Sphinxで売り物の書籍を作ってみた

鹿野 桂一郎
ラムダノート株式会社
k16.shikano@lambdanote.com
@golden_lucky

2017年11月28日
於 SphinxCon 2017

出版社経営、趣味ドキュメントシステム

- SphinxCon2015 で基調講演しました
  - 「ドキュメントシステムはこれを使え 2015 年版」 https://www.slideshare.net/k16shikano/2015-55455604
- ドキュメントシステム好きはだいたい友だち
- Re:VIEW で売り物の本を作ったことはある
  - 「Re:VIEW で売り物の本を作ってみた（InDesign 抜き）」 http://note.golden-lucky.net/2015/06/reviewindesign.html
- でも、Sphinx で本を作ったことはなかった

reStructuredText で HTML の記事が執筆されていたので、そのまま Sphinx で LaTeX を生成して、印刷製本用のデータを制作した

渋川よしき 著
A5 判／ 360 頁／本体 3200 円
ISBN：978-4-908686-03-0

Sphinxのmake latexpdfでPDFを生成してません

- Sphinxディレクトリ下のreST原稿は、HTMLなどの生成には影響を与えず、そのままの形で完全に活用したい
- でも、PDFはSphinxで作りたくない。なぜなら、
  - われわれはTeXにはちょっと詳しいので
  - 図はJPGやPNGではなくPDFを埋め込みたい
  - クラスファイルやスタイルファイルは自前のものを使いたい（後述）
- LaTeX制作環境下でmakeすると、Sphinxディレクトリ下でmake latexして.texを生成し、自前のルーチンでPDFが生成される、というフローを採用

必要だったハック

- 1. デフォルトの LaTeX テンプレートは使いづらい
- 2. 自作の LaTeX スタイルで見た目を変えたい
- 3. ブロック要素内の脚注を特別扱いしたくない
- 4. 書籍の相互参照 ≠ ハイパーリンク
- 5. 整形の自由度が高い表が必要

必要だったハック

1. デフォルトの LaTeX テンプレートは使いづらい
2. 自作の LaTeX スタイルで見た目を変えたい
3. ブロック要素内の脚注を特別扱いしたくない
4. 書籍の相互参照 ≠ ハイパーリンク
5. 整形の自由度が高い表が必要

専用ドキュメントクラスを使いたい、デフォルトのパッケージは使いたくない

- _template/latex.tex_t として、カスタマイズのテンプレートを設定することで解決できた

```
\documentclass[a5,dvipdfmx,uplatex]{lnbook}

\usepackage{sphinx}
\usepackage{progo}
\AtBeginShipoutFirst{\special{pdf:tounicode UTF8-UTF16}}
<%= makeindex %>
\pagestyle{bookheadings}
\usepackage{hyperref}

<%= body %>
<%= atendofbody %>
<%= indices %>
<%= printindex %>
\include{okuduke}

\end{document}
```

「章」以外の構成要素を追加したり、構成要素の順番を変えたりしたい

- 目次どころか \begin{document}までもが <%= body %> の部分に埋め込まれてしまうので、こう書けない！

```
\documentclass[a5,dvipdfmx,uplatex]{lnbook}
...
%%% こう書きたい！（けど、書けない）
%%%
%%% \begin{document}
%%% \include{tobiraura}
%%% \include{preface}
%%% \tableofcontents

<%= body %>
<%= atendofbody %>
<%= indices %>
<%= printindex %>
\include{okuduke}
\end{document}
```

ディレクティブを追加しました

```
def frontmatter(name, arguments, options, content, lineno,
                content_offset, block_text, state, state_machine):
    return [nodes.raw(
        '',
        r"""
\include{tobiraura}
\frontmatter \setcounter{page}{3}
""",
        format='latex')]

def mainmatter(name, arguments, options, content, lineno,
               content_offset, block_text, state, state_machine):
    return [nodes.raw(
        '',
        r"""
\tableofcontents
\mainmatter
""",
        format='latex')]

def setup(app):
    app.add_directive('frontmatter', frontmatter, 1, (0, 0, 0))
    app.add_directive('mainmatter', mainmatter, 1, (0, 0, 0))
```

