

Docutils Front-End Tools

Author: David Goodger
Contact: goodger@users.sourceforge.net
Revision: 1.25
Date: 2002-10-31

Contents

Introduction

[Getting Help](#)

The Tools

[buildhtml.py](#)

[html.py](#)

[Stylesheets](#)

[pep.py](#)

[pep2html.py](#)

[docutils-xml.py](#)

[publish.py](#)

[quicktest.py](#)

Customization

[Command-Line Options](#)

[Configuration Files](#)

[Configuration File Entries](#)

Introduction

Once the Docutils package is unpacked, you will discover a “tools” directory containing several front ends for common Docutils processing. Rather than a single all-purpose program, Docutils has many small front ends, each specialized for a specific “Reader” (which knows how to interpret a file in context), a “Parser” (which understands the syntax of the text), and a “Writer” (which knows how to generate a specific data format). Most front ends have common options and the same command-line usage pattern:

```
toolname [options] [<source> [<destination>]]
```

The exceptions are [buildhtml.py](#) and [pep2html.py](#). See [html.py](#) for concrete examples. Each tool has a “--help” option which lists the [command-line options](#) and arguments it supports. Processing can also be customized with [configuration files](#).

The two arguments, “source” and “destination”, are optional. If only one argument (source) is specified, the standard output (stdout) is used for the destination. If no arguments are specified, the standard input (stdin) is used for the source as well.

Getting Help

First, try the “`--help`” option each front-end tool has.

Users who have questions or need assistance with Docutils or reStructuredText should [post a message](#) to the [Docutils-Users mailing list](#). The [Docutils project web site](#) has more information.

The Tools

buildhtml.py

Readers: Standalone, PEP

Parser: reStructuredText

Writers: HTML, PEP/HTML

Use `buildhtml.py` to generate `.html` from all the `.txt` files (including PEPs) in each `<directory>` given, and their subdirectories too. (Use the `--local` option to skip subdirectories.)

Usage:

```
buildhtml.py [options] [<directory> ...]
```

After unpacking the Docutils package, the following shell commands will generate HTML for all included documentation:

```
cd docutils/tools
buildhtml.py ..
```

For official releases, the directory may be called “`docutils-X.Y`”, where “`X.Y`” is the release version. Alternatively:

```
cd docutils
tools/buildhtml.py --config=tools/docutils.conf
```

The current directory (and all subdirectories) is chosen by default if no directory is named. Some files may generate system messages (`tools/test.txt` contains intentional errors); use the `--quiet` option to suppress all warnings. The `--config` option ensures that the correct stylesheets, templates, and settings are in place (`./docutils.conf` is picked up automatically). Command-line options may be used to override config file settings or replace them altogether.

html.py

Reader: Standalone

Parser: reStructuredText

Writer: HTML

The `html.py` front end reads standalone reStructuredText source files and produces HTML 4 (XHTML 1) output compatible with modern browsers. For example, to process a reStructuredText file “`test.txt`” into HTML:

```
html.py test.txt test.html
```

In fact, there *is* a “`test.txt`” file in the “`tools`” directory. It contains “at least one example of each reStructuredText construct”, including intentional errors. Use it to put the system through its paces and compare input to output.

Now open the “`test.html`” file in your favorite browser to see the results. To get a footer with a link to the source file, date & time of processing, and links to the Docutils projects, add some options:

```
html.py -stg test.txt test.html
```

Stylesheets

`html.py` inserts into the generated HTML a link to a cascading stylesheet, defaulting to “`default.css`” (override with a “`--stylesheet`” or “`--stylesheet-path`” command-line option or with configuration file settings). The “`tools/stylesheet/default.css`” stylesheet is provided for basic use. To experiment with styles, rather than editing the default stylesheet (which will be updated as the project evolves), it is recommended to use an “`@import`” statement to create a “wrapper” stylesheet. For example, a “`my.css`” stylesheet could contain the following:

```
@import url(default.css);

h1, h2, h3, h4, h5, h6, p.topic-title {
    font-family: sans-serif }
```

Generate HTML with the following command:

```
html.py -stg --stylesheet my.css test.txt test.html
```

When viewed in a browser, the new “wrapper” stylesheet will change the typeface family of titles to “sans serif”, typically Helvetica or Arial. Other styles will not be affected. Styles in wrapper stylesheets override styles in imported stylesheets, enabling incremental experimentation.

pep.py

Reader: PEP

Parser: reStructuredText

Writer: PEP/HTML

`pep.py` reads a new-style PEP (marked up with reStructuredText) and produces HTML. It requires a template file and a stylesheet. By default, it makes use of a “`pep-html-template`” file and a “`default.css`” stylesheet in the current directory, but these can be overridden by command-line options or configuration files. The “`tools/stylesheet/pep.css`” stylesheet is intended specifically for PEP use.

