Pod::2::DocBook

Jozef Kutej

Pod::2::DocBook

by Jozef Kutej

Table of Contents

Chapter 1. Introduction

Overview

Note

This document refers to Pod::2::DocBook [http://search.cpan.org/dist/Pod-2-DocBook/] version 0.01 [http://search.cpan.org/~jkutej/Pod-2-DocBook-0.01/].

Pod::2::DocBook Perl module helps to convert existing pod documentation to docbook xml format. For a full reasoning and detail description see Chapter 3, *Developers documentation section*.

Warning

not complete, as any other documentation ;-)

Chapter 2. Usage

Instalation

Installing of Pod::2::DocBook should be as easy and streight forward as executing `cpan -i Pod::2::DocBook`. This will add pod2docbook script to the system. See the section called “Scripts” for description.

DocBook quick start guide

In folder `examples/pod2docbook-docbook/` of Pod::2::DocBook [http://search.cpan.org/dist/Pod-2-DocBook/] is the source of this document. To generate the HTML or PDF version from it you'll need: make, xmllint, xsltproc, fop and docbook xml+xsl.

For Debian: **apt-get install docbook-utils psutils docbook-xml docbook-xsl xsltproc fop make**. To have images in the PDF you need to get jimi [http://java.sun.com/products/jimi/] and copy `jimi.jar` to `/usr/share/java/jimi-1.0.jar`

Then just do **make** or **make Pod-2-DocBook.html** or **make Pod-2-DocBook.pdf** to generate this "book".

Chapter 3. Developers documentation section

Overview

It's aways nice to read some documentation but to write it's not so fun. And to write it and sync it on two different places it's just annoing. To make our life easy let's see how a docbook sections generated directly for Perl module can look like.

Modules

Pod-2-DocBook POD

NAME

Pod::2::DocBook - Convert Pod data to DocBook SGML

SYNOPSIS

```
use Pod::2::DocBook;
my $parser = Pod::2::DocBook->new (title            => 'My Article',
                                   doctype          => 'article',
                fix_double_quotes => 1,
                spaces           => 3);

$parser->parse_from_file ('my_article.pod', 'my_article.sgml');
```

DESCRIPTION

Pod::2::DocBook is a module for translating Pod-formatted documents to DocBook 4.2 SGML (see http://www.docbook.org/). It is primarily a back end for **pod2docbook**, but, as a Pod::Parser subclass, it can be used on its own. The only public extensions to the Pod::Parser interface are options available to `new()`:

doctype	This option sets the output document's doctype. The currently supported types are **article**, **chapter**, **refentry** and **section**. Special processing is performed when the doctype is set to **refentry** (see “Document Types”). You *must* set this option in order to get valid DocBook output.
fix_double_quotes	If this option is set to a true value, pairs of double quote characters ("") in ordinary paragraphs will be replaced with <**quote**> and </**quote**>. See “Ordinary Paragraphs” for details.
header	If this option is set to a true value, Pod::2::DocBook will emit a DOCTYPE as the first line of output.
spaces	Pod::2::DocBook produces pretty-printed output. This option sets the number of spaces per level of indentation in the output.
title	This option sets the output document's title.

The rest of this document only describes issues specific to Pod::2::DocBook; for details on invoking the parser, specifically the `new()`, `parse_from_file()` and `parse_from_filehandle()` methods, see Pod::Parser.

METHODS

```
our @ISA     = qw(Pod::Parser);
```

initialize()

Initialize parser.

begin_pod()

Output docbook header stuff.

end_pod()

Output docbook footer. Will print also errors if any in a comment block.

commans($command, $paragraph, $line_num)

Process POD commands.

textblock ($paragraph, $line_num)

Process text block.

verbatim($paragraph, $line_num)

Process verbatim text block.

interior_sequence($command, $argument, $seq)

Process formatting commands.

error_msg

Returns parser error message(s) if any occured.

POD TO DOCBOOK TRANSLATION

Pod is a deceptively simple format; it is easy to learn and very straightforward to use, but it is suprisingly expressive. Nevertheless, it is not nearly as expressive or complex as DocBook. In most cases, given some Pod, the analogous DocBook markup is obvious, but not always. This section describes how Pod::2::DocBook treats Pod input so that Pod authors may make informed choices. In every case, Pod::2::DocBook strives to make easy things easy and hard things possible.

