Who needs Pandoc when you have Sphinx?

An exploration of the parsers and builders of the Sphinx documentation tool

FOSDEM 2019

@stephenfin

reStructuredText, Docutils & Sphinx

1

```
A little reStructuredText
=========================

This document demonstrates some basic features of |rst|. You can use
**bold** and *italics*, along with ``literals``. It’s quite similar
to `Markdown`_ but much more extensible. CommonMark may one day
approach this [1]_, but today is not that day. `Docutils`__ does all
this for us.

.. |rst| replace:: **reStructuredText**
.. _Markdown: https://daringfireball.net/projects/markdown/
.. [1] https://talk.commonmark.org/t/444
__  http://docutils.sourceforge.net/
```

intro.rst

```rst
A little reStructuredText
=========================

This document demonstrates some basic features of |rst|. You can use
**bold** and *italics*, along with ``literals``. It’s quite similar
to `Markdown`_ but much more extensible. CommonMark may one day
approach this [1]_, but today is not that day. `Docutils`__ does all
this for us.

.. |rst| replace:: **reStructuredText**
.. _Markdown: https://daringfireball.net/projects/markdown/
.. [1] https://talk.commonmark.org/t/444
__  http://docutils.sourceforge.net/
```

intro.rst

A little reStructuredText

This document demonstrates some basic features of **reStructuredText**. You can use **bold** and *italics*, along with `literals`. It's quite similar to Markdown but much more extensible. CommonMark may one day approach this [1], but today is not that day. Docutils does all this for us.

[1] https://talk.commonmark.org/t/444/

```
A little more reStructuredText
==============================
The extensibility really comes into play with directives and
roles. We can do things like link to RFCs (:RFC:`2324`, anyone?)
or generate some more advanced formatting (I do love me some
H\ :sub:`2`\ O).

.. warning::

   The power can be intoxicating.

Of course, all the stuff we showed previously *still works!* The
only limit is your imagination/interest.
```

more.rst

```rst
A little more reStructuredText
==============================
The extensibility really comes into play with directives and
roles. We can do things like link to RFCs (:RFC:`2324`, anyone?)
or generate some more advanced formatting (I do love me some
H\ :sub:`2`\ O).

.. warning::

   The power can be intoxicating.

Of course, all the stuff we showed previously *still works!* The
only limit is your imagination/interest.
```

more.rst

A little more reStructuredText

The extensibility really comes into play with directives and roles. We can do things like link to RFCs (RFC 2324, anyone?) or generate some more advanced formatting (I do love me some H_2O).

> **Warning**
>
> The power can be intoxicating.

Of course, all the stuff we showed previously *still works!* The only limit is your imagination/interest.

reStructuredText provides the syntax

Docutils provides the parsing and file generation

reStructuredText provides the syntax

Docutils provides the parsing and file generation

Sphinx provides the cross-referencing

Docutils use readers, parsers, transforms, and writers

Docutils works with individual files

Docutils use readers, parsers, transforms, and writers

Docutils works with individual files

Sphinx uses readers, parsers, transforms, writers and builders

Sphinx works with multiple, cross-referenced files

How Does Docutils Work?

2

```
About me
========

Hello, world. I am **bold** and *maybe* I am brave.
```

