

## NAME

Pod::Simple::Methody -- turn Pod::Simple events into method calls

## SYNOPSIS

```
require 5;
use strict;
package SomePodFormatter;
use base qw(Pod::Simple::Methody);

sub handle_text {
    my($self, $text) = @_;
    ...
}

sub start_head1 {
    my($self, $attrs) = @_;
    ...
}
sub end_head1 {
    my($self) = @_;
    ...
}
```

...and start/\_end\_ methods for whatever other events you want to catch.

## DESCRIPTION

This class is of interest to people writing Pod formatters based on Pod::Simple.

This class (which is very small -- read the source) overrides Pod::Simple's `_handle_element_start`, `_handle_text`, and `_handle_element_end` methods so that parser events are turned into method calls. (Otherwise, this is a subclass of *Pod::Simple* and inherits all its methods.)

You can use this class as the base class for a Pod formatter/processor.

## METHOD CALLING

When Pod::Simple sees a `"=head1 Hi there"`, for example, it basically does this:

```
$parser->_handle_element_start( "head1", \%attributes );
$parser->_handle_text( "Hi there" );
$parser->_handle_element_end( "head1" );
```

But if you subclass Pod::Simple::Methody, it will instead do this when it sees a `"=head1 Hi there"`:

```
$parser->start_head1( \%attributes ) if $parser->can('start_head1');
$parser->handle_text( "Hi there" )   if $parser->can('handle_text');
$parser->end_head1( )                 if $parser->can('end_head1');
```

If Pod::Simple sends an event where the element name has a dash, period, or colon, the corresponding method name will have an underscore in its place. For example, `"foo.bar:baz"` becomes `start_foo_bar_baz` and `end_foo_bar_baz`.

See the source for Pod::Simple::Text for an example of using this class.

## SEE ALSO

*Pod::Simple*, *Pod::Simple::Subclassing*

**COPYRIGHT AND DISCLAIMERS**

Copyright (c) 2002 Sean M. Burke. All rights reserved.

This library is free software; you can redistribute it and/or modify it under the same terms as Perl itself.

This program is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose.

**AUTHOR**

Sean M. Burke [sburke@cpan.org](mailto:sburke@cpan.org)