The primary motivation behind Pod::2::DocBook is to facilitate single-source publishing. That is, you should be able to generate man pages, web pages, PDF and PostScript documents, or any other format your SGML and/or Pod tools can produce, from the same Pod source, without the need for hand-editing any intermediate files. This may not always be possible, or you may simply choose to render Pod to DocBook and use that as your single source. To satisfy the first requirement, Pod::2::DocBook always processes the entire Pod source and tries very hard to produce valid DocBook markup, even in the presence of malformed Pod (see “DIAGNOSTICS”). To satisfy the second requirement (and to be a little nifty), Pod::2::DocBook pretty-prints its output. If you're curious about what specific output to expect, read on.

Document Types

DocBook's structure is very modular; many of its document types can be embedded directly into other documents. Accordingly, Pod::2::DocBook will generate four different document types: **article**, **chapter**, **refentry**, and **section**. This makes it easy, for instance, to write all the chapters of a book in separate Pod documents, translate them into DocBook markup and later glue them together before processing the entire book. You could do the same with each section in an article, or you could write the entire article in a single Pod document. Other document types, such as **book** and **set**,

do not map easily from Pod, because they require structure for which there is no Pod equivalent. But given sections and chapters, making larger documents becomes much simpler.

The **refentry** document type is a little different from the others. Sections, articles, and chapters are essentially composed of nested sections. But a refentry has specialized elements for the *NAME* and *SYNOPSIS* sections. To accommodate this, Pod::2::DocBook performs extra processing on the Pod source when the **doctype** is set to **refentry**. You probably don't have to do anything to your document to assist the processing; typical man page conventions cover the requirements. Just make sure that the *NAME* and *SYNOPSIS* headers are both **=head1**s, that "NAME" and "SYNOPSIS" are both uppercase, and that **=head1 NAME** is the first line of Pod source.

Ordinary Paragraphs

Ordinary paragraphs in a Pod document translate naturally to DocBook paragraphs. Specifically, after any formatting codes are processed, the characters `<`, `>` and `&` are translated to their respective SGML character entities, and the paragraph is wrapped in <**para**> and </**para**>.

For example, given this Pod paragraph:

```
Here is some text with I<italics> & an ampersand.
```

Pod::2::DocBook would produce DocBook markup similar to this:

```
<para>
  Here is some text with <emphasis role="italic">italics</emphasis>
  &amp; an ampersand.
</para>
```

Depending on your final output format, you may sometimes want double quotes in ordinary paragraphs to show up ultimately as "smart quotes" (little 66s and 99s). Pod::2::DocBook offers a convenient mechanism for handling double quotes in ordinary paragraphs and letting your SGML toolchain manage their presentation: the **fix_double_quotes** option to `new()`. If this option is set to a true value, Pod::2::DocBook will replace pairs of double quotes in ordinary paragraphs (and *only* in ordinary paragraphs) with <**quote**> and </**quote**>.

For example, given this Pod paragraph:

```
Here is some text with I<italics> & an "ampersand".
```

Pod::2::DocBook, with **fix_double_quotes** set, would produce DocBook markup similar to this:

```
<para>
  Here is some text with <emphasis role="italic">italics</emphasis>
  &amp; an <quote>ampersand</quote>.
</para>
```

If you have a paragraph with an odd number of double quotes, the last one will be left untouched, which may or may not be what you want. If you have such a document, replace the unpaired double quote character with **E<quot>**, and Pod::2::DocBook should be able to give you the output you expect. Also, if you have any **=begin docbook** ... **=end docbook** regions (see “Embedded DocBook Markup”) in your Pod, you are responsible for managing your own quotes in those regions.

Verbatim Paragraphs

Verbatim paragraphs translate even more naturally; perlpodspec mandates that absolutely no processing should be performed on them. So Pod::2::DocBook simply marks them as CDATA and wraps them in <**screen**> and </**screen**>. They are not indented the way ordinary paragraphs are, because they treat whitespace as significant.

For example, given this verbatim paragraph (imagine there's leading whitespace in the source):

```
my $i = 10;
```

```
while (<> && $i--) {
    print "$i: $_";
}
```

Pod::2::DocBook would produce DocBook markup similar to this:

```
<screen><![CDATA[my $i = 10;
while (<> && $i--) {
    print "$i: $_";
}]] ></screen>
```

Multiple contiguous verbatim paragraphs are treated as a single *screen* element, with blank lines separating the paragraphs, as dictated by perlpodspec.