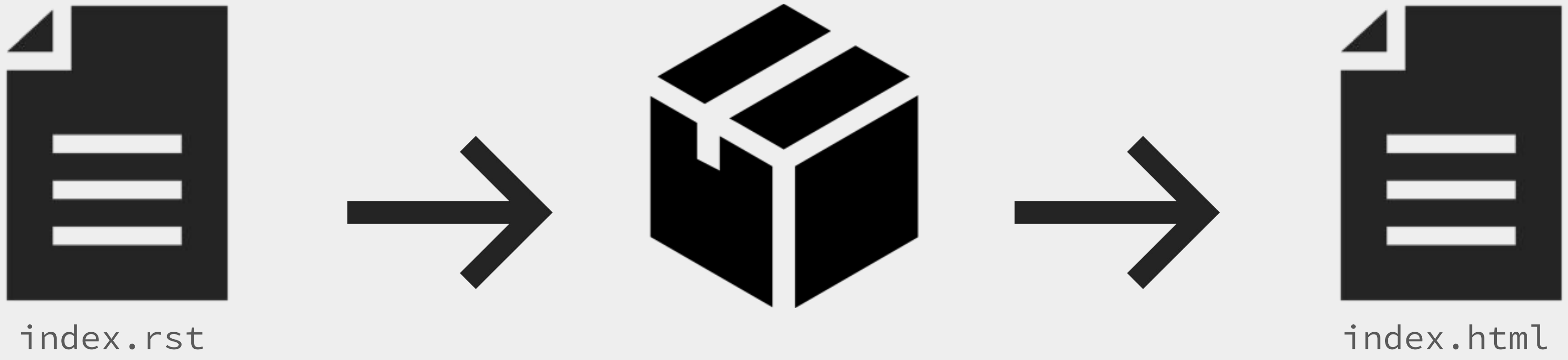
index.rst

```
$ rst2html index.rst
```

About me

Hello, world. I am **bold** and *maybe* I am brave.

index.html


index.rst
index.html

```
$ rst2pseudoxml index.rst
```

```
<document ids="about-me" names="about\ me" source="index.rst"
title="About me">
    <title>
        About me
    <paragraph>
        Hello, world. I am
        <strong>
            bold
        and
        <emphasis>
            maybe
        I am brave.
```

index.xml

```
$ ./docutils/tools/quicktest.py index.rst
```

```
<document source="index.rst">
    <section ids="about-me" names="about\ me">
        <title>
            About me
        <paragraph>
            Hello, world. I am
            <strong>
                bold
            and
            <emphasis>
                maybe
            I am brave.
```

index.xml

Readers (reads from source and passes to the parser)

Parsers (creates a doctree model from the read file)

Transforms (add to, prune, or otherwise change the doctree model)

Writers (converts the doctree model to a file)

Readers (reads from source and passes to the parser)

Parsers (creates a doctree model from the read file)

Transforms (add to, prune, or otherwise change the doctree model)

Writers (converts the doctree model to a file)

What About Sphinx?

3

```rst
About me
========

Hello, world. I am **bold** and *maybe* I am brave.
```

index.rst

```python
master_doc = 'index'
```

conf.py

```
$ sphinx-build -b html . _build
```

About me

Hello, world. I am **bold** and *maybe* I am brave.

index.html

Readers (reads from source and passes to the parser)

Parsers (creates a doctree model from the read file)

Transforms (add to, prune, or otherwise change the doctree model)

Writers (converts the doctree model to a file)

Builders (call the readers, parsers, transformers, writers)

Application (calls the builder(s))

Environment (store information for future builds)

Builders (call the readers, parsers, transformers, writers)

Application (calls the builder(s))

Environment (store information for future builds)

```
...
updating environment: 1 added, 0 changed, 0 removed
reading sources... [100%] index
looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
generating indices... done
writing additional pages... done
copying static files... done
copying extra files... done
dumping search index in English (code: en) ... done
dumping object inventory... done
build succeeded.
```

Docutils provides almost 100 **node types**

`document`	(the root element of the document tree)
`section`	(the main unit of hierarchy for documents)
`title`	(stores the title of a document, section, ...)
`subtitle`	(stores the subtitle of a `document`)
`paragraph`	(contains the text and inline elements of a single paragraph)
`block_quote`	(used for quotations set off from the main text)
`bullet_list`	(contains `list_item` elements marked with bullets)
`note`	(an admonition, a distinctive and self-contained notice)
`...`	`...`

Sphinx provides its own custom **node types**

`translatable`	(indicates content which supports translation)
`not_smartquotable`	(indicates content which does not support smart-quotes)
`toctree`	(node for inserting a "TOC tree")
`versionmodified`	(version change entry)
`seealso`	(custom "see also" admonition)
`productionlist`	(grammar production lists)
`manpage`	(reference to a man page)
`pending_xref`	(cross-reference that cannot be resolved yet)
`...`	...

