Emacs for Perl Sources

A few small tricks

Harald Jörg
Prior Art

- Emacs, a performant IDE for Perl
  - Laurent Dami, YAPC::EU::2009 in Lisbon
So, a refresher might be just fine, because....

- The times, they are a changing...
  - Bob Dylan, 1964
Emacs is...

- "the extensible self-documenting text editor"
- ... well, more sort of a desktop environment.
  - Project planner (org-mode)
  - Mail/News/RSS reader (gnus)
  - Support for version control systems
  - Emacs Lisp for customizing
  - ... and, last but not least, a text editor!
- First public release in 1985
- Current version: 26.1
Perl Support in Emacs

- CPerl mode
- "IDE"-like stuff:
  - 2001: Emacs-EPL
  - 2008: Emacs::PDE
  - 2010: perlnow
  - 2011: Sepia - Simple Emacs Perl Interface
  - 2016: plsense - haven't tried this one yet
  - 2018: Devel::PerlySense
Reading Perl: Devel::Perl::PerlySense

- Highlight: "Class Overview"
- Shows project and system libs
- Understands (most) Moose syntax
- Run tests / scripts and grab results
- Uses PPI to analyze source code
- Has quite some dependencies
package main;

use Test::More;

use Sphere;

my $sphere;

isa_ok ($sphere = Sphere->new
    {coordinates => [2,2,2],
     radius     => 1,
     color      => [0.5,0.5,0.5],
    },
    'Sphere',
    'Create a sphere',
);

is ($sphere->red,0.5,
    "Sphere's color is a dull grey",
);
Inheritance
[<main>]

API

Bookmarks

Uses
[Sphere] [Test::More]
Inheritance
[ Vector ]
[ Object ]
[ <Sphere> ] --> [ Color ]

API
\>color \>_coordinates \>move(\#vector) \>red \>x()

Bookmarks

Uses
[ Moose ] [ MooseX::SemiAffordanceAccessor ]
=head2 Method move(@vector)

This method is provided for consumption by objects. It adds the vector given to the coordinates, thereby moving the object by the same amount.

=cut

=head3 Sub move

my $self = shift;
my (@vector) = @_;

 @{$self->{_coordinates}} = map {
    $_ + shift @vector
} @{$self->{_coordinates}};

return $self;

=cut
Writing Perl: CPerl Mode

- Included with Emacs...
- ...is a release from 2009 (Emacs 26.1)
- But this doesn't hurt too much.

Features include:

- Syntax highlighting...
  ...but not for Moose keywords
- Control construct templates...
  ...but not for Moose boilerplate
Trick: Extending CPerl Mode

- I'm an absolute beginner with Emacs Lisp... ...yet it turned out to be remarkably easy to extend CPerl mode.
- I want to type...
  - "has" and get an attribute boilerplate
  - "method" and get a sub boilerplate
A Simple Moose Mode

- Emacs helpers: abbrev-mode and skeletons
- Add the templates you want to be expanded
- Define a moose mode derived from CPerl-mode
- Map shortcuts to the templates
- Steal some magic from Emacs wiki
- Done!
sub move {
    my $self = shift;
    my (@vector) = @_;

    @{$self->{_coordinates}} = map {
        $_ + shift @vector
    } @{$self->{_coordinates}};

    return $self;
}

method

__END__

=head1 SYNOPSIS

---
sub move {
    my $self = shift;
    my (@vector) = @_;

    @{$self->{_coordinates}} = map { 
        $_ + shift @vector 
    } @{$self->{_coordinates}};

    return $self;
}

=head2 Method

1;

__END__

=head1 SYNOPSIS

-::- Vector.pm  75% L65  Git-master  (Moose Flymake Abbrev)
Sub name:
=head2 Method moveto

=cut

sub moveto {
  my $self = shift;
  my () = $_[0];

  __END__

=head1 SYNOPSIS

-:::* Vector.pm  83% L67  Git-master  (Moose Flymake Abbrev)
Auto-saving...done
Debugging Perl: perldb

- **Pro**
  - Examine data structures of any size
  - Current source visible in editor window
  - 1 line change to make it work with Ovids::Debugger

- **Contra**
  - No separate buffers after a fork
package main;

use Test::More;

use Sphere;

my $sphere;

isa_ok ($sphere := Sphere->new (center => [2, 2, 2],
  radius => 1,
  color => [0.5, 0.5, 0.5],
  ) ), 'Sphere',

'Create a sphere',

);;

is ($sphere->red, 0.5, "Sphere's color is a dull grey",

);;

--- create.t  Top L17  Git-master  (CPerl Flymake)
Run perldb (like this): perl -d -I../lib/home/haj/devel/oddd/t/create.t
Loading DB routines from perl5db.pl version 1.49_05
Editor support enabled.

Enter h or 'h h' for help, or 'man perlddebug' for more help.

```
DB<1>  n
DB<1>  n
ok 1 - 'Create a sphere' isa 'Sphere'
DB<1>  
```

```
U:**-  *gud-create.t*  Bot L30  (Debugger:run)
13    ....... 'Sphere',$
14    ....... 'Create a sphere',$
15   );$
16$
17    is ($sphere->red, 0.5,$
18    ....  'Sphere's color is a dull grey',$
19   );$
20$
21    is ($sphere->x, 2,$
```

--- create.t  53% L17  Git-master (CPerl Flymake)
Using Perl within Emacs

- shell-command-on-region
  - Not exactly Perl-specific
  - Example: Run perltidy on a block
  - Example: Fix double UTF-8 encoding
- Code-Blocks in Org mode
  - Not exactly Perl-specific
  - ...but can run Perl blocks
(defun decode-utf8-in-region (from to)
  "UTF8-decodes the current region in-place"
  (interactive "r")
  (shell-command-on-region from to
   "perl -CS -p -e 'utf8::decode($_)'"
   (current-buffer) t
   )
  )
)

(global-set-key (kbd "<f12>") 'decode-utf8-in-region)
Code block example

```bash
# BEGIN_SRC sh
sudo:postgres@localhost: /results raw replace
sql="CREATE ROLE \"dbd-pg-test\"
LOGIN ENCRYPTED PASSWORD 'md58db4465b0bb3c51e8e29448469ae0da7'
CREATEDB
VALID UNTIL '9999-12-31 00:00:00';
COMMENT ON ROLE \"dbd-pg-test\"
IS 'For tests using the Perl connector DBD::Pg';"
echo $sql | psql
dbsql="CREATE DATABASE \"dbd-pg-test\"
    WITH OWNER = \"dbd-pg-test\"
    ENCODING = 'UTF8'
    CONNECTION LIMIT = -1;"
echo $dbsql | psql
# END_SRC```
```
Thanks for listening!
Questions?