Command Paragraphs

```
=head1 Heading Text
=head2 Heading Text
=head3 Heading Text
=head4 Heading Text
```

All of the Pod heading commands produce DocBook *section* elements, with the heading text as titles. Pod::2::DocBook (perlpod) only allows for 4 heading levels, but DocBook allows arbitrary nesting; see “Embedded DocBook Markup” if you need more than 4 levels. Pod::2::DocBook only looks at relative heading levels to determine nesting. For example, this bit of Pod:

```
=head1 1

Contents of section 1

=head2 1.1

Contents of section 1.1
```

and this bit of Pod:

```
=head1 1

Contents of section 1

=head3 1.1

Contents of section 1.1
```

both produce the same DocBook markup, which will look something like this:

```
<section id="article-My-Article-1"><title>1</title>
  <para>
    Contents of section 1
  </para>
  <section id="article-My-Article-1-1"><title>1.1</titl
    <para>
      Contents of section 1.1
    </para>
  </section>
</section>
```

Note that Pod::2::DocBook automatically generates section

identifiers from your doctype, document title and section title. It does the same when you make internal links (see “Formatting Codes”, ensuring that if you supply the same link text as you did for the section title, the resulting identifiers will be the same.

```
=over indentlevel
=item stuff...
=back
```

`=over` ... `=back` regions are somewhat complex, in that they can lead to a variety of DocBook constructs. In every case, *indentlevel* is ignored by Pod::2::DocBook, since that's best left to your stylesheets.

An `=over` ... `=back` region with no `=item`s represents indented text and maps directly to a DocBook *blockquote* element. Given this source:

```
=over 4

This text should be indented.

=back
```

Pod::2::DocBook will produce DocBook markup similar to this:

```
<blockquote>
  <para>
    This text should be indented.
  </para>
</blockquote>
```

Inside an `=over` ... `=back` region, `=item` commands generate lists. The text that follows the first `=item` determines the type of list that will be output:

- "*" (an asterisk) produces <**itemizedlist**>
- "1" or "1." produces <**orderedlist numeration="arabic"**>
- "a" or "a." produces <**orderedlist numeration="loweralpha"**>
- "A" or "A." produces <**orderedlist numeration="upperalpha"**>
- "i" or "i." produces <**orderedlist numeration="lowerroman"**>
- "I" or "I." produces <**orderedlist numeration="upperroman"**>
- anything else produces <**variablelist**>

Since the output from each of these is relatively verbose, the best way to see examples is to actually render some Pod into DocBook.

```
=pod
=cut
```

Pod::Parser recognizes these commands, and, therefore, so does Pod::2::DocBook, but they don't produce any output.

`=begin formatname`	
`=end formatname`	
`=for formatname text...`	Pod::2::DocBook supports two formats: **docbook**, explained in "Embedded DocBook Markup", and **table**, explained in "Simple Tables".
`=encoding encodingname`	This command is currently not supported. If Pod::2::DocBook encounters a document that contains `=encoding`, it will ignore the command and report an error ("unknown command `%s' at line %d in file %s").

Embedded DocBook Markup

There are a wide range of DocBook structures for which there is no Pod equivalent. For these, you will have to provide your own markup using **=begin docbook** ... **=end docbook** or **=for docbook ...**. Pod::2::DocBook will directly output whatever text you provide, unprocessed, so it's up to you to ensure that it's valid DocBook.

Images, footnotes and many inline elements are obvious candidates for embedded markup. Another possible use is nesting sections more than four-deep. For example, given this source:

```
=head1  1

This is Section 1

=head2 1.1

This is Section 1.1

=head3 1.1.1

This is Section 1.1.1

=head4 1.1.1.1

This is Section 1.1.1.1

=begin docbook

<section>
<title>1.1.1.1.1</title>
<para>This is Section 1.1.1.1.1</para>
</section>

=end docbook
```

Pod::2::DocBook will generate DocBook markup similar to this:

```
  <section id="article-My-Article-1"><title>1</title>
    <para>
      This is Section 1
    </para>
    <section id="article-My-Article-1-1"><title>1.1</title>
      <para>
    This is Section 1.1
      </para>
      <section id="article-My-Article-1-1-1"><title>1.1.1</title>
        <para>
      This is Section 1.1.1
        </para>
        <section id="article-My-Article-1-1-1-1"><title>1.1.1.1</title>
          <para>
        This is Section 1.1.1.1
```

```
          </para>
<section>
<title>1.1.1.1.1</title>
<para>This is Section 1.1.1.1.1</para>
</section>
        </section>
      </section>
    </section>
  </section>
```

