NAME

MooseX::POE − The Illicit Love Child of Moose and POE

VERSION

version 0.214

SYNOPSIS

```
package Counter;
use MooseX::POE;

has count => (
    isa     => 'Int',
    is      => 'rw',
    lazy    => 1,
    default => sub { 0 },
);

sub START {
    my ($self) = @_;
    $self->yield('increment');
}

event increment => sub {
    my ($self) = @_;
    print "Count is now " . $self->count . "\n";
    $self->count( $self->count + 1 );
    $self->yield('increment') unless $self->count > 3;
};

no MooseX::POE;

Counter->new();
POE::Kernel->run();
```

or with MooseX::Declare:

```
class Counter {
    use MooseX::POE::SweetArgs qw(event);

    has count => (
        isa     => 'Int',
        is      => 'rw',
        lazy    => 1,
        default => sub { 0 },
    );

    sub START {
        my ($self) = @_;
        $self->yield('increment')
    }

    event increment => sub {
        my ($self) = @_;
        print "Count is now " . $self->count . "\n";
        $self->count( $self->count + 1 );
        $self->yield('increment') unless $self->count > 3;
    }
}

Counter->new();
POE::Kernel->run();
```

DESCRIPTION

MooseX::POE is a Moose wrapper around a POE::Session.

METHODS

event `$name` `$subref`

Create an event handler named `$name`.

get_session_id

Get the internal POE Session ID, this is useful to hand to other POE aware functions.

yield
call
delay
alarm
alarm_add
delay_add
alarm_set
alarm_adjust
alarm_remove
alarm_remove_all
delay_set
delay_adjust

A cheap alias for the same POE::Kernel function which will gurantee posting to the object's session.

STARTALL
STOPALL

KEYWORDS

METHODS

Default POE-related methods are provided by MooseX::POE::Meta::Trait::Object which is applied to your base class (which is usually Moose::Object) when you use this module. See that module for the documentation for. Below is a list of methods on that class so you know what to look for:

NOTES ON USAGE WITH MooseX::Declare

MooseX::Declare support is still ''experimental''. Meaning that I don't use it, I don't have any code that uses it, and thus I can't adequately say that it won't cause monkeys to fly out of any orifices on your body beyond what the tests and the SYNOPSIS cover.

That said there are a few caveats that have turned up during testing.

1. The `method` keyword doesn't seem to work as expected. This is an integration issue that is being resolved but I want to wait for MooseX::Declare to gain some more polish on their slurpy arguments.

2. MooseX::POE attempts to re-export Moose, which MooseX::Declare has already exported in a custom fashion. This means that you'll get a keyword clash between the features that MooseX::Declare handles for you and the features that Moose handles. To work around this you'll need to write:

```
use MooseX::POE qw(event);
# or
use MooseX::POE::SweetArgs qw(event);
# or
use MooseX::POE::Role qw(event);
```

to keep MooseX::POE from exporting the sugar that MooseX::Declare doesn't like. This is fixed in the Git version of MooseX::Declare but that version (as of this writing) is not on the CPAN.

SEE ALSO

AUTHORS

- Chris Prather <chris AT prather DOT org>
- Ash Berlin <ash AT cpan DOT org>
- Chris Williams <chris AT bingosnet DOT co DOT uk>
- Yuval (nothingmuch) Kogman
- Torsten Raudssus <torsten AT raudssus DOT de> <http://www.raudssus.de/>

COPYRIGHT AND LICENSE