Sphinx memo Documentation

リリース *0.1*

Y.Chubachi

2011 年 08 月 24 日

Contents

CHAPTER 1

reStructuredTextとは

reStructuredText は，その名の通り「構造的な（Structured）文書（Text）」を作成するためのツールです．

「構造的な文書」というと，その筆頭は研究者が書く学術論文でしょう．それ以外でも，ビジネスで作成するレポートや，プレゼンテーション資料もある種の構造をもった文書に該当します．情報技術者にとってなじみ深い仕様書やマニュアルも構造的な文書です．

reStructuredText を使うと，これらの構造的な文書をテキストエディタだけで手軽に作成することができます．作成したテキストを各種のコンバータにかけると，見た目のきれいな HTML や，PDF ファイル，電子出版に使える epub 形式のファイルなどが簡単に生成できます．

reStructuredText はもともとは Python というプログラミング言語の設計書を記述するために作られたものです．しかしながら，近年になってツールが整備され，ちょっとした文書の作成や，Web サイトの構築にも簡単に利用できるようになりました．

この文書は reStructuredText を試してみたい人向けに，Windows の cygwin を利用して文章作成のための環境を整えるためのマニュアルです．これ自体も reStructuredText で作成していますので，どんなことができるのかをお伝えするサンプルにもなるでしょう．

なお，reStructuredText については，下記の URL を参考にしてください．

- reStructuredText 入門
- User refference

CHAPTER 2

docutilsを使った文書作成

2.1 gnupack環境をインストールと設定

簡単にcygwin環境とemacs等をインストールできるパッケージ，grupack [1] をインストールします．ここでは，ユーザのホームディレクトリ（C:\Users\<name>\）[2] にインストールします．

1. gnupack-basic-7.00.exe をダウンロードします
2. ダブルクリックして実行します
3. “解凍先” に C:\Users\<name>\ を指定します

新たに， C:\Users\<name>\gnupack_basic-7.00 ディレクトリができます．その下にファイルが展開されていますので，確認してください．

2.2 cygwin用パッケージ管理ツール

cygwinにパッケージを追加するために， setup.exe をダウンロードします．

1. setup.exe をダウンロードします．
2. ダウンロードしたファイルを C:\Users\<name>\gnupack_basic-7.00 ディレクトリにコピーしてください．
3. また，同じ場所に”package” という名前でフォルダを作成してください．

2.2.1 setup.exeの起動とディレクトリの指定

1. setup.exe を実行します．
2. “次へ” でページをめくり，”Cygwin Setup - Choose Installation Directory” 画面に進みます．
3. “Root Directory” に C:\Users\<name>\gnupack_basic-7.00\app\cygwin\cygwin を指定してください．
4. “次へ” でページをめくり，”Cygwin Setup - Select Local Package Directory” 画面に進みます．
5. “Local Package Directory” に C:\Users\<name>\gnupack_basic-7.00\package を指定してください．

[1] gnupack には basic 版と develop 版があります．develop 版の gcc は今回利用しないので，basic 版をインストールしてください．
[2] \ は「¥」記号に読み替えてください．*<name>* の部分は，あなたのログイン名に置き換えてください．

6. “次へ” でページをめくり，”Cygwin Setup - Choose Download Site(s)” 画面に進みます.
7. “ftp://ftp.iij.ad.jp“を選択してください

2.2.2 パッケージの選択

まず，次のパッケージを追加します.

- Python
- curl

1. “次へ” でページをめくり，”Cygwin Setup - Select Packages” 画面に進みます.
2. “Search” に”python” と入力してください（画面が更新されるまでに若干ラグがあります）.
3. “All:Interpreters” の中にある”python: Python language interpreter” の行にある”Skip” をクリックします（表示がバージョン番号に変わります）.
4. “Search” に”curl” と入力してください.
5. “All:Net” の中にある”curl: Multi-protocol file transfer command-line tool” の行にある”Skip” をクリックします（表示がバージョン番号に変わります）.

2.2.3 パッケージのダウンロードとインストール

1. “次へ” でページをめくり，”Cygwin Setup” 画面に進みます.
2. ダウンロードが終了すると，”Cygwin Setup - Installation Status and CReate Icons” 画面がでます.
3. “Crate icon on Desktop” のチェックは外して，” 完了” を押してください.

