NAME

Pod::ParseUtils - helpers for POD parsing and conversion

SYNOPSIS

```
use Pod::ParseUtils;

my $list = new Pod::List;
my $link = Pod::Hyperlink->new('Pod::Parser');
```

DESCRIPTION

Pod::ParseUtils contains a few object-oriented helper packages for POD parsing and processing (i.e. in POD formatters and translators).

Pod::List

Pod::List can be used to hold information about POD lists (written as =over ... =item ... =back) for further processing. The following methods are available:

Pod::List->new()

Create a new list object. Properties may be specified through a hash reference like this:

```
my $list = Pod::List->new({ -start => $., -indent => 4 });
```

See the individual methods/properties for details.

$list->file()

Without argument, retrieves the file name the list is in. This must have been set before by either specifying **-file** in the **new()** method or by calling the **file()** method with a scalar argument.

$list->start()

Without argument, retrieves the line number where the list started. This must have been set before by either specifying **-start** in the **new()** method or by calling the **start()** method with a scalar argument.

$list->indent()

Without argument, retrieves the indent level of the list as specified in `=over n`. This must have been set before by either specifying **-indent** in the **new()** method or by calling the **indent()** method with a scalar argument.

$list->type()

Without argument, retrieves the list type, which can be an arbitrary value, e.g. `OL`, `UL`, ... when thinking the HTML way. This must have been set before by either specifying **-type** in the **new()** method or by calling the **type()** method with a scalar argument.

$list->rx()

Without argument, retrieves a regular expression for simplifying the individual item strings once the list type has been determined. Usage: E.g. when converting to HTML, one might strip the leading number in an ordered list as `<OL>` already prints numbers itself. This must have been set before by either specifying **-rx** in the **new()** method or by calling the **rx()** method with a scalar argument.

$list->item()

Without argument, retrieves the array of the items in this list. The items may be represented by any scalar. If an argument has been given, it is pushed on the list of items.

$list->parent()

Without argument, retrieves information about the parent holding this list, which is represented as an arbitrary scalar. This must have been set before by either specifying **-parent** in the **new()** method or by calling the **parent()** method with a scalar argument.

$list->tag()

Without argument, retrieves information about the list tag, which can be any scalar. This must have been set before by either specifying **-tag** in the **new()** method or by calling the **tag()** method with a scalar argument.

Pod::Hyperlink

Pod::Hyperlink is a class for manipulation of POD hyperlinks. Usage:

```
  my $link = Pod::Hyperlink->new('alternative text|page/"section in
page"');
```

The **Pod::Hyperlink** class is mainly designed to parse the contents of the `L<...>` sequence, providing a simple interface for accessing the different parts of a POD hyperlink for further processing. It can also be used to construct hyperlinks.

Pod::Hyperlink->new()

The **new()** method can either be passed a set of key/value pairs or a single scalar value, namely the contents of a `L<...>` sequence. An object of the class `Pod::Hyperlink` is returned. The value `undef` indicates a failure, the error message is stored in `$@`.

$link->parse($string)

This method can be used to (re)parse a (new) hyperlink, i.e. the contents of a `L<...>` sequence. The result is stored in the current object. Warnings are stored in the **warnings** property. E.g. sections like `L<open(2)>` are deprecated, as they do not point to Perl documents. `L<DBI::foo(3p)>` is wrong as well, the manpage section can simply be dropped.

$link->markup($string)

Set/retrieve the textual value of the link. This string contains special markers `P<>` and `Q<>` that should be expanded by the translator's interior sequence expansion engine to the formatter-specific code to highlight/activate the hyperlink. The details have to be implemented in the translator.

$link->text()

This method returns the textual representation of the hyperlink as above, but without markers (read only). Depending on the link type this is one of the following alternatives (the + and * denote the portions of the text that are marked up):

```
  +perl+                    L<perl>
  *$|* in +perlvar+         L<perlvar/$|>
  *OPTIONS* in +perldoc+    L<perldoc/"OPTIONS">
  *DESCRIPTION*             L<"DESCRIPTION">
```

$link->warning()

After parsing, this method returns any warnings encountered during the parsing process.

$link->file()

$link->line()

Just simple slots for storing information about the line and the file the link was encountered in. Has to be filled in manually.

$link->page()

This method sets or returns the POD page this link points to.

$link->node()

As above, but the destination node text of the link.

$link->alttext()

Sets or returns an alternative text specified in the link.

$link->type()

The node type, either `section` or `item`. As an unofficial type, there is also `hyperlink`, derived from e.g. `L<http://perl.com>`

$link->link()

Returns the link as contents of `L<>`. Reciprocal to **parse()**.

Pod::Cache

Pod::Cache holds information about a set of POD documents, especially the nodes for hyperlinks. The following methods are available:

Pod::Cache->new()

Create a new cache object. This object can hold an arbitrary number of POD documents of class Pod::Cache::Item.

$cache->item()

Add a new item to the cache. Without arguments, this method returns a list of all cache elements.

$cache->find_page($name)

Look for a POD document named `$name` in the cache. Returns the reference to the corresponding Pod::Cache::Item object or undef if not found.

Pod::Cache::Item

Pod::Cache::Item holds information about individual POD documents, that can be grouped in a Pod::Cache object. It is intended to hold information about the hyperlink nodes of POD documents. The following methods are available:

Pod::Cache::Item->new()

Create a new object.

$cacheitem->page()

Set/retrieve the POD document name (e.g. "Pod::Parser").

$cacheitem->description()

Set/retrieve the POD short description as found in the `=head1 NAME` section.

$cacheitem->path()

Set/retrieve the POD file storage path.

$cacheitem->file()

Set/retrieve the POD file name.

$cacheitem->nodes()

Add a node (or a list of nodes) to the document's node list. Note that the order is kept, i.e. start with the first node and end with the last. If no argument is given, the current list of nodes is returned in the same order the nodes have been added. A node can be any scalar, but usually is a pair of node string and unique id for the `find_node` method to work correctly.

$cacheitem->find_node($name)

Look for a node or index entry named `$name` in the object. Returns the unique id of the node (i.e. the second element of the array stored in the node array) or undef if not found.

$cacheitem->idx()

Add an index entry (or a list of them) to the document's index list. Note that the order is kept, i.e. start with the first node and end with the last. If no argument is given, the current list of index entries is returned in the same order the entries have been added. An index entry can be any scalar, but usually is a pair of string and unique id.

AUTHOR

Please report bugs using *http://rt.cpan.org*.

Marek Rouchal <marekr@cpan.org>, borrowing a lot of things from *pod2man* and *pod2roff* as well as other POD processing tools by Tom Christiansen, Brad Appleton and Russ Allbery.

SEE ALSO

pod2man, *pod2roff*, *Pod::Parser*, *Pod::Checker*, *pod2html*