The “`docutils.conf`” [configuration file](#) in the “`spec`” directory of Docutils contains a default setup for use in processing the PEP files there (`spec/pep-*.txt`) into HTML. It specifies a default template (`tools/pep-html-template`) and a default stylesheet (`tools/stylesheet/pep.css`). See [Stylesheets](#) above for more information.

pep2html.py

Reader: PEP

Parser: reStructuredText

Writer: PEP/HTML

`pep2html.py` is a modified version of the original script by Fredrik Lundh, with support for Docutils added. It reads the beginning of a PEP text file to determine the format (old-style indented or new-style reStructuredText) and processes accordingly. Since it does not use the Docutils front end mechanism (the common command-line options are not supported), it must be configured using [configuration files](#). The template and stylesheet requirements of `pep2html.py` are the same as those of [pep.py](#) above.

Arguments to `pep2html.py` may be a list of PEP numbers or `.txt` files. If no arguments are given, all files of the form “`pep-*.txt`” are processed.

docutils-xml.py

Reader: Standalone

Parser: reStructuredText

Writer: XML (Docutils native)

The `docutils-xml.py` front end produces Docutils-native XML output. This can be transformed with standard XML tools such as XSLT processors into arbitrary final forms.

publish.py

Reader: Standalone

Parser: reStructuredText

Writer: Pseudo-XML

`publish.py` is used for debugging the Docutils Reader to Transform to Writer pipeline. It produces a compact pretty-printed “pseudo-XML”, where nesting is indicated by indentation (no end-tags). External attributes for all elements are output, and internal attributes for any leftover “pending” elements are also given.

quicktest.py

Reader: N/A

Parser: reStructuredText

Writer: N/A

The `quicktest.py` tool is used for testing the reStructuredText parser. It does not use a Docutils Reader or Writer or the standard Docutils command-line options. Rather, it does its own I/O and calls the parser directly. No transforms are applied to the parsed document. Various forms output are possible:

- Pretty-printed pseudo-XML (default)
- Test data (Python list of input and pseudo-XML output strings; useful for creating new test cases)
- Pretty-printed native XML
- Raw native XML (with or without a stylesheet reference)

Customization

Command-Line Options

Each front-end tool supports command-line options for one-off customization. For persistent customization, use [configuration files](#).

Use the “-help” option on each of the front ends to list the command-line options it supports. Command-line options and their corresponding configuration file entry names are listed in [Configuration File Entries](#) below.

Configuration Files

Configuration files are used for persistent customization; they can be set once and take effect every time you use a front-end tool.

By default, Docutils checks the following places for configuration files, in the following order:

- 1 `/etc/docutils.conf`: This is a system-wide configuration file, applicable to all Docutils processing on the system.
- 2 `./docutils.conf`: This is a project-specific configuration file, located in the current directory. The Docutils front end has to be executed from the directory containing this configuration file for it to take effect (note that this may have nothing to do with the location of the source files). Settings in the project-specific configuration file will override corresponding settings in the system-wide file.
- 3 `~/.docutils`: This is a user-specific configuration file, located in the user's home directory. Settings in this file will override corresponding settings in both the system-wide and project-specific configuration files.

If more than one configuration file is found, all will be read but later entries will override earlier ones. For example, a “stylesheet” entry in a user-specific configuration file will override a “stylesheet” entry in the system-wide file.

In addition, a configuration file may be explicitly specified with the “-config” command-line option. This configuration file is read after the three implicit ones listed above.

