump_filter(\&filter)

Unregister the given callback function as filter callback. This undoes the effect of add_filter.

dump_filtered(..., \&filter)

Works like Data::Dump::dump(), but the last argument should be a filter callback function. As objects are visited the filter callback is invoked at it might influence how objects are dumped.

Any filters registered with add_filter() are ignored when this interface is invoked.

Actually, passing "undef" as \&filter is allowed and "dump_filtered(..., undef)" is the official way to force unfiltered dumps.

Filter callback

A filter callback is a function that will be invoked with 2 arguments; a context object and reference to the object currently visited. The return value should either be a hash

reference or "undef".

```
sub filter_callback {
    my($ctx, $object_ref) = @_;
    ...
    return { ... }
}
```

If the filter callback returns "undef" (or nothing) then normal processing and formatting of the visited object happens. If the filter callback returns a hash it might replace or annotate the representation of the current object.

Filter context

The context object provide methods that can be used to determine what kind of object is currently visited and where it's located. The context object has the following interface:

$ctx->object_ref

Alternative way to obtain a reference to the current object

$ctx->class

If the object is blessed this return the class. Returns "" for objects not blessed.

$ctx->reftype

Returns what kind of object this is. It's a string like "SCALAR", "ARRAY", "HASH", "CODE",...

$ctx->is_ref

Returns true if a reference was provided.

$ctx->is_blessed

Returns true if the object is blessed. Actually, this is just an alias for "$ctx->class".

$ctx->is_array

Returns true if the object is an array

$ctx->is_hash

Returns true if the object is a hash

$ctx->is_scalar

Returns true if the object is a scalar (a string or a number)

$ctx->is_code

Returns true if the object is a function (aka subroutine)

$ctx->container_class

Returns the class of the innermost container that contains this object. Returns "" if there is no blessed container.

$ctx->container_self

Returns an textual expression relative to the container object that names this object. The variable $self in this expression is the container itself.

$ctx->object_isa($class)

Returns TRUE if the current object is of the given class or is of a subclass.

$ctx->container_isa($class)

Returns TRUE if the innermost container is of the given class or is of a subclass.

$ctx->depth

Returns how many levels deep have we recursed into the structure (from the original dump_filtered() arguments).

$ctx->expr

$ctx->expr($top_level_name)

Returns an textual expression that denotes the current object. In the expression $var is used as the name of the top level object dumped. This can be overridden by providing a different name as argument.

Filter return hash

The following elements has significance in the returned hash:

dump => $string

incorporate the given string as the representation for the current value

object => $value

dump the given value instead of the one visited and passed in as $object. Basically the same as specifying "dump => Data::Dump::dump($value)".

comment => $comment

prefix the value with the given comment string

bless => $class

make it look as if the current object is of the given $class instead of the class it really has (if any). The internals of the object is dumped in the regular way. The $class can be the empty string to make Data::Dump pretend the object wasn't blessed at all.

hide_keys => ['key1', 'key2',...]

hide_keys => \&code

If the $object is a hash dump is as normal but pretend that the listed keys did not exist. If the argument is a function then the function is called to determine if the given key should be hidden.

SEE ALSO

Data::Dump