Docutils provides dozens of **transforms**

`DocTitle`	(promote `title` elements to the document level)
`DocInfo`	(transform initial field lists to docinfo elements)
`SectNum`	(assign numbers to the titles of document sections)
`Contents`	(generate a table of contents from a `document` or sub-node)
`Footnotes`	(resolve links to footnotes, citations and their references)
`Messages`	(place system messages into the document)
`SmartQuotes`	(replace ASCII quotation marks with typographic form)
`Admonitions`	(transform specific admonitions to generic ones)
...	...

Sphinx also provides additional **transforms**

`MoveModuleTargets`	(promote initial module targets to the section title)
`AutoNumbering`	(register IDs of tables, figures and literal blocks to assign numbers)
`CitationReferences`	(replace citation references with `pending_xref` nodes)
`SphinxSmartQuotes`	(custom `SmartQuotes` to avoid transform for some extra node types)
`DoctreeReadEvent`	(emit `doctree-read` event)
`ManpageLink`	(find manpage section numbers and names)
`SphinxDomains`	(collect objects to Sphinx domains for cross referencing)
`Locale`	(replace translatable nodes with their translated doctree)
...	...

Using Additional Parsers

4

There are a number of **parsers** available

reStructuredText (part of `docutils`)

Markdown (part of `recommonmark`)

Jupyter Notebooks (part of `nbsphinx`)

```
# About me

Hello, world. I am **bold** and *maybe* I am brave.
```

index.md

```
$ cm2html index.md
```

About me

Hello, world. I am **bold** and *maybe* I am brave.

index.html

```
$ cm2pseudoxml index.md
```

```
<document ids="about-me" names="about\ me" source="index.md"
title="About me">
    <title>
        About me
    <paragraph>
        Hello, world. I am
        <strong>
            bold
        and
        <emphasis>
            maybe
        I am brave.
```

index.xml

```
# About me

Hello, world. I am **bold** and *maybe* I am brave.
```

index.md

```python
from recommonmark.parser import CommonMarkParser

master_doc = 'index'

source_parsers = {'.md': CommonMarkParser}
source_suffix = '.md'
```

conf.py

```
from recommonmark.parser import CommonMarkParser

master_doc = 'index'

source_parsers = {'.md': CommonMarkParser}
source_suffix = '.md'
```

conf.py

```
$ sphinx-build -b html . _build
```

About me

Hello, world. I am **bold** and *maybe* I am brave.

Using Additional Writers, Builders

5

Docutils provides a number of in-tree **writers**

`docutils_xml`	(simple XML document tree Writer)
`html4css1`	(simple HTML document tree Writer)
`latex2e`	(LaTeX2e document tree Writer)
`manpage`	(simple man page Writer)
`null`	(a do-nothing Writer)
`odf_odt`	(ODF Writer)
`pep_html`	(PEP HTML Writer)
`pseudoxml`	(simple internal document tree Writer)
`...`	...

```
$ rst2html5 index.rst
```

```python
from docutils.core import publish_file
from docutils.writers import html5_polyglot

with open('README.rst', 'r') as source:
    publish_file(source=source,
                 writer=html5_polyglot.Writer())
```

```
$ pip install rst2txt
$ rst2txt index.rst
```

```python
from docutils.core import publish_file
from rst2txt

with open('README.rst', 'r') as source:
    publish_file(source=source,
                 writer=rst2txt.Writer())
```

Sphinx provides its own in-tree **builders**

`html`	(generates output in HTML format)
`qthelp`	(like `html` but also generates Qt help collection support files)
`epub`	(like `html` but also generates an epub file for eBook readers)
`latex`	(generates output in LaTeX format)
`text`	(generates text files with most rST markup removed)
`man`	(generates manual pages in the groff format)
`texinfo`	(generates textinfo files for use with `makeinfo`)
`xml`	(generates Docutils-native XML files)
`...`	...

```
$ sphinx-build -b html . _build
$ pip install sphinx-asciidoc
$ sphinx-build -b asciidoc . _build
```

Writing Your Own Parsers, Writers

6

Reading (reads from source and passes to the parser)

