Documentation for txt2docbook

Thomas Weber

Table of Contents

Purpose

This program reads a ascii document and converts it into a valid docbook xml file.

Why would one need this?

Docbook is a really cool format to write complex technical documents. On the other hand its to complicated to use it for rather simple papers because one has to write to much *overhead* to get a nice formated result of his work.

With this small tool, you can write a ordinary `README.TXT` like file (following certain simple rules), convert it to xml and send it through one ore more stylesheets to publish it.

Additionally its possible to use a differtent backend module to generate other output formats. This is a bit odd because of the fact that one of the strenghts of docbook, or xml in general, is the possibility to convert it easily to various formats by applying a xsl stylesheet. For special purposes, however, its feasible to use the *shortcut* way through the perl backend.

But it does not support a special feature i need!

By using this converter, you can also add any valid docbook tag into the 'source' file. This way you are not limited to the elements it supports. Instead, you can use the full power of docbook, freed of the nasty routine work (tagging sections, paragraphs, lists)

You can also extend/customize this program very easily to your personal needs.

Its such a simple idea, is there no other tool like this?

In my search for a solution to write well formated papers, i found only one program. It is named 'APT-Convert' and its available under GPL from http://www.xmlmind.com/aptconvert.html.

However, it did not satisfy all my needs, so i started to write my own converter. The syntax of the ascii files is slightly inspired by the APT ("Almost Plain Text) format, though.

Usage

Requirements

All you need to run this tool is `perl` No special modules are used.

However, to get your final document, you need to install and configure the `docbook.dtd` , certain stylesheets and a XSLT-processor. Installing these tools is out of the scope of this guide. Please refer to to http://www.docbook.org for more information.

Configuration

Before you begin to use `txt2docbook` you have to set the public identifier for the docbook DTD in the file `out-put.pl`

```
 $SYSTEMIDENTIFIER="/your/path/to/docbookx.dtd";
```

or:

```
 $SYSTEMIDENTIFIER="http://your.host/dtds/docbookx.dtd";
```

It is possible to omit the public identifier by commenting it out (not recommended). A XML validator can't check the XML file if there is no identifier available! Some XSLT-Processors will also refuse to transform the file.

```
 # $SYSTEMIDENTIFIER="docbook/docbookx.dtd";
 # $IDENTIFIER='<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN" "'.$SYSTEM
```

Converting

```
 txt2docbook [inputfile]
```

The program parses the inputfile and sends the resulting XML to `STDOUT` . Hence you can send it into a file or pipe it right into your XML processor.

By setting the -o switch, you can use a alternative output module. The module must reside in the path of the main script and it needs to follow this naming rule: output_FORMATNAME.pl (i.e. output_html.pl, output_foo.pl).

Thus

```
 txt2docbook -o html [inputfile]
```

will generate a html file from of your source document.

Syntax

As sayed before, the ascii format has to follow some syntax rules in order to get correctly converted to a XML file. The current implementation supports the following elements:

paragraphs	To write the text
sections	Split your document into several parts
item lists	Simple list with bullets.
variable lists	Term + explanation of the term. This block uses a `varlist` , for example.
markup	Markup words
tags	Usage of docbook tags

In the next sections, you can learn more about using the several features.

The document head

This converter generates always a docbook *article* document. A article needs a title. To set it, the converter uses *the very first line of text* of the input file. So the first line is not a section but the title of the document. In addition, the 2. line of the source file gets converted to the author first- and surname tags. If you don't want to set the author name, simply leave the 2. line of your document empty.

Paragraphs

The basic building blocks of a text are paragraphs. You start a new paragraph by closing the previous one with a empty line after the last line of text.

```
First paragraph
multilined
still goes on here

next paragraph
```

Sections

You can use two different methods to split your text into several sections.

Asterisk (*) marker

```
* section level 1

 ** section level 2

  *** section level 3

* next level 1 section

** not indented level 2 section
```

If you choose this method, you are freed of counting the section numbers. Moving section is effortless. A apropriate stylesheet will generate the section numbers later.

Dotted numbers

```
1. section level 1

 1.2 section level 2
 1.3. also section level 2

  1.3.1 section level 3

2. next level 1 section

2.1. indenting is not needed
```

You mark a new section by using numbers as the first columns of a line. By using dots between the numbers, you can denote subsections. It is no difference whether you use a trailing dot or not.

Using the numbers to mark section is a advantage for small `README` like documents. Both the ascii and docbook version will have sectionnumbers.

Lists

You begin a list by using one of several list item markers. See the following example:

```
This is text
Valid list markers are -, o, +, =

- the first list item
- another list item

- 3rd list item
goes on in this line

This is text (=list end)

o also a list item
- you can even mix the markers
+ item
= item
```

A list ends with the first empty line after a item is not followed by another item.

Varlists

DocBook uses a tag called `varlist` to markup term-description lists. The basic usage is similiar to a list block, so you begin the `varlist` with the first `varlist item` and end it by a empty line followed by another block.

```
This is text

[term] Description of the term,
 can have multiple lines

[next term]
 Description can start also in the next line

Text continues.
```

Text markup

Two basic tags of DocBook are supported by this version:

- emphasis (!....!)
- filename (_...._)

```
 A text with a !very important! message in it.

 Some text with a filename like _/usr/bin/perl_ in it.
```

If you use Urls (xxxx://xxx.xxx.xxx) in your text, you'll find corresponding `ULINK` tags in the converted document.

Tags

As sayed before: you can use arbitrary docbook tags in your text. However, there are two modes how you can do this.

The first way is to simply write the tags into your text (easy, huh?).

By writing a *single* tag into a line, you can turn off the converter for all the lines until the corresponding closing (/...) tag gets readed. This way, you can make use of `programlisting` and other "do-not-format" blocks.

Includes

You can include any other ascii file into the current one. To do so, use the following syntax:

```
 &filename;
```

`txt2docbook` will continue by converting the new file into the output document and returns to the parent file when finished.

The include depth is not limited.

Customization

As soon as you know some basic stuff about the architecture of this program, it should be very easy for you to extend/customize it.

The tool consists of only 3 parts:

- `txt2docbook.pl`
- `blocks.pm`
- `output.pl`

All 3 parts have to reside in the same directory.

`txt2docbook.pl`

This is the *main* part of the program. It reads the source file, parses it line by line and starts new *blocks* as needed.

`blocks.pm`

For each supported docbook element, you'll find one perl class in here. Most of the hard word (i.e. deciding whether a block can be closed by a certain other block) is done here. Touching this file is only needed if you want to write a completely new feature.

`output.pl`

When a block gets started or closed, it calls the appropriate function in this sourcefile. All the formating of the output file is done here (including urls and markup abbreviations). Thus it will be your primary playground if you want to change the resulting tags of a parsed file. By writing a new `output.pl` , you can even change the output format from docbook-xml to whatever you want.

`output_html.pl`

Use this alternative backend to generate html instead of xml. Don't expect a fancy design of the output file at this time.

License