NAME

MooseX::NonMoose − easy subclassing of non−Moose classes

VERSION

version 0.26

SYNOPSIS

```
package Term::VT102::NBased;
use Moose;
use MooseX::NonMoose;
extends 'Term::VT102';

has [qw/x_base y_base/] => (
    is      => 'ro',
    isa     => 'Int',
    default => 1,
);

around x => sub {
    my $orig = shift;
    my $self = shift;
    $self->$orig(@_) + $self->x_base - 1;
};

# ... (wrap other methods)

no Moose;
# no need to fiddle with inline_constructor here
__PACKAGE__->meta->make_immutable;

my $vt = Term::VT102::NBased->new(x_base => 0, y_base => 0);
```

DESCRIPTION

`MooseX::NonMoose` allows for easily subclassing non-Moose classes with Moose, taking care of the annoying details connected with doing this, such as setting up proper inheritance from Moose::Object and installing (and inlining, at `make_immutable` time) a constructor that makes sure things like `BUILD` methods are called. It tries to be as non-intrusive as possible − when this module is used, inheriting from non-Moose classes and inheriting from Moose classes should work identically, aside from the few caveats mentioned below. One of the goals of this module is that including it in a Moose::Exporter−based package used across an entire application should be possible, without interfering with classes that only inherit from Moose modules, or even classes that don't inherit from anything at all.

There are several ways to use this module. The most straightforward is to just `use MooseX::NonMoose;` in your class; this should set up everything necessary for extending non-Moose modules. MooseX::NonMoose::Meta::Role::Class and MooseX::NonMoose::Meta::Role::Constructor can also be applied to your metaclasses manually, either by passing a `−traits` option to your `use Moose;` line, or by applying them using Moose::Util::MetaRole in a Moose::Exporter−based package. MooseX::NonMoose::Meta::Role::Class is the part that provides the main functionality of this module; if you don't care about inlining, this is all you need to worry about. Applying MooseX::NonMoose::Meta::Role::Constructor as well will provide an inlined constructor when you immutabilize your class.

`MooseX::NonMoose` allows you to manipulate the argument list that gets passed to the superclass constructor by defining a `FOREIGNBUILDARGS` method. This is called with the same argument list as the `BUILDARGS` method, but should return a list of arguments to pass to the superclass constructor. This allows `MooseX::NonMoose` to support superclasses whose constructors would get confused by the extra arguments that Moose requires (for attributes, etc.)

Not all non-Moose classes use `new` as the name of their constructor. This module allows you to extend these classes by explicitly stating which method is the constructor, during the call to `extends`. The syntax looks like this:

```
extends 'Foo' => { -constructor_name => 'create' };
```

similar to how you can already pass `-version` in the `extends` call in a similar way.

BUGS/CAVEATS

- The reference that the non-Moose class uses as its instance type **must** match the instance type that Moose is using. Moose's default instance type is a hashref, but other modules exist to make Moose use other instance types. MooseX::InsideOut is the most general solution − it should work with any class. For globref-based classes in particular, MooseX::GlobRef will also allow Moose to work. For more information, see the `032-moosex-insideout` and `033-moosex-globref` tests bundled with this dist.
- Modifying your class' `@ISA` after an initial `extends` call will potentially cause problems if any of those new entries in the `@ISA` override the constructor. `MooseX::NonMoose` wraps the nearest `new()` method at the time `extends` is called and will not see any other `new()` methods in the `@ISA` hierarchy.
- Completely overriding the constructor in a class using `MooseX::NonMoose` (i.e. using `sub new { ... }`) currently doesn't work, although using method modifiers on the constructor should work identically to normal Moose classes.

Please report any bugs to GitHub Issues at <https://github.com/doy/moosex-nonmoose/issues>.

SEE ALSO

- "How do I make non-Moose constructors work with Moose?" in Moose::Manual::FAQ
- MooseX::Alien

  serves the same purpose, but with a radically different (and far more hackish) implementation.

SUPPORT

You can find this documentation for this module with the perldoc command.

```
perldoc MooseX::NonMoose
```

You can also look for information at:

- MetaCPAN

  <https://metacpan.org/release/MooseX-NonMoose>
- Github

  <https://github.com/doy/moosex-nonmoose>
- RT: CPAN's request tracker

  <http://rt.cpan.org/NoAuth/Bugs.html?Dist=MooseX-NonMoose>
- CPAN Ratings

  <http://cpanratings.perl.org/d/MooseX-NonMoose>

AUTHOR

Jesse Luehrs <doy AT tozt DOT net>

COPYRIGHT AND LICENSE