Parsing (creates a doctree model from the read file)

Transforming (applies transforms to the doctree model)

Writing (converts the doctree model to a file)

```python
from docutils import parsers

class Parser(parsers.Parser):
  supported = ('null',)
  config_section = 'null parser'
  config_section_dependencies = ('parsers',)

  def parse(self, inputstring, document):
    pass
```

docutils/parsers/null.py

We’re not covering Compilers 101

We’re not covering Compilers 101

We’re going to **cheat** 😆

```xml
<?xml version="1.0" encoding="utf-8"?>
<document source="index.rst">
    <section ids="about-me" names="about\ me">
        <title>About me</title>
        <paragraph>Hello, world. I am <strong>bold</strong> and
        <emphasis>maybe</emphasis> I am brave.</paragraph>
    </section>
</document>
```

index.xml

```python
from docutils import parsers
import xml.etree.ElementTree as ET

class Parser(parsers.Parser):
  supported = ('xml',)
  config_section = 'XML parser'
  config_section_dependencies = ('parsers',)

  def parse(self, inputstring, document):
    xml = ET.fromstring(inputstring)
    self._parse(document, xml)

  ...
```

xml_parser.py

```python
...

def _parse(self, node, xml):
  for attrib, value in xml.attrib.items():
    # NOTE(stephenfin): this isn't complete!
    setattr(node, attrib, value)

  for child in xml:
    child_node = getattr(nodes, child.tag)(text=child.text)
    node += self._parse(child_node, child)

  if xml.tail:
    return node, nodes.Text(xml.tail)
  return node
```

xml_parser.py

Reading (reads from source and passes to the parser)

Parsing (creates a doctree model from the read file)

Transforming (applies transforms to the doctree model)

Writing (converts the doctree model to a file)

```python
from docutils import writers

class Writer(writers.Writer):
  supported = ('pprint', 'pformat', 'pseudoxml')
  config_section = 'pseudoxml writer'
  config_section_dependencies = ('writers',)
  output = None

  def translate(self):
    self.output = self.document.pformat()
```

docutils/writers/pseudoxml.py

```python
from docutils import writers

class Writer(writers.Writer):
  supported = ('pprint', 'pformat', 'pseudoxml')
  config_section = 'pseudoxml writer'
  config_section_dependencies = ('writers',)
  output = None

  def translate(self):
    self.output = self.document.pformat()
```

docutils/writers/pseudoxml.py

```python
from docutils import nodes, writers

class TextWriter(writers.Writer):
  supported = ('text',)
  config_section = 'text writer'
  config_section_dependencies = ('writers',)
  output = None

  def translate(self):
    visitor = TextTranslator(self.document)
    self.document.walkabout(visitor)
    self.output = visitor.body
```

rst2txt/writer.py

```python
from docutils import nodes, writers

class TextWriter(writers.Writer):
  supported = ('text',)
  config_section = 'text writer'
  config_section_dependencies = ('writers',)
  output = None

  def translate(self):
    visitor = TextTranslator(self.document)
    self.document.walkabout(visitor)
    self.output = visitor.body
```

rst2txt/writer.py

```python
...

class TextTranslator(nodes.NodeVisitor):
  ...

  def visit_document(self, node):
    pass

  def depart_document(self, node):
    pass

  def visit_section(self, node):
    pass
```

```python
from sphinx.builders import Builder

class TextBuilder(Builder):
  name = 'text'

  def __init__(self):
    pass

  def get_outdated_docs(self):
    pass

  def get_target_uri(self):
    pass
```

sphinx/builders/text.py

```python
...

def prepare_writing(self, docnames):
  pass

def write_doc(self, docnames, doctree):
  pass

def finish(self):
  pass
```

sphinx/builders/text.py

Wrap Up

6

Sphinx and **Docutils** share most of the same architecture…

Readers

Parsers

Transforms

Writers

…but **Sphinx** builds upon and extends Docutils’ core functionality

Builders

Application

Environment

There are multiple **writers/builders** provided by both…

HTML

Manpage

LaTeX

XML

texinfo (Sphinx only)

ODF (Docutils only)

...

