Command line apps for the lazy and impatient

```
package MyMessage {
    use Moose;
    use Moose::Util::TypeConstraints qw();

    has 'recipient' => (
        is          => 'rw',
        required    => 1,
    );
    has 'message' => (
        is          => 'rw',
        required    => 1,
    );
    has 'level' => (
        is          => 'rw',
        isa         => Moose::Util::TypeConstraints::enum([qw(low medium high)]),
        default     => 'medium',
    );
    has 'verbose' => (
        is          => 'rw',
        isa         => 'Bool',
    );

    sub run { ... }
};
```

```
package MyMessage {
    use MooseX::App::Simple;
    use Moose::Util::TypeConstraints qw();

    hasparameter 'recipient' => (
        is          => 'rw',
        required    => 1,
    );
    hasoption 'message' => (
        is          => 'rw',
        required    => 1,
    );
    hasoption 'level' => (
        is          => 'rw',
        isa         => Moose::Util::TypeConstraints::enum([qw(low medium high)]),
        default     => 'medium',
    );
    hasoption 'verbose' => (
        is          => 'rw',
        isa         => 'Bool',
    );

    sub run { ... }
};
```

```
package MyMessage {
    use MooseX::App::Simple;
    use Moose::Util::TypeConstraints qw();

    parameter 'recipient' => (
        is            => 'rw',
        required      => 1,
        documentation => q[Recipient e-mail address],
    );
    option 'message' => (
        is            => 'rw',
        required      => 1,
        documentation => q[Message text],
    );
    option 'level' => (
        is            => 'rw',
        isa           => Moose::Util::TypeConstraints::enum([qw(low medium high)]),
        default       => 'medium',
        documentation => q[Message level],
    );
    option 'verbose' => (
        is            => 'rw',
        isa           => 'Bool',
        documentation => q[Be verbose],
    );

    sub run { ... }
};
```

```perl
#!/usr/bin/env perl
use strict;
use warnings;
use MyMessage;

MyMessage->new_with_options()->run;
```

```
$> my_message
Required parameter 'recipient' missing
Required option 'message' missing
usage:
    mymessage [recipient] [long options...]
    mymessage --help

parameters:
    recipient  Recipient e-mail address [Required]

options:
    --help -h --usage -?  Prints this usage information. [Flag]
    --level               Message level [Default:"medium"; Possible values:
                          low, medium, high]
    --message             Message text [Required]
    --verbose             Be verbose [Flag]
```

Command line apps for the not so lazy and impatient

- Add multiple sub-commands
- Add command documentation
- Colorised output
- Additonally read options from %ENV or config files
- Correct typos in user input (Did you mean …?)
- Prompt user for missing values
- Bash completion
- Display manpage, version and license
- Customise behaviour
- Handle list and key-value options
- Yada yada yada (or …)

```
package MyMessage {
    use MooseX::App::Simple qw(Color Config Term);
    use Moose::Util::TypeConstraints qw();

    app_strict(1);
    parameter 'recipient' => (
        […]
        cmd_env        => 'MYMESSAGE_RECIPIENT',
    );
    option 'message' => (
        […]
        cmd_term       => 1,
    );
    option 'verbose' => (
        […]
        cmd_aliases    => [‘v’],
    );

=head1 DESCRIPTION
This is how we roll
=cut
}
```

Demo