第 77 回 東京エリア Debian 勉強会資料

月刊 Deb専

日本唯一のDebian専門月刊誌

2011 年 6 月 18 日

特集1: DebianでXSLTを使ってみた

特集2: SphinxとDoxygenを使ってみた

1 Introduction

上川 純一

今月の Debian 勉強会へようこそ。これから Debian の世界にあしを踏み入れるという方も、すでにどっぷりとつかっているという方も、月に一回 Debian について語りませんか?

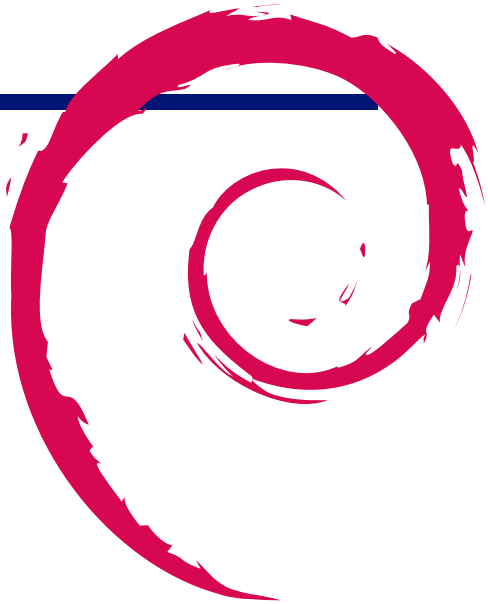
Debian 勉強会の目的は下記です。

- Debian Developer (開発者) の育成。
- 日本語での「開発に関する情報」を整理してまとめ、アップデートする。
- 場の提供。
  - 普段ばらばらな場所にいる人々が face-to-face で出会える場を提供する。
  - Debian のためになることを語る場を提供する。
  - Debian について語る場を提供する。

Debian の勉強会ということで究極的には参加者全員が Debian Package をがりがりと作るスーパーハッカーになった姿を妄想しています。情報の共有・活用を通して Debian の今後の能動的な展開への土台として、「場」としての空間を提供するのが目的です。

デビアン勉強会

目次



2 事前課題

岩松 信洋

今回の事前課題は以下です:

1. 最近 Debian で自分がやったことや興味のあることを教えてください。

この課題に対して提出いただいた内容は以下です。

2.1 やまだ

最近新しいカーネルを自分でビルドすることが再び増えてきたので、その周りでごそごそやってます。

1. git でタグを打って
2. そのタグをバージョンにして
3. ビルドして
4. テスト VM に導入して
5. ついでに配布サーバに設置

の自動化とかやりました。

興味というか次に調べたいのは *-module-source なカーネルモジュールパッケージの作り方とか DKMS の使い方。普通のパッケージと違う点が多々ありそう。

2.2 野首

ホームの MH フォルダを外から見れるよう uw-imapd を入れました。インストールして debconf に答えるだけで SSL ready な imap 環境になりました。

2.3 MATOHARA

2010 年 11 月の勉強会資料を見ながら nilfs2 をバックアップディスクに設定してみました．リサイズ機能も来たようなので試してみたいです．- [PATCH 0/4] nilfs2 resize support – Linux NILFS Development `http://www.spinics.net/lists/linux-nilfs/msg00869.html`

2.4 hattorin

バングラデシュで Debian インストールして、ネットワークの監視系ツールをいろいろとセットアップしてきました。今は Debian on Squeeze で TokyoTyrant の KeyValue を使い、特定の問題を早く計算するためにネットワークを使って計算を早くするような仕組み作りをしています。

2.5 キタハラ

実家でプリンタの設定したり、スキャナーの設定に失敗してました。

2.6 koedoyoshida

最近 Debian で自分がやったこと

- ようやく、メイン環境を Squeeze に update wide-dhcp が update できずにはまったが、下記を見て解決。 `http://www.flcl.org/~takasugi/tdiary-org/?date=20061023`
- OSC 仙台に参加、出展。

