NAME

Package::DeprecationManager − Manage deprecation warnings for your distribution

VERSION

version 0.04

SYNOPSIS

```
package My::Class;

use Package::DeprecationManager −deprecations => {
    'My::Class::foo' => '0.02',
    'My::Class::bar' => '0.05',
    'feature−X'      => '0.07',
};

sub foo {
    deprecated( 'Do not call foo!' );

    ...
}

sub bar {
    deprecated();

    ...
}

sub baz {
    my %args = @_;

    if ( $args{foo} ) {
        deprecated(
            message => ...,
            feature => 'feature−X',
        );
    }
}

package Other::Class;

use My::Class −api_version => '0.04';

My::Class−>new()−>foo(); # warns
My::Class−>new()−>bar(); # does not warn
My::Class−>new()−>far(); # does not warn again
```

DESCRIPTION

This module allows you to manage a set of deprecations for one or more modules.

When you import `Package::DeprecationManager`, you must provide a set of `−deprecations` as a hash ref. The keys are “feature” names, and the values are the version when that feature was deprecated.

In many cases, you can simply use the fully qualified name of a subroutine or method as the feature name. This works for cases where the whole subroutine is deprecated. However, the feature names can be any string. This is useful if you don’t want to deprecate an entire subroutine, just a certain usage.

You can also provide an optional array reference in the `−ignore` parameter. This is a list of package names to ignore when looking at the stack to figure out what code used the deprecated feature. This should be packages in your distribution that can appear on the call stack when a deprecated feature is used.

As part of the import process, `Package::DeprecationManager` will export two subroutines into its caller. It proves an `import()` sub for the caller and a `deprecated()` sub.

The `import()` sub allows callers of *your* class to specify an `−api_version` parameter. If this is supplied, then deprecation warnings are only issued for deprecations for api versions earlier than the one specified.

You must call `deprecated()` sub in each deprecated subroutine. When called, it will issue a warning using `Carp::cluck()`.

The `deprecated()` sub can be called in several ways. If you do not pass any arguments, it will generate an appropriate warning message. If you pass a single argument, this is used as the warning message.

Finally, you can call it with named arguments. Currently, the only allowed names are `message` and `feature`. The `feature` argument should correspond to the feature name passed in the `−deprecations` hash.

If you don't explicitly specify a feature, the `deprecated()` sub uses `caller()` to identify its caller, using its fully qualified subroutine name.

A given deprecation warning is only issued once for a given package. This module tracks this based on both the feature name *and* the error message itself. This means that if you provide severaldifferent error messages for the same feature, all of those errors will appear.

BUGS

Please report any bugs or feature requests to `bug−package−deprecationmanager AT rt DOT cpan DOT org`, or through the web interface at <http://rt.cpan.org>. I will be notified, and then you'll automatically be notified of progress on your bug as I make changes.

DONATIONS

If you'd like to thank me for the work I've done on this module, please consider making a ''donation'' to me via PayPal. I spend a lot of free time creating free software, and would appreciate any support you'd care to offer.

Please note that **I am not suggesting that you must do this** in order for me to continue working on this particular software. I will continue to do so, inasmuch as I have in the past, for as long as it interests me.

Similarly, a donation made in this way will probably not make me work on this software much more, unless I get so many donations that I can consider working on free software full time, which seems unlikely at best.

To donate, log into PayPal and send money to autarch AT urth DOT org or use the button on this page: <http://www.urth.org/˜autarch/fs−donation.html>

CREDITS

The idea for this functionality and some of its implementation was originally created as Class::MOP::Deprecated by Goro Fuji.

AUTHOR

```
Dave Rolsky <autarch AT urth DOT org>
```

COPYRIGHT AND LICENSE

This software is Copyright (c) 2010 by Dave Rolsky.

This is free software, licensed under:

```
The Artistic License 2.0
```