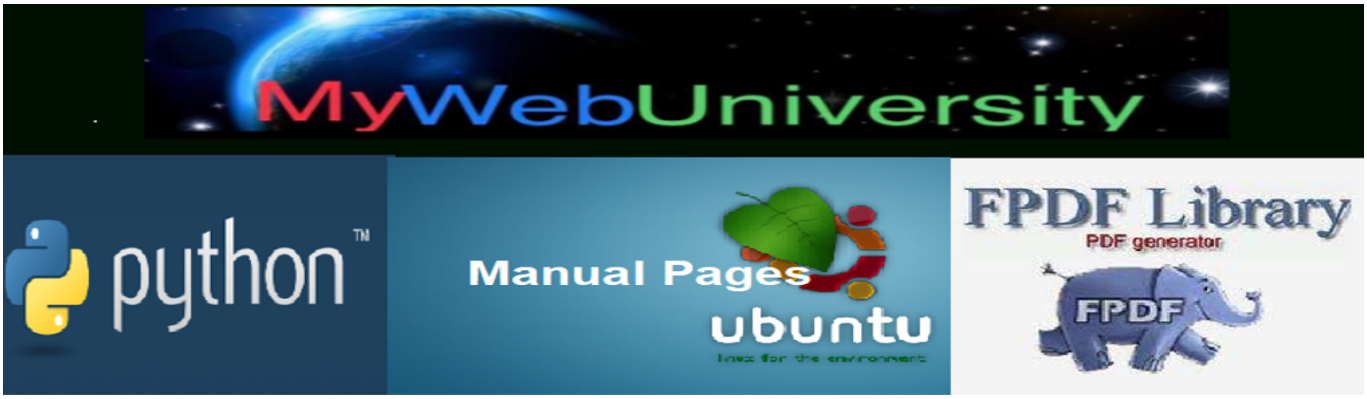

Rocky Enterprise Linux 9.2 Manual Pages on command 'Class::Load.3pm'

$ man Class::Load.3pm

Class::Load(3pm) User Contributed Perl Documentation Class::Load(3pm)

NAME

Class::Load - A working (require "Class::Name") and more

VERSION

version 0.25

SYNOPSIS

```
use Class::Load ':all';

try_load_class('Class::Name')
    or plan skip_all => "Class::Name required to run these tests";

load_class('Class::Name');

is_class_loaded('Class::Name');

my $baseclass = load_optional_class('Class::Name::MightExist')
    ? 'Class::Name::MightExist'
    : 'Class::Name::Default';
```

DESCRIPTION

"require EXPR" only accepts "Class/Name.pm" style module names, not "Class::Name". How frustrating! For that, we provide "load_class 'Class::Name'".

It's often useful to test whether a module can be loaded, instead of throwing an error when it's not available. For that, we provide "try_load_class 'Class::Name'".

Finally, sometimes we need to know whether a particular class has been loaded. Asking %INC is an option, but that will miss inner packages and any class for which the filename does not correspond to the package name. For that, we provide "is_class_loaded 'Class::Name'".

FUNCTIONS

load_class Class::Name, \%options

"load_class" will load "Class::Name" or throw an error, much like "require".

If "Class::Name" is already loaded (checked with "is_class_loaded") then it will not try to load the class. This is useful when you have inner packages which "require" does not check.

The %options hash currently accepts one key, "-version". If you specify a version, then this subroutine will call "Class::Name->VERSION($options{-version})" internally, which will throw an error if the class's version is not equal to or greater than the version you requested.

This method will return the name of the class on success.

try_load_class Class::Name, \%options -> (0|1, error message)

Returns 1 if the class was loaded, 0 if it was not. If the class was not loaded, the error will be returned as a second return value in list context.

Again, if "Class::Name" is already loaded (checked with "is_class_loaded") then it will not try to load the class. This is useful when you have inner packages which "require" does not check.

Like "load_class", you can pass a "-version" in %options. If the version is not sufficient, then this subroutine will return false.

is_class_loaded Class::Name, \%options -> 0|1

This uses a number of heuristics to determine if the class "Class::Name" is loaded. There heuristics were taken from Class::MOP's old pure-perl implementation.

Like "load_class", you can pass a "-version" in %options. If the version is not sufficient, then this subroutine will return false.

load_first_existing_class Class::Name, \%options, ...

This attempts to load the first loadable class in the list of classes given. Each class name can be followed by an options hash reference.

If any one of the classes loads and passes the optional version check, that class name will be returned. If none of the classes can be loaded (or none pass their version check), then an error will be thrown.

If, when attempting to load a class, it fails to load because of a syntax error, then an error will be thrown immediately.

load_optional_class Class::Name, \%options -> 0|1

"load_optional_class" is a lot like "try_load_class", but also a lot like "load_class".

If the class exists, and it works, then it will return 1. If you specify a version in %options, then the version check must succeed or it will return 0.

If the class doesn't exist, and it appears to not exist on disk either, it will return 0.

If the class exists on disk, but loading from disk results in an error (e.g.: a syntax error), then it will "croak" with that error.

This is useful for using if you want a fallback module system, i.e.:

```
my $class = load_optional_class($foo) ? $foo : $default;
```

That way, if $foo does exist, but can't be loaded due to error, you won't get the behaviour of it simply not existing.

CAVEATS

Because of some of the heuristics that this module uses to infer whether a module has been loaded, some false positives may occur in "is_class_loaded" checks (which are also performed internally in other interfaces) -- if a class has started to be loaded but then dies, it may appear that it has already been loaded, which can cause other things to make the wrong decision. Module::Runtime doesn't have this issue, but it also doesn't do some things that this module does -- for example gracefully handle packages that have been defined inline in the same file as another package.

SEE ALSO

<http://blog.fox.geek.nz/2010/11/searching-design-spec-for-ultimate.html>

This blog post is a good overview of the current state of the existing modules for loading other modules in various ways.

<http://blog.fox.geek.nz/2010/11/handling-optional-requirements-with.html>

This blog post describes how to handle optional modules with Class::Load.

<http://d.hatena.ne.jp/tokuhirom/20110202/1296598578>

This Japanese blog post describes why DBIx::Skinny now uses Class::Load over its competitors.

Moose, Jifty, Prophet, etc

This module was designed to be used anywhere you have "if (eval "require $module"; 1)", which occurs in many large projects.

Module::Runtime

A leaner approach to loading modules

SUPPORT

Bugs may be submitted through the RT bug tracker

<https://rt.cpan.org/Public/Dist/Display.html?Name=Class-Load> (or

bug-Class-Load@rt.cpan.org <mailto:bug-Class-Load@rt.cpan.org>).

There is also a mailing list available for users of this distribution, at

<http://lists.perl.org/list/moose.html>.

There is also an irc channel available for users of this distribution, at "#moose" on

"irc.perl.org" <irc://irc.perl.org/#moose>.

AUTHOR

Shawn M Moore <sartak at bestpractical.com>

CONTRIBUTORS

- Dave Rolsky <autarch@urth.org>
- Karen Etheridge <ether@cpan.org>
- Shawn Moore <sartak@bestpractical.com>
- Jesse Luehrs <doy@tozt.net>
- Kent Fredric <kentfredric@gmail.com>
- Paul Howarth <paul@city-fan.org>
- Olivier Mengu? <dolmen@cpan.org>
- Caleb Cushing <xenoterracide@gmail.com>

COPYRIGHT AND LICENSE