2.7 emasaka

sid でちょっとはまったとこについて、 GitHub で upstream に簡単なパッチを pull request したら、そのバージョンが数日後に sid に降りてきました。

2.8 dictoss(杉本　典充)

最近やっていることは kfreebsd を常用していること、 Debian を開発環境として gtk アプリを試作していること。今後は kfreebsd で IS03 を使ってテザリング、 klinux より kfreebsd の方がおすすめといえる有利分野を見つけることをやりたい。

2.9 なかおけいすけ

興味があること： DebianLive 最近ネカフェで一夜を明かしたのですが、ネカフェの PC にコンパイラが入ってなくて困ったので、 USB や SD カードにインストールした Debian を持ち歩いているとハッピーになれるのではないかと。

2.10 吉野 (yy_y_ja_jp)

バグレポートと DDTSS/DDTP ぐらいでしょうか.

2.11 henrich

いくつかパッケージやメッセージをルーティンの更新しました。

2.12 Osamu MATSUMOTO

- インフラ, サーバ管理の自動化
  自動インストール, 構成管理, コンフィグ投入, 監視のまでをラフに綺麗に繋ぎたい。 Debian 的な良い組み合わせあったら教えてください。 (cobbler+ puppet/cfengine+ nagios + なんか webcgi 的な)

2.13 まえだこうへい

- *diag シリーズ (http://blockdiag.com/) の deb パッケージ化中。
- さくらの VPS を先月契約して、 lxc & Squeeze で開発&検証環境に。
- あらきさんメンテナンスの Debian の AMI 使って AWS でごにょごにょと。

2.14 yamamoto

- 最近 Debian で自分がやったこと
  ポチポチと公式パッケージのリビルドをしました。
- 興味のあること
  移植。

2.15 岩松 信洋

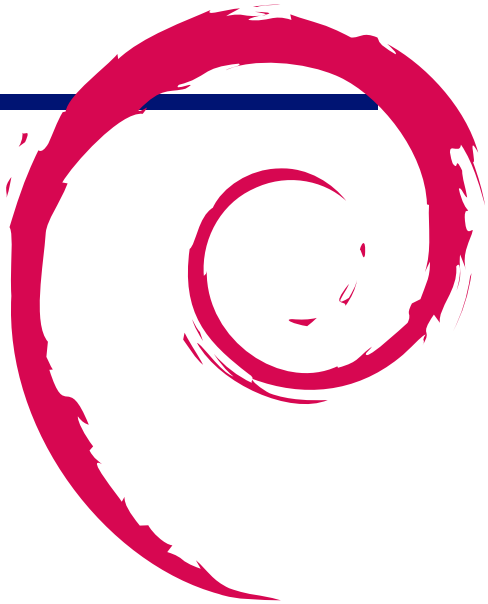
- SH4 buildd のメンテ。
- libpng15 の experimental へのアップロードと transition 作業。
- スポンサーのパッケージチェック、アップロードなど。

2.16 野島　貴英

- python で gnome-notify めがけて再生中のデータのメタデータ送る「 できるだけ簡単にできる」 totem のプラグイン書いてみた。 `http://d.hatena.ne.jp/nozzy123nozzy/20110502/1304322969`
- sid の alsa-lib,alsa-utils,alsa-driver の解析中。(bluetooth ヘッドフォン出力相手に、 alsa のみで、演奏中の音を capture したいし、音が出る複数のアプリの音を mix して出力したいっ)
- sid の linux-image-2.6.39-2-686-pae にて、 multi-state な usb 装置に eject コマンド発行すると高い確率で Oops する件のデバッグを誰かがパッチ出すまでやり中。(いったい誰だ、 dpt ぶっ壊すのは...)