自在に本を構成できるようになりました

```
Go ならわかるシステムプログラミング
==================================

.. only:: latex

   .. frontmatter::

.. toctree::
   :maxdepth: 2

   preface

.. only:: latex

   .. mainmatter::

.. toctree::
   :maxdepth: 3
   :numbered:

   introduction
   iowriter
   :
   :
   container
```

```
.. only:: latex

   .. appendix::

.. toctree::
   :maxdepth: 3
   :numbered:

   appendix
   appendix_c

.. only:: latex

   .. backmatter::

.. toctree::

   afterword
```

実際の index.rst のようす

「はじめに」を「第０章」にしたくない

- \chapter{はじめに}になる見出しと、\chapter*{はじめに}になる見出しとを、.rst ファイル内で切り換えることは無理っぽい
- しかたがないので、Sphinx が生成した .tex をスクリプトで後処理してから、upLaTeX で PDF 生成

```
tex:
    cd ../; make latex
    cp ../_build/latex/Go.tex book.tex
    gosh script/post-sphinx-latex.scm
```

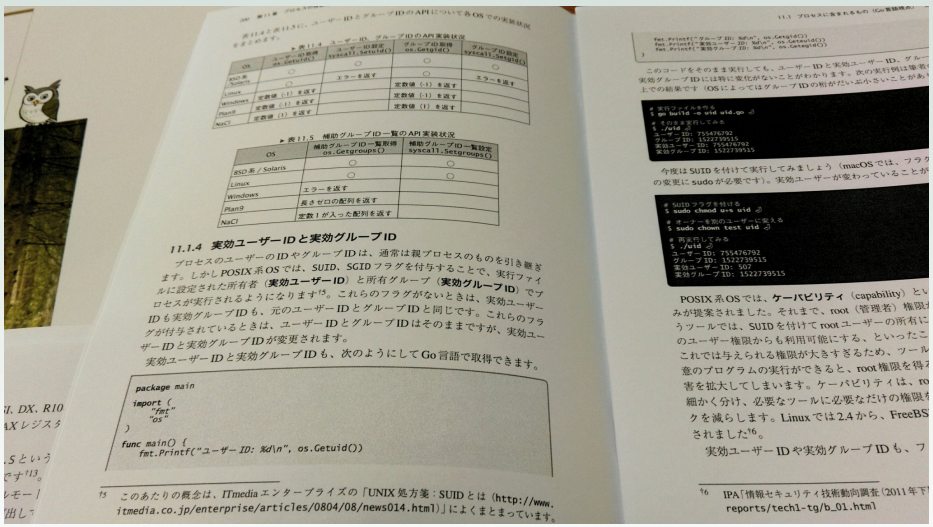
実際の Makefile のようす

必要だったハック

- 1. デフォルトの LaTeX テンプレートは使いづらい
- 2. 自作の LaTeX スタイルで見た目を変えたい
- 3. ブロック要素内の脚注を特別扱いしたくない
- 4. 書籍の相互参照 ≠ ハイパーリンク
- 5. 整形の自由度が高い表が必要

例：コード片と実行画面を区別したい

- `..  code-block::`ディレクティブは、LaTeX の `sphinxVerbatim` 環境内のコンテンツへと一律に変換される
- つまり、PDF での見た目をスタイルでスイッチできない（HTML では `:class:` オプションで実現可能）



好きな環境でラップできるようにしよう

- code-block の改造はつらいので TEX でカバー

```
.. customenv:: execquote

   .. code-block:: sh

      # apt-get install strace ⏎
```