Simple Tables

Pod::2::DocBook also provides a mechanism for generating basic tables with **=begin table** and **=end docbook**. If you have simple tabular data or a CSV file exported from some application, Pod::2::DocBook makes it easy to generate a table from your data. The syntax is intended to be simple, so DocBook's entire table feature set is not represented, but even if you do need more complex table markup than Pod::2::DocBook produces, you can rapidly produce some markup which you can hand-edit and then embed directly in your Pod with **=begin docbook** ... **=end docbook**. Each table definition spans multiple lines, so there is no equivalent **=for table** command.

The first line of a table definition gives the table's title. The second line gives a list of comma-separated column specifications (really just column alignments), each of which can be **left**, **center** or **right**. The third line is a list of comma-separated column headings, and every subsequent line consists of comma-separated row data. If any of your data actually contain commas, you can enclose them in double quotes; if they also contain double quotes, you must escape the inner quotes with backslashes (typical CSV stuff).

Here's an example:

```
=begin table

Sample Table
left,center,right
Powers of Ten,Planets,Dollars
10,Earth,$1
100,Mercury,$5
1000,Mars,$10
10000,Venus,$20
100000,"Jupiter, Saturn",$50

=end table
```

And here's what Pod::2::DocBook would do with it:

```
<table>
  <title>Sample Table</title>
  <tgroup cols="3">
    <colspec align="left">
    <colspec align="center">
    <colspec align="right">
    <thead>
      <row>
        <entry>Powers of Ten</entry>
        <entry>Planets</entry>
        <entry>Dollars</entry>
      </row>
    </thead>
    <tbody>
      <row>
        <entry>10</entry>
        <entry>Earth</entry>
        <entry>$1</entry>
      </row>
      <row>
        <entry>100</entry>
```

```
        <entry>Mercury</entry>
        <entry>$5</entry>
      </row>
      <row>
        <entry>1000</entry>
        <entry>Mars</entry>
        <entry>$10</entry>
      </row>
      <row>
        <entry>10000</entry>
        <entry>Venus</entry>
        <entry>$20</entry>
      </row>
      <row>
        <entry>100000</entry>
        <entry>Jupiter, Saturn</entry>
        <entry>$50</entry>
      </row>
    </tbody>
  </tgroup>
</table>
```

Formatting Codes

Pod formatting codes render directly into DocBook as inline elements:

- I<text>

  ```
  <emphasis role="italic">text</emphasis>
  ```

- B<text>

  ```
  <emphasis role="bold">text</emphasis>
  ```

- C<code>

  ```
  <literal role="code"><![CDATA[code]] ></literal>
  ```

- L<name>

  ```
  <citerefentry><refentrytitle>name</refentrytitle></citerefentry>
  ```

- L<name(n)>

  ```
  <citerefentry><refentrytitle>name</refentrytitle>
  <manvolnum>n</manvolnum></citerefentry>
  ```

- L<name/"sec"> or L<name/sec>

  ```
  <quote>sec</quote> in <citerefentry>
  <refentrytitle>name</refentrytitle></citerefentry>
  ```

- L<name(n)/"sec"> or L<name(n)/sec>

  ```
  <quote>sec</quote> in <citerefentry>
  <refentrytitle>name</refentrytitle><manvolnum>n</manvolnum>
  </citerefentry>
  ```

- L</"sec"> or L</sec> or L<"sec">

  <link linkend="article-My-Article-sec"><quote>sec</quote></link>

- L<text|name>

  text

- L<text|name/"sec"> or L<text|name/sec>

  text

- L<text|/"sec"> or L<text|/sec> or L<text|"sec">

  <link linkend="article-My-Article-sec"><quote>text</quote></link>

- L<scheme:...>

  <ulink url="scheme:...">scheme:...</ulink>

- E<verbar>

  |

- E<sol>

  /

- E<number>

  &#number;

- any other E<escape>

  &escape;

- F<filename>

  <filename>filename</filename>

- S<text with spaces>

  text with spaces

- X<topic name>

  <indexterm><primary>topic name</primary></indexterm>

DIAGNOSTICS

Pod::2::DocBook makes every possible effort to produce valid DocBook markup, even with malformed POD source. Any processing errors will be noted in comments at the end of the output document. Even when errors occur, Pod::2::DocBook always reads the entire input document and never exits with a non-zero status.