2.17 上川純一

- xslt のツールチェインをいろいろといじってみたり。
- Javascript のコードを書いてみたり。



3 最近の Debian 関連のミーティング報告

岩松 信洋

3.1 東京エリア Debian 勉強会 76 回目報告

東京エリア Debian 勉強会 76 回目は新大久保駅の近くにある戸山生涯学習館で行いました。上川さんが apache2 のモジュールの作成についてお話しし、 Debian の apache2 モジュール作成のはまりどこについて盛り上がりました。
Nifty さんが Nifty Cloud に Debian 使えるようにするということで、どのようなイメージにしたらいいのか、どのようなバックエンドが必要なのか議論しました。 Nifty さんでそろそろ Debian が使えるようになっていると思います(たぶん）。岩松が Debian での m68k 開発についての状況と方法を話しました。 Debian では実機はあまり使われておらず、Aranym というエミュレータ上で開発を行っているとのこと。 Debian で m68k が再度リリースされる日はくるのか楽しみです。山本さんが ppc64 ポーティングについて報告しました。前回の宿題では powerpcspe とのベンチマークを取ってきて、 ppc64 と比べてどのようなメリットデメリットがあるのかを見てみようということでしたが、なぜか ppc とのベンチを取ってきたので再度宿題になりました。パッケージは揃っているので家に ppc64 マシンが眠っている人は試してみてはいかがでしょうか。

3.2 OSC 2011 Hokkaido 出張 Debian 勉強会報告

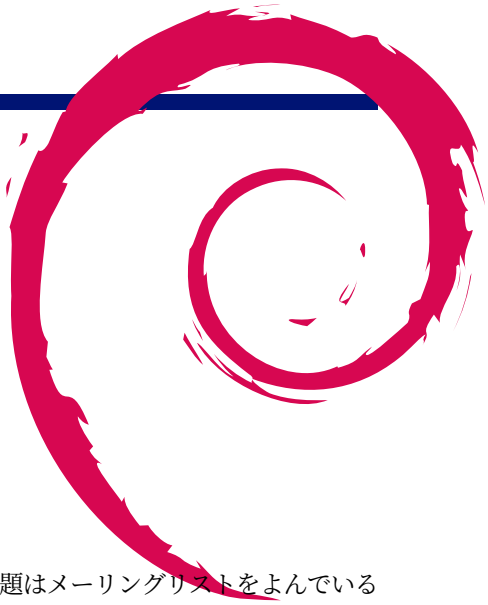
2011 年 6 月 11 日 (土) に北海道札幌市でオープンソースカンファレンス 2011 北海道が開催されました。
Debian 勉強会ではタイトル「 Debian 6.0(squeeze) & Debian Update」として佐々木洋平さんがセッションを行いました。セッションには 14 名が参加し、質問ではバグ報告の手順を確認する質問が出されました。
GPG キーサインパーティの参加者はコーディネーターを含めて 3 名であり、北海道の方々へ GPG キーの啓蒙活動を引き続き行っていく必要がありそうです。

3.3 OSC 2011 Sendai 出張 Debian 勉強会報告

2011 年 5 月 21 日 (土) に宮城県仙台市でオープンソースカンファレンス 2011 仙台が開催されました。
Debian 勉強会では吉田@板橋が展示で出展しました。

1.「 東京エリア Debian 勉強会/関西 Debian 勉強会」の紹介
2.「 あんどきゅめんてっどでびあん (Debian 勉強会資料のサマリ)」の展示、販売
3.「 Debian Squeeze の展示」具体的には「 Debian GNU/kFreeBSD」の展示

主に上記をおこないました。メインホール入り口正面という配置にも恵まれ、多くの方に立ち寄って頂きました。



4 Debian Trivia Quiz

岩松 信洋

ところで、みなさん Debian 関連の話題においついていますか? Debian 関連の話題はメーリングリストをよんでいると追跡できます。ただよんでいるだけではりあいがないので、理解度のテストをします。特に一人だけでは意味がわからないところもあるかも知れません。みんなで一緒に読んでみましょう。

今回の出題範囲は `debian-devel-announce@lists.debian.org` や `debian-devel@lists.debian.org` に投稿された内容と Debian Project News からです。

問題 1. alioth.debian.org が 2 台に分かれました。そのサーバ名は?