↓ make latex

```
\begin{customadmonition}{execquote}

\def\sphinxLiteralBlockLabel{\label{\detokenize{systemcall:id66}}
\begin{sphinxVerbatim}[commandchars=\\\{\}]
\PYG{c+c1}{\PYGZsh{} \color{white}apt\PYGZhy{}get install strace \crmark{}}
\end{sphinxVerbatim}
\let\sphinxVerbatimTitle\empty
\let\sphinxLiteralBlockLabel\empty
\end{customadmonition}
```

issue を出した

- `.rst` 内でこんな入れ子を書くのは筋悪
- LaTeX 出力で `:class:` オプションの値が捨てられてるけど、この値を環境名にしてディレクティブの出力がラップされてくれれば、`.sty` でユーザがスタイルを指定できる
  - HTML 出力だと `<div>` の `class` 値になるので CSS でスタイル付けできる、という話と同じです
- 「LaTeX representation for :class: (of like ‘..code-block::‘ or ‘..note::‘) #4010」
  https://github.com/sphinx-doc/sphinx/issues/4010

必要だったハック

- 1. デフォルトの LaTeX テンプレートは使いづらい
- 2. 自作の LaTeX スタイルで見た目を変えたい
- **3. ブロック要素内の脚注を特別扱いしたくない**
- 4. 書籍の相互参照 ≠ ハイパーリンク
- 5. 整形の自由度が高い表が必要

LaTeX の脚注は透過的にうまく扱えない

- **Sphinx でも苦労のあとが見える** (footnotehyper-sphinx.sty)
- **Sphinx 提供のスタイル以外を使う場合、それが仇になる**

```
def visit_collected_footnote(self, node):
    # type: (nodes.Node) -> None
    self.in_footnote += 1
    if 'footnotetext' in node:
        self.body.append('%%\n\\begin{footnotetext}[%s]'
                         '\\sphinxAtStartFootnote\n' % node['number'])
    else:
        if self.in_parsed_literal:
            self.body.append('\\begin{footnote}[%s]' % node['number'])
        else:
            self.body.append('%%\n\\begin{footnote}[%s]' % node['number'])
        self.body.append('\\sphinxAtStartFootnote\n')
```

writers/latex.py のようす

TeX のふつうの脚注に戻しました

```
def visit_collected_footnote(self, node):
    # type: (nodes.Node) -> None
    self.in_footnote += 1
    if 'footnotetext' in node:
        self.body.append('\\footnotetext{')
    else:
        if self.in_parsed_literal:
            self.body.append('\\footnote{')
        else:
            self.body.append('\\footnote{')

def depart_collected_footnote(self, node):
    # type: (nodes.Node) -> None
    if 'footnotetext' in node:
        self.body.append('}\\ignorespaces ')
    else:
        if self.in_parsed_literal:
            self.body.append('}')
        else:
            self.body.append('}')
    self.in_footnote -= 1
```

必要だったハック

- 1. デフォルトの LaTeX テンプレートは使いづらい
- 2. 自作の LaTeX スタイルで見た目を変えたい
- 3. ブロック要素内の脚注を特別扱いしたくない
- **4. 書籍の相互参照 ≠ ハイパーリンク**
- 5. 整形の自由度が高い表が必要

人間が参照先を目で探すための文字列

- :doc:や:ref:だと、章や節のタイトルしか出力しない
- 人間が参照先にたどり着くためには、章や節の「番号」のほうが重要だったりする
- かといって、:numref:を使うためにラベルを追加するのは面倒

独自に:numdoc:ロールを用意しました

```
def numdoc_role(name, rawtext, text, lineno, inliner, options={},
                content=[]):
    :
    pnode = nodes.inline(rawtext, title, classes=['xref','doc'])
    pnode['reftarget'] = target
    return [pnode], []

def visit_inline(self, node):
    classes = node.get('classes', [])
    if classes in [['menuselection'], ['guilabel']]:
        :
    elif classes in [['xref','doc']] and not self.in_title:
        self.body.append(ur'第\DUrole{%s}{' % ','.join(classes))
        self.context.append(u'}章')
    elif classes and not self.in_title:
        :
```