unknown command `%s' at line %d in file %s	See “Command Paragraph” in perlpod for a list of valid commands. The command referenced in the error message was ignored.
formatting code `%s' nested within `%s' at line %d in file %s	See “Formatting Codes” in perlpod for details on which formatting codes can be nested. The offending code was translated into the output document as the raw text inside its angle brackets.
unknown formatting code `%s' at line in file %s	The input contained a formatting code not listed in perlpod; it was translated into the output document as the raw text inside the angle brackets.
empty L<> at line %d in file %s	Self-explanatory.
invalid escape `%s' at line %d in file %s	Self-explanatory; it was translated into the output document as the raw text inside the angle brackets.
=item must be inside an =over ... =back section at line %d in file %s	Self-explanatory. The `=item' referenced in the error was ignored.
`=end %s' found but current region opened with `=begin %s'	The closest `=end' command to the referenced `=begin' didn't match; processing continued as if the mismatched `=end' wasn't there.
no matching `=end' for `=begin %s'	Pod::2::DocBook reached the end of its input without finding an `=end' command to match the `=begin' referenced in the error; end-of-file processing continued.
unknown colspec `%s' in table at line %d in file %s	See “Simple Tables” for a list of supported column specifications.
encountered unknown state `%s' (this should never happen)	The state referred to is an internal variable used to properly manage nested DocBook structures. You should indeed never see this message, but if you do, you should contact the module's author.

SEE ALSO

pod2docbook, perlpod, Pod::DocBook, SVN repo - https://cle.sk/repos/pub/cpan/Pod-2-DocBook/, http://www.ohloh.net/projects/pod-2-docbook, `doc/` + `examples/pod2docbook-docbook/` for Pod::2::DocBook DocBook documentation

DocBook related links: http://www.docbook.org/, http://www.sagehill.net/docbookxsl/, http://developers.cogentrts.com/cogent/prepdoc/pd-axfrequentlyuseddocbooktags.html

AUTHOR

Alligator Descartes <descarte@symbolstone.org> wrote a module called Pod::2::DocBook, which was later maintained by Jan Iven <jan.iven@cern.ch>. That module was based on the original pod2html by Tom Christiansen <tchrist@mox.perl.com>.

Nandu Shah <nandu@zvolve.com> wrote Pod::DocBook, which is unrelated to the previous module (even though they both perform the same function). (http://search.cpan.org/~nandu/Pod-DocBook-1.2/)

Jozef Kutej <jkutej@cpan.org> renamed the module to Pod::2::DocBook because Nandus version was buried in the CPAN archive as an "UNAUTHORIZED RELEASE".

COPYRIGHT

Scripts

pod2docbook script

NAME

pod2docbook - Convert POD data to DocBook SGML

SYNOPSIS

pod2docbook [**--help**] [**--doctype**=**article**|**chapter**|**section**|**refentry**] [**--title**=*title*] [**--spaces**=*# spaces per indent level*] [**--fix-double-quotes**] [**--no-header**] [*infile* [*outfile*]]

DESCRIPTION

pod2docbook converts files from pod format (see perlpod) to DocBook 4.2 SGML (see http://www.docbook.org/). The program itself is merely a driver for the Pod::2::DocBook class; if you're interested in details of pod-to-SGML translation see Pod::2::DocBook.

OPTIONS AND ARGUMENTS

[**--help**]	Print usage and exit.

|**refentry**]	Specifies the document type for the output file; the default is **section**.
[**--title**=*title*]	Specifies the document title. The default is *infile*, if it is supplied, or empty string otherwise.
[**--spaces**=*# spaces per indent level*]	Specifies the number of spaces per indent level in the SGML output; the default is 2.
[**--fix-double-quotes**]	Replace pairs of double quotes in regular paragraphs with <quote> and </quote> (see Pod::2::DocBook for details).
[**--no-header**]	Omit the DOCTYPE line from the output.
infile	The name of the file from which to read pod source; if not supplied, STDIN is used for input.
outfile	The name of the file to which to write SGML; if not supplied, STDOUT is used for output.

SEE ALSO

perlpod, Pod::2::DocBook

AUTHOR

Nandu Shah <nandu@zvolve.com>

COPYRIGHT

Chapter 4. Open questions

some terms used in this book

terms

POD	Plain Old Documentation
DocBook	

Warning
--- to be done ---