☐ A vasks.debian.org と wagner.debian.org
☐ B volks.debian.org と don.debian.org
☐ C dennys.debian.org と gusto.debian.org

問題 2. 現在行われている Perl transition の Perl バージョンは?

☐ A 5.12
☐ B 5.13
☐ C 5.14

問題 3. プライマリミラーサーバが新しく追加された国は?

☐ A チュニジア
☐ B 中国
☐ C マダガスカル

5 Debian JP 定例会議処理系に XSLT を使ってみた

上川純一

5.1 背景

Debian 勉強会の企画会議は IRC を中心として 2006 年に開始し、 Debian JP の定例会議として今も続いています。当初は決定事項などについてテキストファイルでまとめるという形をとっていました。 IRC でより効率よく議論する方法を模索した結果、議論しながら議事録を編集するというスタイルが確立し、それを支援するためのツールを整備しました。

議事録のソースは議長が XML で記述して、議論の最中は非同期に Javascript で内容が更新される HTML ファイルを利用します (世間一般では AJAX 的とでもよぶようです)。

IRC での定例会議の議論の前と後には議事案と議事録をメーリングリストにおくっています。メーリングリストにメールでなげる際には、テキストフォーマットにして送っています。

あと、現在用途がないですが、 LaTeX 経由で PDF 形式での出力などもサポートしています。

現在の実装は歴史的な経緯により XML の処理系は dancer-xml ライブラリと boost を利用した C++ のプログラムになっています。 dancer-xml[4] は 10 年前に若気のいたりで実装した XML 風文書のパーサーです。一部エンティティーまわりなど真面目に実装していない部分があるため、適切な処理がなされていないことがありますが、僕の好みに空白文字処理はチューニングされており快適です。

今回の挑戦は、独自 C++ コードベースを XSLT にのせ替えてみるという挑戦です。

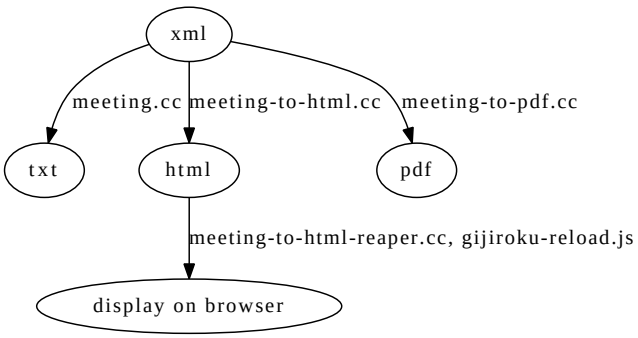


図 1 IRC 会議システムのデータ形式の概要

5.2 XSLT ってどんな言語?

XSLT は XML で記述する XML 処理言語です。

XSLT 規格には 1999 年に策定された XSLT 1.0 と、 2007 年ごろの XSLT 2.0 があります。今回は実装が十分枯れて

いると思われる XSLT 1.0 の処理系を採用しました。 XSLT2.0 の Debian で利用できる実装としては libsaxonb-java があるようですが、今回は調査していません。

プログラムを書くときにすべき文書としては、 XSLT 自体の 1999 年に策定された規格 [2] と、 XSLT の中で記述できる XPATH の規格 [3] を参照するとよいでしょう。

XSLT だけではあまり高度なプログラミングはできないんじゃないかと思われるかもしれませんが、関数型言語として十分な機能を提供できる力はあるようです [1]。

5.3 Debian で利用可能な XSLT 処理系

Debian で幅広く使われていて安定しているとおもわれるのと、簡単に利用できるという理由で処理系として xsltproc を採用しました。

Debian での xsltproc のインストールは簡単

```
$ apt-get install xsltproc
```

コマンドラインで以下のように実行すると標準出力に処理済み XML が出力されます。

```
$ xsltproc [スタイルシート] [処理する XML ファイル]
```

5.4 具体例:HTML

それでは、 HTML 出力の場合を見てみましょう。 meetinglog:html.xsl です。 XML 文書から HTML 文書を生成するにはそれなりに便利な言語です。