- いにしえの互換性を維持するための機能が sphinx.sty にあったので、それを利用できた

```
\expandafter\def\csname DUrole\detokenize{xref,doc}\expandafter\endcsname
  \expandafter{\csname ref\endcsname}
\providecommand*{\DUrole}[2]{%
    :
```

相互参照UIの落としどころ難しい

- :numdoc:と:doc:を併記するのは、ちょっといけてない
- :numref:と:ref:も併記するしかないし、まあいいかな
- そもそも現在のSphinxでは参照アンカーをすべてSphinx側で事前に解決し、その結果を\hyperrefのオプション引数に埋め込んでしまう。でも、節番号の相互参照のような「LaTeXが得意なこと」は、LaTeXにやらせたい
  - LaTeXには採番機能があるけど、HTMLやEPUBに対応するには上位で採番機能が必要なんだよな
  - LaTeXにしても、章や節のタイトル再利用は大変なので、どうがんばっても痒みは残りそうではある…
- HTMLやEPUBでは無意味だけど、PDFではLaTeXの「ページ参照」が使いたい

ページ参照といえば、索引

- 索引のアンカーは、「項目が出現するセクション」ではなく、「項目が出現するページ」に必要
- 必然的に、`..  index::`ディレクティブが使えない……

独自に:tex:ロールを用意しました

```
def tex_role(name, rawtext, text, lineno, inliner, options={}, content=[]):
    text = utils.unescape(text, restore_backslashes=True)
    has_explicit, texsnipet, target = split_explicit_title(text)
    pnode = nodes.raw(rawtext, texsnipet, format='latex')
    return [pnode], []

def setup(app):
    app.add_role('tex', tex_role)
```

- 局所的な TeX ソースの埋め込みによるスタイル指定にも使えて、やりたい放題！

独自に:tex:ロールを用意しました

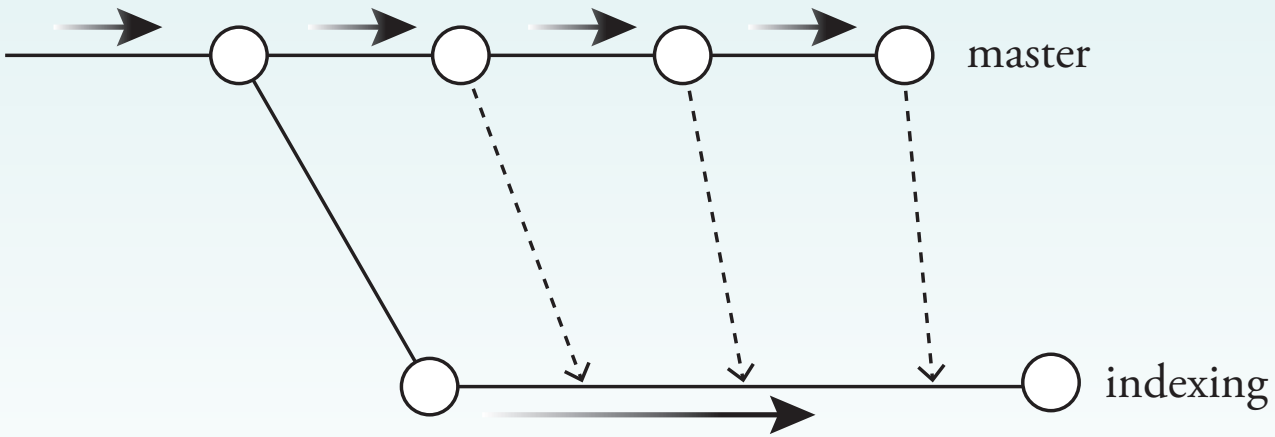
```
def tex_role(name, rawtext, text, lineno, inliner, options={}, content=[]):
    text = utils.unescape(text, restore_backslashes=True)
    has_explicit, texsnipet, target = split_explicit_title(text)
    pnode = nodes.raw(rawtext, texsnipet, format='latex')
    return [pnode], []

def setup(app):
    app.add_role('tex', tex_role)
```