...and many more **writers/builders** available along with **readers**

Markdown (reader and builder)

Text (writer)

ODF (builder)

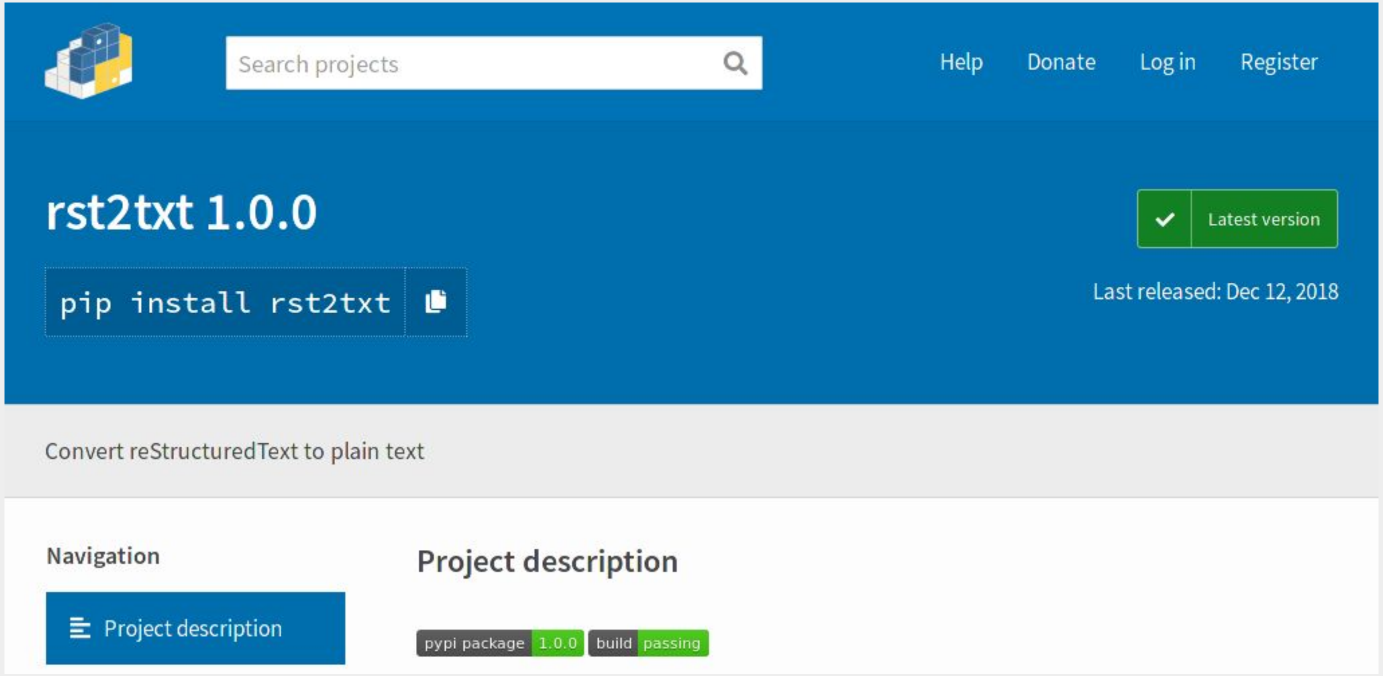
AsciiDoc (builder)

EPUB2 (builder)

reStructuredText (builder)

...