前半のコードをそのまま掲載します。これは、 XML ドキュメント全体にマッチするルールを記述しはじめるまでの部分です。 XML 名前空間として、デフォルトを HTML、 xsl を xslt の名前空間に割り当てています。

xml:output で出力形式を HTML と指定することで XML ヘッダが出力されずに便利です。

```
<?xml version="1.0"?>
<!DOCTYPE xsl:stylesheet>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns="http://www.w3.org/1999/xhtml">
  <xsl:output method="html" />
  <xsl:template match="/">
    <html>
      <head>
```

他に XSLT の特徴的なところは、 value-of で値をとってきています。 XPATH の書式で指定していますが、

```
        <h1><xsl:value-of select="meetinglog/head/title"/></h1>
```

は、 XML 文書の以下のようなエレメントに入っている値を抽出します。

```
<meetinglog>
  <head>
    <title>タイトル</title>
  </head>
</meetinglog>
```

議事録の場合のメインループは、各議事に対しての処理です。 xsl の xsl:for-each をつかい、 XML のエレメントノードの数だけループします。 HTML タグはそのまま出力されますが、もし HTML のアトリビュートなどを XSLT で生成したい場合は、 xsl:element を使ってエレメントを生成します。

position() 関数は現在のエレメント番号をくれるのでこういう場合に便利です。

```
          <xsl:for-each select="meetinglog/body">
             <tr>
               <th>
                 <xsl:element name="a">
                   <xsl:attribute name="href">#gian<xsl:value-of select="position()" /></xsl:attribute>
                 </xsl:element>
                 議案<xsl:value-of select="position()" />
               </th>
               <td class="bodytitle">
                 <xsl:value-of select="./title" />
               </td>
....
</xsl:for-each>
```

5.5 具体例:Text 出力

Text 出力の場合もみてみましょう。 meetinglog:txt.xsl テキスト出力をしようとしはじめると若干苦しくなってきます。できないわけではないのですが、空白文字の処理のルールを僕がいまいち理解できていないのと、コードがそのままテンプレートとして出力されるのでインデンテーショ ンが適切にできないのがつらいところです。

xsl:output で出力がテキスト形式であると指定すると XML ヘッダが出力されず便利です。

ヘッダ部分で、毎回 xsl:text で改行などを入力するのが面倒なので、 ENTITY を定義して省略できるようにしています。この記法が正しいのかどうかは不明です。

```
<?xml version="1.0"?>
<!DOCTYPE xsl:stylesheet [
<!ENTITY space  "<xsl:text xmlns:xsl='http://www.w3.org/1999/XSL/Transform'> </xsl:text>">
<!ENTITY indent "<xsl:text xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>  </xsl:text>">
<!ENTITY cr     "<xsl:text xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
</xsl:text>">]>
  <xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/1999/xhtml">
  <xsl:output method="text" />
  <xsl:template match="/">
    <xsl:text>------------------------------------------------------------------------
概要
------------------------------------------------------------------------
```

本文のコアとなる本文の内容ですが、読めたものではないです。悩んだところとしては、文章が空白かどうかチェックするのに string-length(normalize-space()) をつかっていて、それがいまいちただしいのかどうか自信がないところ。

```
    <xsl:for-each select="meetinglog/body">
------------------------------------------------------------------------
[<xsl:value-of select="position()" />.&space;<xsl:value-of select="./title" />]
------------------------------------------------------------------------

目的: &cr;<xsl:value-of select="./aim" />&cr;
&cr;<xsl:if
 test="string-length(normalize-space(./previous))>1"
 >前回までの経緯:&cr;<xsl:value-of select="./previous"
 disable-output-escaping="yes" />&cr;&cr;</xsl:if>
<xsl:if test="string-length(normalize-space(./discussed))>1"
 >議論:&cr;<xsl:value-of select="./discussed"
 disable-output-escaping="yes" />&cr;&cr;&cr;
</xsl:if>
</xsl:for-each>
```