2.3 python 用パッケージ管理ツール

1. C:\Users\<name>\gnupack_basic-7.00\mintty.exe を起動して，コマンドプロンプトを表示させます.
2. 次のコマンドを実行してください.

```
# curl -O http://peak.telecommunity.com/dist/ez_setup.py
# python ez_setup.py
```

2.4 docutils のインストール

- rst2pdf は docutils をベースに作られています.
- http://docutils.sourceforge.net/docs/user/tools.html

```
# easy_install docutils
```

2.5 docutils のテスト

2.5.1 テスト用 rst の作成

emacs を起動します.

```
# emacs test.rst &
```

次の通り入力します.

```
タイトル
========

- ぐー
- ちょき
- ぱー
```

2.5.2 rst2html の実行

```
# rst2html.py test.rst > test.html
```

2.5.3 html の表示

結果

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtm
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta name="generator" content="Docutils 0.8: http://docutils.sourceforge.net/" />
<title>タイトル</title>
<style type="text/css">
... 省略...
</style>
</head>
<body>
<div class="document" id="id1">
<h1 class="title">タイトル</h1>

<ul class="simple">
<li>ぐー</li>
<li>ちょき</li>
<li>ぱー</li>
</ul>
</div>
</body>
</html>
```

任意のブラウザで test.html を開き，内容を確認してください.

2.6 その他の文書生成

2.6.1 LaTeX の生成

platex がインストールされている場合，以下のコマンドを入力することで LaTeX のソースファイルが生成され，コンパイルできます．

```
# rst2latex.py hoge.rst
# platex -kanji=utf8 hoge.tex
```

2.6.2 S5 スライドの生成

- http://meyerweb.com/eric/tools/s5/
- http://meyerweb.com/eric/tools/s5/primer.html

2.6.3 OpenOffice Writer 形式ファイルの生成

Writer 形式で出力することもできます．日本語が乱れますので，使い物にはならないかも．

```
# rst2odt.py hoge.rst > hoge.odt
```

CHAPTER 3

PDFを作成する

3.1 rst2pdfのインストール

3.1.1 パッケージの選択

setup.exe を起動して次のパッケージインストールしてください..

1. “All:Devel” の中にある”gcc: C compiler upgrade helper”
2. “All:Graphics” にある”jpeg: A library for manipulating JPEG image format files”
3. “All:Graphics” にある”lcms: Little color management engine - (Python bindings)”
4. “All:X11” にある”libfreetype6: High-quality software font engine (runtime library)”

3.1.2 python パッケージの追加

C:\Users\<name>\gnupack_basic-7.00\mintty.exe を起動して，コマンドプロンプトを表示させます．次のコマンドを実行してください.

```
# easy_install PIL
# easy_install reportlab
# easy_install rst2pdf
```

3.1.3 コンパイルエラーへの対処

コンパイル中に以下のエラーになる場合があります.

```
*** fatal error - unable to remap \\?\C:\Users\<name>\gnupack_basic-7.00\app\cygwin\cygwin\lib\python
Stack trace:
Frame     Function  Args
00224DF8  6102796B  (00224DF8, 00000000, 00000000, 00000000)
002250E8  6102796B  (6117EC60, 00008000, 00000000, 61180977)
00226118  61004F1B  (611A7FAC, 612483E4, 003D0000, 003F0000)
End of stack trace
1 [main] python 6956 fork: child 7068 - died waiting for dll loading, errno 11
error: Setup script exited with error: Resource temporarily unavailable
```

その場合， 一旦すべてのプロセス（シェルや，emacs など）を終了させます。

次に， C:\Users\<name>\gnupack_basic-7.00\app\cygwin\cygwin の下にある\bin\ash.exe を起動して，次のコマンドを入力します.

```
$ /bin/rebaseall
$ exit
```

完了したら，もう一度 easy_install を使ってパッケージを追加します。

3.1.4 フォントの追加

フォントを組み込むために，~/fonts ディレクトリをを作成します。

```
mkdir ~/fonts
```

IPA フォント をダウンロードして解凍します。

TrueType フォントファイル（ipag.ttf,ipagp.ttf,ipam.ttf,ipamp.ttf）を~/font（ C:\Users\<name>\gnupack_basic-7.00 の下の\home\fonts）にコピーします。ls コマンドで確認して，次の結果になれば大丈夫です。

```
# ls ~/fonts
ipag.ttf  ipagp.ttf  ipam.ttf  ipamp.ttf
```

3.1.5 スタイルファイルの作成

以下の内容を”ja.json” というファイル名で作成し，保存します。

```
{
  "embeddedFonts" :[[
    "ipag.ttf",
    "ipagp.ttf",
    "ipam.ttf",
    "ipamp.ttf"
  ]],
  "fontsAlias" : {
    "stdFont": "IPAPGothic",
    "stdBold": "IPAPGothic",
    "stdItalic": "IPAPGothic",
    "stdBoldItalic": "IPAPGothic",
    "stdMono": "IPAPGothic",
    "stdMonoBold": "IPAPGothic",
    "stdSanBold": "IPAPGothic",
    "stdSansBold": "IPAPGothic"
  },
  "styles" : [
    ["base" , {
      "wordWrap": "CJK",
      "kerning" : true
    }],
    ["literal" , {
      "wordWrap": "None"
    }]
  ]
}
```

3.1.6 docutils の修正

次の通りエディタでファイルを開きます.