It’s possible to **write your own**



rst2txt 1.0.0

```
pip install rst2txt
```

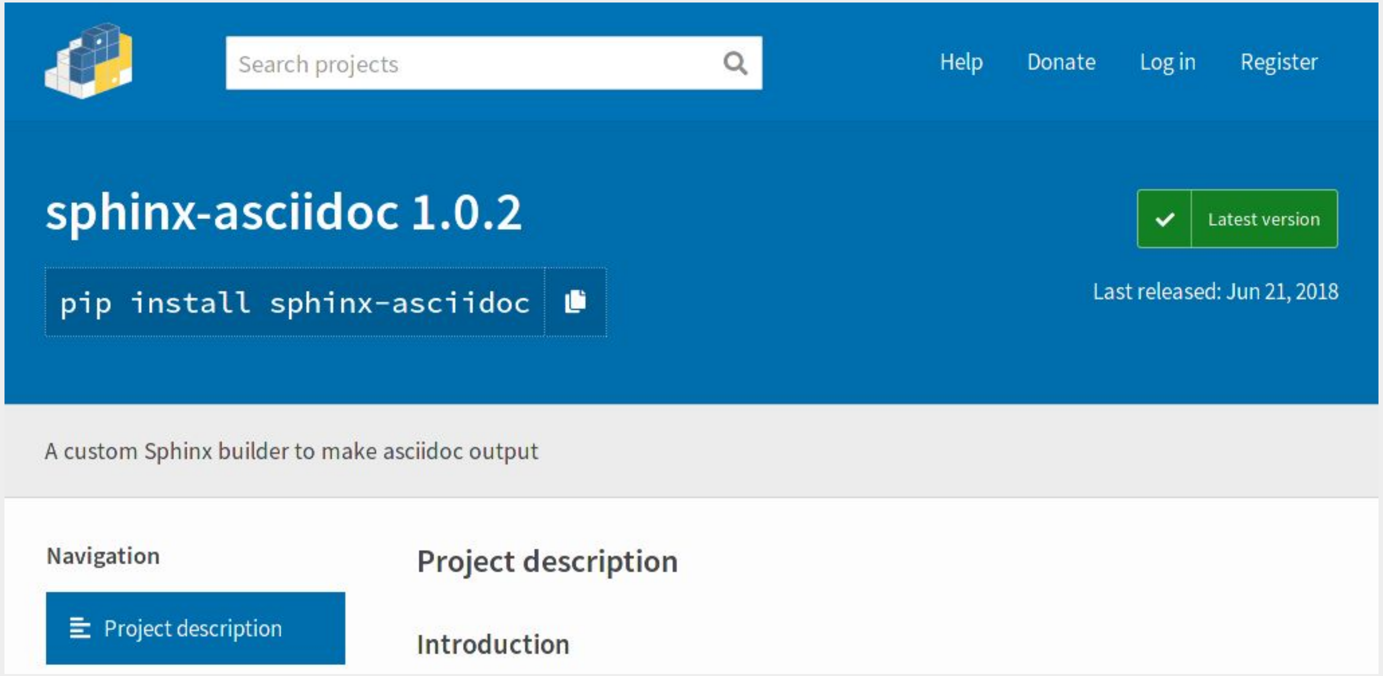
Last released: Dec 12, 2018

Convert reStructuredText to plain text

Project description

pypi package 1.0.0 build passing

It's possible to **write your own**



sphinx-asciidoc 1.0.2

```
pip install sphinx-asciidoc
```

Last released: Jun 21, 2018

A custom Sphinx builder to make asciidoc output

Project description

Introduction

Fin

Who needs Pandoc when you have Sphinx?

An exploration of the parsers and builders of the Sphinx documentation tool

FOSDEM 2019

@stephenfin

Useful Packages and Tools

- recommonmark (provides a Markdown reader)
- sphinx-markdown-builder (provides a Markdown builder)
- sphinx-asciidoc (provides an AsciiDoc builder)
- rst2txt (provides a plain text writer)
- asciidoclive.com (online AsciiDoc Editor)
- rst.ninjs.org (online rST Editor)

References

- Quick reStructuredText
- Docutils Reference Guide
    - reStructuredText Markup Specification
    - reStructuredText Directives
    - reStructuredText Interpreted Text Roles
- Docutils Hacker’s Guide
- PEP-258: Docutils Design Specification

References

- A brief tutorial on parsing reStructuredText (reST) -- Eli Bendersky
- A lion, a head, and a dash of YAML -- Stephen Finucane (🌟)
- OpenStack + Sphinx In A Tree -- Stephen Finucane (🌟)
- Read the Docs & Sphinx now support Commonmark -- Read the Docs Blog