残念ながら C++ で実装していた文字をメールの 70 文字幅くらいにきれいにまとめるというロジックが欠落しています。めんどくさすぎる。

5.6 具体例:LaTeX

LaTeX 出力をみてみましょう。 meetinglog:latex.xsl

ヘッダ部分はどうぜ LaTeX のヘッダなのと何度もでてきているのでメインループだけ。

```
    <xsl:for-each select="meetinglog/body">

      \discussion{<xsl:value-of select="./title" />}{<xsl:value-of
select="./aim" />}{<xsl:value-of
select="translate(./previous,'#&amp;',’--’)"
disable-output-escaping="yes" />}{<xsl:value-of
select="translate(./discussed,'#&amp;',’--’)"
disable-output-escaping="yes" />}
    </xsl:for-each>
```

個人的な感想ですが、自分で書いておきながら後で読み返す気力が沸きません。

現在実装できていない点として、 LaTeX で使えない文字列#<>&などの文字列のエスケープがあります。今はハイフンに変更してお茶を濁しています。

XPATH には文字列置換のための transform() 関数がありますが、一文字を一文字に置換することしかできません。今回行いたいのは一文字を複数文字に置換することなのでそれでは機能が不十分です。

5.7 仮の定量的な比較

現状すべての機能をおきかえているわけではないので、妥当な比較ではないですが、 C++ の処理と XSLT のコードの比較をしてみると (1)、 XSLT のほうが行数は少ないことがわかります。

表 1 lines of code for each implementation

	c++	xslt
txt	151	53
html	157	99
latex(PDF)	158	87

5.8 結論

XSLT を使うことでメンテナンスする行数は少なくなります。しかし、 XPATH / XSLT により提供されている機能が制限されているため、その中で実現しにくい機能についてはがんばるか提供を諦めるのか、難しい判断を迫られます。

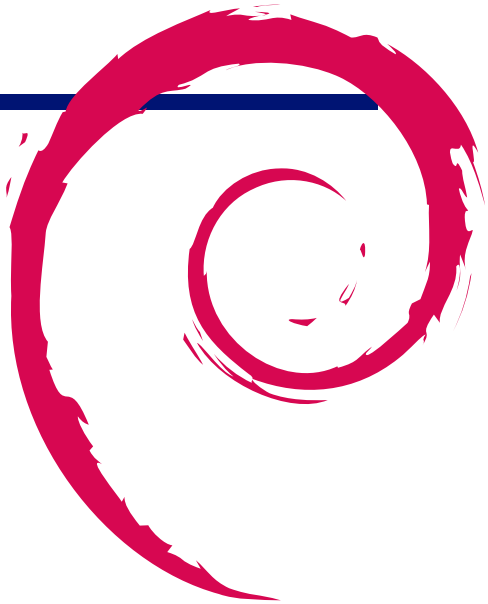
参考文献

[1] Dimitre Novatchev, “Functional programming in XSLT using the FXSL library,” Extreme Markup Languages 2003.

[2] James Clark, “XSL Transformations (XSLT) Version 1.0,” W3C Recommendation 16 November 1999. http://www.w3.org/TR/xslt

[3] James Clark, Steve DeRose, “XML Path Language (XPath) Version 1.0,” W3C Recommendation 16 November 1999. http://www.w3.org/TR/xpath/

[4] Junichi Uekawa, “dancer-xml - Simple non-comformant XML parsing library,” 2000. http://www.netfort.gr.jp/~dancer/software/dancer-xml.html



6 Debian で Sphinx と Doxygen を使ってみた

まえだこうへい

6.1 最近の流行りのようです

Python 関連のプロジェクトやエンジニアを中心に最近流行っているようです。Sphinx-Users.jp のサイトを見る[*1]と、2011 年 6 月現在、Sphinx のデフォルトテーマだけでなく、カスタムテーマやオリジナルテーマを使った、50 弱の日本語のサイトが紹介されています。

6.1.1 reST と Sphinx の概要

Sphinx は reST(reStructuredText) という軽量マークアップ言語で書いたソースを様々なフォーマットのドキュメントに変換・生成するためのツールです。出力可能なフォーマットには、HTML、LaTeX、PDF、ePub、man、平文テキスト、JSON などがあります。

