

NAME

Module::Metadata - Gather package and POD information from perl module files

SYNOPSIS

```
use Module::Metadata;

# information about a .pm file
my $info = Module::Metadata->new_from_file( $file );
my $version = $info->version;

# CPAN META 'provides' field for .pm files in a directory
my $provides = Module::Metadata->provides(
    dir => 'lib', version => 2
);
```

DESCRIPTION

This module provides a standard way to gather metadata about a .pm file through (mostly) static analysis and (some) code execution. When determining the version of a module, the `$VERSION` assignment is `eval`d, as is traditional in the CPAN toolchain.

USAGE**Class methods**

`new_from_file($filename, collect_pod => 1)`

Constructs a `Module::Metadata` object given the path to a file. Returns undef if the filename does not exist.

`collect_pod` is an optional boolean argument that determines whether POD data is collected and stored for reference. POD data is not collected by default. POD headings are always collected.

If the file begins by an UTF-8, UTF-16BE or UTF-16LE byte-order mark, then it is skipped before processing, and the content of the file is also decoded appropriately starting from perl 5.8.

`new_from_handle($handle, $filename, collect_pod => 1)`

This works just like `new_from_file`, except that a handle can be provided as the first argument.

Note that there is no validation to confirm that the handle is a handle or something that can act like one. Passing something that isn't a handle will cause an exception when trying to read from it. The `filename` argument is mandatory or undef will be returned.

You are responsible for setting the decoding layers on `$handle` if required.

`new_from_module($module, collect_pod => 1, inc => \@dirs)`

Constructs a `Module::Metadata` object given a module or package name. Returns undef if the module cannot be found.

In addition to accepting the `collect_pod` argument as described above, this method accepts an `inc` argument which is a reference to an array of directories to search for the module. If none are given, the default is `@INC`.

If the file that contains the module begins by an UTF-8, UTF-16BE or UTF-16LE byte-order mark, then it is skipped before processing, and the content of the file is also decoded appropriately starting from perl 5.8.

`find_module_by_name($module, \@dirs)`

Returns the path to a module given the module or package name. A list of directories can be passed in as an optional parameter, otherwise `@INC` is searched.

Can be called as either an object or a class method.

`find_module_dir_by_name($module, \@dirs)`

Returns the entry in `@dirs` (or `@INC` by default) that contains the module `$module`. A list of directories can be passed in as an optional parameter, otherwise `@INC` is searched.

Can be called as either an object or a class method.

`provides(%options)`

This is a convenience wrapper around `package_versions_from_directory` to generate a CPAN META `provides` data structure. It takes key/value pairs. Valid option keys include:

`version (required)`

Specifies which version of the `CPAN::Meta::Spec` should be used as the format of the `provides` output. Currently only '1.4' and '2' are supported (and their format is identical). This may change in the future as the definition of `provides` changes.

The `version` option is required. If it is omitted or if an unsupported version is given, then `provides` will throw an error.

`dir` Directory to search recursively for `.pm` files. May not be specified with `files`.

`files`

Array reference of files to examine. May not be specified with `dir`.

`prefix`

String to prepend to the `file` field of the resulting output. This defaults to `lib`, which is the common case for most CPAN distributions with their `.pm` files in `lib`. This option ensures the META information has the correct relative path even when the `dir` or `files` arguments are absolute or have relative paths from a location other than the distribution root.

For example, given `dir` of 'lib' and `prefix` of 'lib', the return value is a hashref of the form:

```
{
  'Package::Name' => {
    version => '0.123',
    file => 'lib/Package/Name.pm'
  },
  'OtherPackage::Name' => ...
}
```

`package_versions_from_directory($dir, \@files?)`

Scans `$dir` for `.pm` files (unless `@files` is given, in which case looks for those files in `$dir` - and reads each file for packages and versions, returning a hashref of the form:

```
{
  'Package::Name' => {
    version => '0.123',
    file => 'Package/Name.pm'
  },
  'OtherPackage::Name' => ...
}
```

The `DB` and `main` packages are always omitted, as are any “private” packages that have leading underscores in the namespace (e.g. `Foo::_private`)

Note that the file path is relative to `$dir` if that is specified. This **must not** be used directly for CPAN META `provides`. See the `provides` method instead.

`log_info (internal)`

Used internally to perform logging; imported from `Log::Contextual` if `Log::Contextual` has already been loaded, otherwise simply calls `warn`.

Object methods**name()**

Returns the name of the package represented by this module. If there are more than one packages, it makes a best guess based on the filename. If it's a script (i.e. not a *.pm) the package name is 'main'.

version(\$package)

Returns the version as defined by the `$VERSION` variable for the package as returned by the `name` method if no arguments are given. If given the name of a package it will attempt to return the version of that package if it is specified in the file.

filename()

Returns the absolute path to the file.

packages_inside()

Returns a list of packages. Note: this is a raw list of packages discovered (or assumed, in the case of `main`). It is not filtered for `DB`, `main` or private packages the way the `provides` method does. Invalid package names are not returned, for example "Foo:Bar". Strange but valid package names are returned, for example "Foo::Bar::", and are left up to the caller on how to handle.

pod_inside()

Returns a list of POD sections.

contains_pod()

Returns true if there is any POD in the file.

pod(\$section)

Returns the POD data in the given section.

AUTHOR

Original code from [Module::Build::ModuleInfo](#) by Ken Williams <kwilliams@cpan.org>, Randy W. Sims <RandyS@ThePierianSpring.org>

Released as [Module::Metadata](#) by Matt S Trout (mst) <mst@shadowcat.co.uk> with assistance from David Golden (xdg) <dagolden@cpan.org>.

COPYRIGHT & LICENSE

Original code Copyright (c) 2001-2011 Ken Williams. Additional code Copyright (c) 2010-2011 Matt Trout and David Golden. All rights reserved.

This library is free software; you can redistribute it and/or modify it under the same terms as Perl itself.