```
# emacs /usr/lib/python2.6/site-packages/docutils-0.8-py2.6.egg/docutils/languages/__init__.py &
```

18 行目を以下のコードに置き換えます.

```
def get_language(language_code, reporter = None):
```

3.1.7 reportlab の修正

次の通りエディタでファイルを開きます.

```
# emacs /usr/lib/python2.6/site-packages/reportlab-2.5-py2.6-cygwin-1.7.9-i686.egg/reportlab/platypus
```

335 行目を以下のコードに置き換えます.

```
simple = last or abs(extraSpace)<=1e-8 or getattr(line, 'lineBreak', False)
```

3.2 PDF ファイル生成

以下のコマンドを実行してください

```
# rst2pdf -s ja --font-path=~/fonts/ rst2pdf.rst
```

rst2pdf.pdf が作成されれば完了です.

3.3 参考にさせて頂いた Web ページ

- rst2pdf の get_language で発生する問題について
- rebaseall について
- reportlab の修正について
- Creating presentations using restructured text

CHAPTER 4

Sphinx

4.1 Sphinx のインストール

```
# easy_install sphinx
```

4.2 作業フォルダの作成

“sphinx” という名前のフォルダを作成します.

```
# mkdir ~/sphinx
# cd ~/sphinx
```

4.3 初期化

sphinx の初期設定を行います.

```
# sphinx-quickstart
```

質問項目が順番に出てきますので，答えてください．最低限，次の項目を入力してあとはデフォルトでよいでしょう.

- Project name
- Author name(s): User Name
- Project version: 0.1

以下の例では，これらのほか

- Separate source and build directories (y/N) [n]:

のみ「y」と答えています.

```
Welcome to the Sphinx 1.0.7 quickstart utility.

Please enter values for the following settings (just press Enter to
accept a default value, if one is given in brackets).
```

```
Enter the root path for documentation.
> Root path for the documentation [.]:

You have two options for placing the build directory for Sphinx output.
Either, you use a directory "_build" within the root path, or you separate
"source" and "build" directories within the root path.
> Separate source and build directories (y/N) [n]: y

Inside the root directory, two more directories will be created; "_templates"
for custom HTML templates and "_static" for custom stylesheets and other static
files. You can enter another prefix (such as ".") to replace the underscore.
> Name prefix for templates and static dir [_]:

The project name will occur in several places in the built documentation.
> Project name: My first sphinx
> Author name(s): User Name

Sphinx has the notion of a "version" and a "release" for the
software. Each version can have multiple releases. For example, for
Python the version is something like 2.5 or 3.0, while the release is
something like 2.5.1 or 3.0a1.  If you don't need this dual structure,
just set both to the same value.
> Project version: 0.1
> Project release [0.1]:

The file name suffix for source files. Commonly, this is either ".txt"
or ".rst".  Only files with this suffix are considered documents.
> Source file suffix [.rst]:

One document is special in that it is considered the top node of the
"contents tree", that is, it is the root of the hierarchical structure
of the documents. Normally, this is "index", but if your "index"
document is a custom template, you can also set this to another filename.
> Name of your master document (without suffix) [index]:

Sphinx can also add configuration for epub output:
> Do you want to use the epub builder (y/N) [n]:

Please indicate if you want to use one of the following Sphinx extensions:
> autodoc: automatically insert docstrings from modules (y/N) [n]:
> doctest: automatically test code snippets in doctest blocks (y/N) [n]:
> intersphinx: link between Sphinx documentation of different projects (y/N) [n]:
> todo: write "todo" entries that can be shown or hidden on build (y/N) [n]:
> coverage: checks for documentation coverage (y/N) [n]:
> pngmath: include math, rendered as PNG images (y/N) [n]:
> jsmath: include math, rendered in the browser by JSMath (y/N) [n]:
> ifconfig: conditional inclusion of content based on config values (y/N) [n]:
> viewcode: include links to the source code of documented Python objects (y/N) [n]:

A Makefile and a Windows command file can be generated for you so that you
only have to run e.g. 'make html' instead of invoking sphinx-build
directly.
> Create Makefile? (Y/n) [y]:
> Create Windows command file? (Y/n) [y]:

Finished: An initial directory structure has been created.