Configuration files use the standard [ConfigParser.py Python](#) module. From its documentation:

The configuration file consists of sections, lead by a “[section]” header and followed by “name: value” entries, with continuations in the style of [RFC 822](#); “name=value” is also accepted. Note that leading whitespace is removed from values. The optional values can contain format strings which refer to other values in the same section, or values in a special DEFAULT section. Additional defaults can be provided upon initialization and retrieval. Lines beginning with “#” or “;” are ignored and may be used to provide comments.

Docutils currently only uses an “[options]” section; all other sections are ignored.

Note

The configuration file format may change in the future.

Configuration entry names correspond to internal option attributes. Underscores (“_”) and hyphens (“-”) can be used interchangeably in entry names. The correspondence between entry names and command-line options is listed in [Configuration File Entries](#) below.

Configuration File Entries

Names in the first column of the table below are runtime settings. Most may be specified in [configuration files](#), where hyphens may be used in place of underscores. Some knowledge of [Python](#) is assumed for some attributes.

Setting/Config Entry	Description
expose_internals	List of internal attributes to expose as external attributes (with “internal:” namespace prefix). Default: don't (None). Options: <code>--expose-internal-attribute</code> (hidden, for development use only).
compact_lists	(HTML Writer.) Remove extra vertical whitespace between items of bullet lists and enumerated lists, when list items are “simple” (i.e., all items each contain one paragraph and/or one “simple” sublist only). Default: enabled (1). Options: <code>--compact-lists</code> , <code>--no-compact-lists</code> .

Setting/Config Entry	Description
config	Path to a configuration file to read (if it exists) [1]. Settings may override defaults and earlier settings. This is only effective as a command-line option; setting it in a config file has no effect. Filesystem path settings contained within the config file will be interpreted relative to the config file's location (<i>not</i> relative to the current working directory). Default: None. Options: <code>--config</code> .
datestamp	Include a time/datestamp in the document footer. Contains a format string for <code>time.strftime</code> . Default: None. Options: <code>--date</code> , <code>-d</code> , <code>--time</code> , <code>-t</code> , <code>--no-datestamp</code> .
debug	Report debug-level system messages. Default: don't (None). Options: <code>--debug</code> , <code>--no-debug</code> .
dump_internals	At the end of processing, write all internal attributes of the document (<code>document.__dict__</code>) to stderr. Default: don't (None). Options: <code>--dump-internals</code> (hidden, for development use only).
dump_pseudo_xml	At the end of processing, write the pseudo-XML representation of the document to stderr. Default: don't (None). Options: <code>--dump-pseudo-xml</code> (hidden, for development use only).
dump_settings	At the end of processing, write all Docutils settings to stderr. Default: don't (None). Options: <code>--dump-settings</code> (hidden, for development use only).
dump_transforms	At the end of processing, write a list of all transforms applied to the document to stderr. Default: don't (None). Options: <code>--dump-transforms</code> (hidden, for development use only).
embed_stylesheet	(HTML Writer.) Embed the stylesheet in the output HTML file. The stylesheet file must be accessible during processing. The stylesheet is embedded inside a comment, so it must not contain the text “ <code>--</code> ” (two hyphens). Default: link, don't embed (None). Options: <code>--embed-stylesheet</code> , <code>--link-stylesheet</code> .
footnote_backlinks	Enable or disable backlinks from footnotes and citations to their references. Default: enabled (1). Options: <code>--footnote-backlinks</code> , <code>--no-footnote-backlinks</code> .
footnote_references	(HTML Writer.) Format for footnote references, one of “superscript” or “brackets”. Default: “superscript”; “brackets” in PEP/HTML Writer. Options: <code>--footnote-references</code> .
generator	Include a “Generated by Docutils” credit and link in the document footer. Default: off (None). Options: <code>--generator</code> , <code>-g</code> , <code>--no-generator</code> .
halt_level	The threshold at or above which system messages are converted to exceptions, halting execution immediately. Default: severe (4). Options: <code>--halt</code> , <code>--strict</code> .
indents	(XML Writer.) Generate XML with indents and newlines. Default: don't (None). Options: <code>--indents</code> .
input_encoding	Default: auto-detect (None). Options: <code>--input-encoding</code> , <code>-i</code> .
language_code	ISO 639 2-letter language code (3-letter codes used only if no 2-letter code exists). Default: English (“en”). Options: <code>--language</code> , <code>-l</code> .
newlines	(XML Writer.) Generate XML with newlines before and after tags. Default: don't (None). Options: <code>--newlines</code> .

