NAME

MooseX::App::Simple − Single command applications

SYNOPSIS

```
package MyApp;
use MooseX::App::Simple qw(Config Color);

parameter 'param' => (
    is            => 'rw',
    isa           => 'Str',
    documentation => q[First parameter],
    required      => 1,
); # Positional parameter

option 'my_option' => (
    is            => 'rw',
    isa           => 'Bool',
    documentation => q[Enable this to do fancy stuff],
); # Option (--my_option)

has 'private' => (
    is              => 'rw',
); # not exposed

sub run {
    my ($self) = @_;
    # Do something
}
```

And then in some simple wrapper script:

```
#!/usr/bin/env perl
use MyApp;
MyApp->new_with_options->run;
```

DESCRIPTION

MooseX-App-Simple works basically just as MooseX::App, however it does not search for commands and assumes that you have all options and parameters defined in the current class.

Read the Tutorial for getting started with a simple MooseX::App command line application.

METHODS

new_with_options

```
my $myapp_command = MyApp->new_with_options();
```

This method reads the command line arguments from the user and tries to create instantiate the current class with the ARGV-input. If it fails it returns a MooseX::App::Message::Envelope object holding an error message.

You can pass a hash or hashref of default params to new_with_options

```
MyApp->new_with_options( %default );
```

Optionally you can pass a custom ARGV to this constructor

```
my $obj = MyApp->new_with_options( ARGV => \@myARGV );
```

However, if you do so you must take care of propper `@ARGV` encoding yourself.

OPTIONS

Same as in MooseX::App

PLUGINS

Same as in MooseX::App. However plugings adding commands (eg. version) will not work with MooseX::App::Simple.

SEE ALSO

Read the Tutorial for getting started with a simple MooseX::App command line application.

See MooseX::Getopt and MooX::Options for alternatives