You should now populate your master file ./source/index.rst and create other documentation
```

```
source files. Use the Makefile to build the docs, like so:
   make builder
   where "builder" is one of the supported builders, e.g. html, latex or linkcheck.
```

4.4 日本語化する設定

設定ファイルをエディタで開きます.

```
# emacs ~/sphinx/source/conf.py &
```

次の通り，language を ja（日本語）にします.

```
# The language for content autogenerated by Sphinx. Refer to documentation
# for a list of supported languages.
language = 'ja'
```

4.5 文書を生成する

4.5.1 HTML を生成する

以下のコマンドで html を生成することができます.

```
# make html
```

ls コマンドで確認してみましょう.

```
# ls ~/spinx/_build/html
```

4.5.2 PDF を生成する

config.py の extensions を次の通り変更します.

```
# Add any Sphinx extension module names here, as strings. They can be extensions
# coming with Sphinx (named 'sphinx.ext.*') or your custom ones.
extensions = ['rst2pdf.pdfbuilder']
```

config.py に次の設定を追加します.

```
-- Options for PDF output --------------------------------------------------

# Grouping the document tree into PDF files. List of tuples
# (source start file, target name, title, author, options).
#
# If there is more than one author, separate them with \\.
# For example: r'Guido van Rossum\\Fred L. Drake, Jr., editor'
#
# The options element is a dictionary that lets you override
# this config per-document.
# For example,
# ('index', u'MyProject', u'My Project', u'Author Name',
#  dict(pdf_compressed = True))
# would mean that specific document would be compressed
# regardless of the global pdf_compressed setting.
```

```
pdf_documents = [
('index', u'MyProject', u'My Project', u'Author Name'),
]

# A comma-separated list of custom stylesheets. Example:
pdf_stylesheets = ['sphinx','ja']

# Create a compressed PDF
# Use True/False or 1/0
# Example: compressed=True
#pdf_compressed = False

# A colon-separated list of folders to search for fonts. Example:
pdf_font_path = ['~/fonts']

# Language to be used for hyphenation support
pdf_language = "ja"

# Mode for literal blocks wider than the frame. Can be
# overflow, shrink or truncate
#pdf_fit_mode = "shrink"

# Section level that forces a break page.
# For example: 1 means top-level sections start in a new page
# 0 means disabled
#pdf_break_level = 0

# When a section starts in a new page, force it to be 'even', 'odd',
# or just use 'any'
#pdf_breakside = 'any'

# Insert footnotes where they are defined instead of
# at the end.
#pdf_inline_footnotes = True

# verbosity level. 0 1 or 2
#pdf_verbosity = 0

# If false, no index is generated.
#pdf_use_index = True

# If false, no modindex is generated.
#pdf_use_modindex = True

# If false, no coverpage is generated.
#pdf_use_coverpage = True

# Documents to append as an appendix to all manuals.
#pdf_appendices = []

# Enable experimental feature to split table cells. Use it
# if you get "DelayedTable too big" errors
#pdf_splittables = False

# Set the default DPI for images
#pdf_default_dpi = 72
```

Makefile を編集します.

```
# emacs ~/sphinx/Makefile &
```

次の設定を追加します.

```
pdf:
        $(SPHINXBUILD) -b pdf $(ALLSPHINXOPTS) $(BUILDDIR)/pdf
        @echo
        @echo "Build finished. The PDF files are in $(BUILDDIR)/pdf.
```

前節で作成した rst2pdf 用スタイルファイル ja.json をコピーします.

```
# cp ~/ja.json ~/sphinx/
```

4.5.3 LaTeX を生成する

docutils を修正します.

```
emacs /usr/lib/python2.6/site-packages/docutils-0.8-py2.6.egg/docutils/writers/latex2e/__init__.py &
```

361 行目を次の通り変更します.

```
def __init__(self, language_code, reporter = None):
```

414 行目以降を次の通り変更します.

```
def get_language(self, language_code = None):
    """Return TeX language name for ‘language_code‘"""
    if language_code == None:
        language_code = self.language_code
```

latex の生成

```
make latex
```

4.5.4 参考にさせて頂いた Web ページ

- Sphinx 1.0
- Sphinx-Users.jp
- Windows へのインストール
- config.py の PDF 用設定
- Makefile の修正

4.6 ToDo

1. テンプレートをカスタマイズしてみる
2. emacs から make できるようにする

README

Hi, everyone!

- one
- two
- tree

public link to this page