Setting/Config Entry	Description
no_random	(PEP/HTML Writer.) Workaround for platforms which core-dump on “import random”. Default: random enabled (None). Options: <code>--no-random</code> (hidden).
output_encoding	Default: UTF-8. Options: <code>--output-encoding</code> , <code>-o</code> .
pep_home	(PEP/HTML Writer.) Home URL prefix for PEPs. Default: current directory (“.”). Options: <code>--pep-home</code> .
pep_references	(reStructuredText Parser.) Recognize and link to PEP references (like “PEP 258”). Default: disabled (None); enabled (1) in PEP Reader. Options: <code>--pep-references</code> .
pep_stylesheet	(PEP/HTML Writer.) CSS stylesheet URL, used verbatim. Overrides HTML stylesheet (<code>--stylesheet</code>). Default: None. Options: <code>--pep-stylesheet</code> .
pep_stylesheet_path	(PEP/HTML Writer.) Path to CSS stylesheet [1]. Path is adjusted relative to the output HTML file. Overrides HTML stylesheet (<code>--stylesheet</code>) and PEP stylesheet (<code>--pep-stylesheet</code>). Default: None. Options: <code>--pep-stylesheet-path</code> .
pep_template	(PEP/HTML Writer.) Path to PEP template file [1]. Default: “pep-html-template” (in current directory). Options: <code>--pep-template</code> .
python_home	(PEP/HTML Writer.) Python’s home URL. Default: parent directory (“..”). Options: <code>--python-home</code> .
recurse	(<code>buildhtml.py</code> front end.) Recursively scan subdirectories. Default: recurse (1). Options: <code>--recurse</code> , <code>--local</code> .
report_level	Verbosity threshold at or above which system messages are reported. Default: warning (2). Options: <code>--report</code> , <code>-r</code> , <code>--verbose</code> , <code>-v</code> , <code>--quiet</code> , <code>-q</code> .
rfc_references	(reStructuredText Parser.) Recognize and link to RFC references (like “RFC 822”). Default: disabled (None); enabled (1) in PEP Reader. Options: <code>--rfc-references</code> .
silent	(<code>buildhtml.py</code> front end.) Work silently (no progress messages). Independent of “report_level”. Default: show progress (None). Options: <code>--silent</code> .
source_link	Include a “View document source” link in the document footer. URL will be relative to the destination. Default: don’t (None). Options: <code>--source-link</code> , <code>--no-source-link</code> .
source_url	An explicit URL for a “View document source” link, used verbatim. Default: compute if source_link (None). Options: <code>--source-uri</code> , <code>--no-source-link</code> .
stylesheet	(HTML Writer.) CSS stylesheet URL, used verbatim. Default: “default.css”. Options: <code>--stylesheet</code> .
stylesheet_path	(HTML Writer.) Path to CSS stylesheet [1]. Overrides “stylesheet” URL option (<code>--stylesheet</code>). Path is adjusted relative to the output HTML file. Default: None. Options: <code>--stylesheet</code> .
tab_width	(reStructuredText Parser.) Number of spaces for hard tab expansion. Default: 8. Options: <code>--tab-width</code> .
toc_backlinks	Enable backlinks from section titles to table of contents entries (“entry”), to the top of the TOC (“top”), or disable (“none”). Default: “entry”. Options: <code>--toc-entry-backlinks</code> , <code>--toc-top-backlinks</code> , <code>--no-toc-backlinks</code> .

Setting/Config Entry	Description
warning_stream	Path to a file for the output of system messages (warnings) [1]. Default: stderr (None). Options: <code>--warnings</code> .
For Internal Use Only (setting these in a config file has no effect)	
_directories	(<code>buildhtml.py</code> front end.) List of paths to source directories, set from positional arguments. Default: current working directory (None). No command-line options.
_destination	Path to output destination, set from positional arguments. Default: stdout (None). No command-line options.
_source	Path to input source, set from positional arguments. Default: stdin (None). No command-line options.

[1] Path relative to the working directory of the process at launch.