Command Line Adventures in Perl: 2021 Edition

Jason A. Crome

The Command Line? Again?

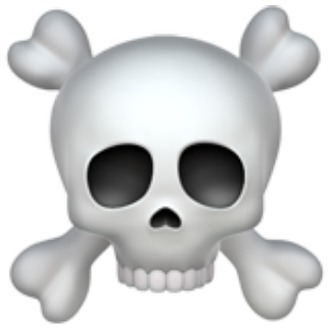
- Not the first time I've talked about the CLI
- Recent events put this in the forefront of my thoughts
- Starting with a trip down memory lane
- Evaluating current CLI options
- Finally, exploring my new favorite CLI toolkit

Let’s Go!

In the beginning…

- The CLI was without form
- And void
- And darkness was upon the face of Perl
- There was…

Getopt::Std



`Getopt::Long(::Descriptive)`

- Later attempts to build upon `Getopt::Std`
- A lot of CLI code build atop the latter specifically
- Brought sanity and much needed features to the command line
- Many other variants, `Getopt::*`

Command Line Frameworks

App::Cmd

- To my knowledge, first framework for building bona fide CLI apps
- OO, but before Moose
- Good balance between speed, stability, features
- This is what Dancer2 CLI was built on until recently

App::Cmd Problems

- Getting kinda crufty
- Now requires Perl 5.20+
- It's heavy

MAR 14, 2021

Distribution: App-Cmd
Module version: 0.333
Source (raw)
Browse (raw)
Changes
Homepage
How to Contribute
Repository (git clone)
Issues
Testers (580 / 0 / 12)
Kwalitee
87.27% Coverage
License: perl_5
Perl: v5.20.0

ACTIVITY

24 month

TOOLS

Download (48.49Kb)
MetaCPAN Explorer
Permissions
Subscribe to distribution
Install Instructions

Search distribution

grep distribution

Jump to version

Diff with version

PERMALINKS

This version
Latest version

15 non-PAUSE users.

RJBS
Ricardo SIGNES 😄
and 27 contributors
show them

DEPENDENCIES

Capture::Tiny
Carp
Class::Load
constant
Data::OptList
experimental
File::Basename
Getopt::Long
Getopt::Long::Descriptive
IO::TieCombine
Module::Pluggable::Object
parent
Pod::Usage
strict
String::RewritePrefix
Sub::Exporter
Sub::Exporter::Util
Sub::Install
Text::Abbrev
warnings

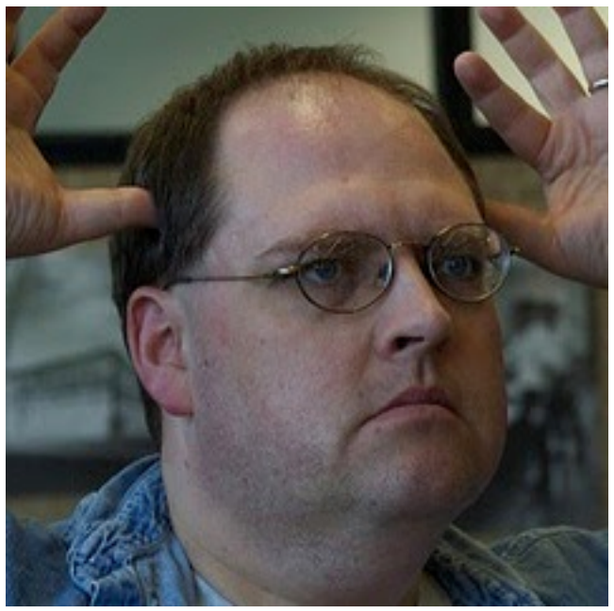
CPAN Testers List
Reverse dependencies
Dependency graph

NAME

App::Cmd - write command line apps with less suffering

The Contenders

MooseX::App



Your CLI, now with more Moose!

MooseX::App

- Makes it easy to create well structured, documented code
- If you know Moose, you know `MooseX::App`
- Anything you do with Moose, you can do in `MooseX::App`
- If you're using code that uses Moose, this costs you nothing extra
- Build a base class with common functionality
- Create roles that contain discrete units of functionality
- Create command classes that use roles, add new functionality

MooseX::App Drawbacks

- Moose? On the command line?
- Maximum overkill for many (read: most) situations

`MooX::Options`

- It's not Moose!
- Lightweight CLI framework built on Moo
- Much of the same `MooseX::App` goodness, but with less Moose
- Mandatory XS components

CLI::Osprey

- Hidden gem
- Lightweight, with few dependencies
- Most all the same goodness of MooseX::App (roles, types, etc.)
- Mostly compatible with MooX::App
- Parameters are attributes

Show me some code!

- This first example shows a whole command in a single file
- Not terribly interesting, but conveys some basics
- Shows how to write a single command with single option
- For codebases that have Moo, not a bad way to write small CLI utilities

hello-world.pl

```
#!/usr/bin/env perl

{
    package HelloWorld;

    use Moo;
    use CLI::Osprey;

    option 'message' ⇒ (
        is      ⇒ 'ro',
        format  ⇒ 's',
        doc     ⇒ 'The message to display',
        default ⇒ 'Hello, world!',
    );

    sub run {
        my( $self ) = @_;
        print $self→message, "\n";
    }
}

use HelloWorld;
HelloWorld→new_with_options→run;
```

We can do better!

- Next example shows multiple subcommands
- Subcommands can be defined inline, or in another module
- Will demonstrate additional option config parameters
- Will show how to access parent command
- Parent command is a great place to put things used by all subcommands (DB schema, config, log engine, etc.)

Putting everything together

- Final example pulls in more advanced Moo concepts
- Types
- Roles
- Lifecycle methods
- Everything from the prior examples

The Bad

- Lacks a few features other frameworks have (repeatable options, config files to name a few)
- Not widely used (therefore, not widely tested)
- Docs are a bit buggy
- Everything needs a `run()` method, even if it does nothing

Questions?

Thank you!