- 局所的な TeX ソースの埋め込みによるスタイル指定にも使えて、やりたい放題！
- でも、原稿がこうなるの、いやですよね？

```
物理的なストレージと対応しない、仮想的なファイルシステム\ :tex:`\index{ファイルシステ
↪ム!仮想的な}`\ もあります。
たとえば、Unix系のシステムにおける\ ``/proc``\
↪:tex:`\index{proc@\texttt{/proc}}`\ は仮想的なファイルシステムの一例です。
```

索引専用の Git ブランチを使う

- 索引タグは、編集者だけが煩わされればよい
- indexing ブランチに master から cherry-pick して make したものを最終入稿データとする
- 索引が組まれた LaTeX 用ファイルのみを master にコピーするという運用もありえる



https://www.slideshare.net/k16shikano/imybp-light より

必要だったハック

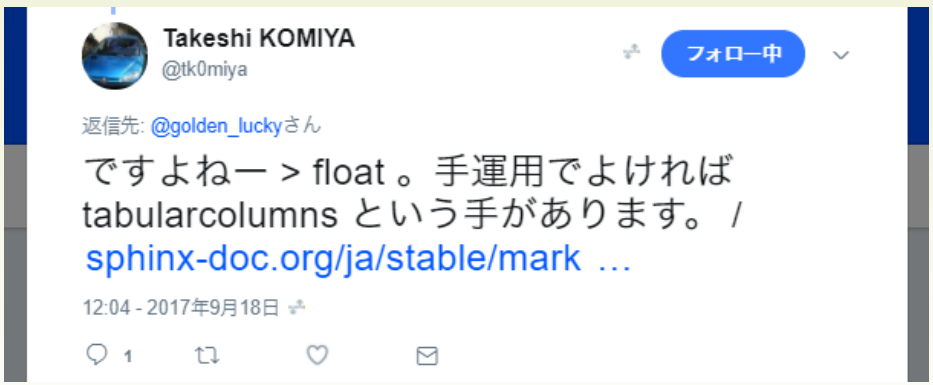
- 1. デフォルトの LaTeX テンプレートは使いづらい
- 2. 自作の LaTeX スタイルで見た目を変えたい
- 3. ブロック要素内の脚注を特別扱いしたくない
- 4. 書籍の相互参照 ≠ ハイパーリンク
- **5. 整形の自由度が高い表が必要**

LaTeX の表は自動できれいには組めない

- Sphinx では、表スタイルの自動解決を LaTeX（tabulary パッケージ）にやらせる戦術
- しかし、列幅や列のスタイルを LaTeX 側で完全に自動で「いい感じ」にするのは無理
- そのため、個人的には、各セルを `\multicolumn` 化して完全に独立制御する戦術をとっている
- せめて colspec を `.rst` で指定できれば……

LaTeX の表は自動できれいには組めない

- Sphinx では、表スタイルの自動解決を LaTeX（tabulary パッケージ）にやらせる戦術
- しかし、列幅や列のスタイルを LaTeX 側で完全に自動で「いい感じ」にするのは無理
- そのため、個人的には、各セルを `\multicolumn` 化して完全に独立制御する戦術をとっている
- せめて colspec を `.rst` で指定できれば……



https://twitter.com/tk0miya/status/909614165852954627

まとめ

- **Sphinx で困ったら日本語でツイートすればいい**

まとめ

- **Sphinx で困ったら日本語でツイートすればいい**
- **ある程度リッチな紙の本をテキストから make いっぱつで作るには、所与のフォーマットと拡張のための機能が Sphinx くらい充実していても、手をかけないといけない部分がけっこうある**
- **ラムダノート株式会社は出版を中心として技術文書まわりのお手伝いをいろいろする会社です**
  - https://lambdanote.com

Lambda Note