6.1.2 reST のサンプル

試しに、東京エリア Debian 勉強会のページを reST で書くとこんな感じになります。

```
========================
 東京エリア Debian 勉強会
========================


背景
====

2005 年当初、東京近辺で、類似の勉強会は存在していませんでした。
Debian について語る場所を提供するため、 Debian 勉強会を開催します。
この勉強会では毎回事前課題を設定しています。
その課題を提出することが参加の条件です。
参加する方は宴会の都合もありますので、事前に登録してください。
また、当日は Debian についての知識に関した簡単な試験を実施するので、
勉強会の会場には筆記用具を持参ください。

現在、 Debian 勉強会は
‘Debian JP Project <http://www.debian.or.jp/>‘_
のメンバーが Debian JP の公式なイベントとして運営しています。

次回の勉強会
============

* ‘2011 年 6 月勉強会（第 77 回東京エリア Debian 勉強会）<2011-06.html>‘_
* ‘第 48 回関西 Debian 勉強会 <http://wiki.debian.org/KansaiDebianMeeting20110626>‘_
* ‘毎週開催のハックカフェ <hackcafe.html>‘_
(snip)
```

具体的な書式については、Sphinx-Users.jp のドキュメント[*2]を参照してください。

[*1] http://sphinx-users.jp/example.html
[*2] http://sphinx-users.jp/doc.html

6.2 Sphinx を使うきっかけ

graphviz の dot 言語と似た書式でブロック図を生成できる blockdiag シリーズ[*3]という python で書かれたツールがあります。最近、岩松さんにスポンサーをお願いして、これらの Debian パッケージ化を行っています。これらの Sphinx 拡張機能 (spyhinxcontrib-blockdiag など) を使うと、 Sphinx で生成するドキュメントの中にブロック図を埋め込むことができます。この blockdiag シリーズが便利なので、 Sphinx を使いだしたようなものです。

また、仕事では基本的に MS Office、とくに Excel や PowerPoint での文書作成がほとんどなのですが、今期の最初に「 もう MS Office なんてでやってられっかー、 Sphinx で作ろうぜ! 」と、プロジェクト内で提案して使い始めました。私個人で作る分には LaTeX でも良いのですが、他の二人は Windows しか普段触ったことがない上、文書と言えば上述のとおり、 Excel か PowerPoint、という状態です。まったく使ったことがない人に LaTeX 文書を作成させるのは敷居が高すぎます。しかし、 reST & Sphinx なら割と簡単に入門できる上、 Windows との共同作業の環境を整えるのもメンドイけど (LaTeX 環境を整えるよりも) 楽だった、という経緯です。[*4]

6.3 Debian で使ってみる

試しに先ほどの東京エリア Debian 勉強会のホームページを reST で書いたものを Sphinx で管理してみましょう。

まずは python-sphinx パッケージをインストールしておきます。

```
$ sudo apt-get install python-sphinx
```

emacs を使う場合は、 python-docutils パッケージをインストールしておけば、拡張子が rst か rest の場合、 rst.el によって自動的に ReST モードになります。

```
$ sudo apt-get install python-docutils
```

Sphinx プロジェクトを作ります。プロジェクト用のディレクトリを作ります。

```
$ mkdir tokyodebian
$ cd tokyodebian
```

作成したディレクトリに移動して、 `sphinx-quickstart` コマンドを実行します。

```
$ sphinx-quickstart
```

このコマンドを実行すると対話形式で聞かれます。 html を生成するので、 Project Name, Author name(s), Project Version 以外はデフォルトのまま (Enter を押下) で良いでしょう。 (表 2)

先ほどの tokyodebian.rst(および、 hackcafe.rst, 2011-06.rst) をコピーします。

```
$ cp -i ~/*.rst .
```

自動的に生成される index.rst にこれらを追記します。[*5]

[*3] `http://blockdiag.com/`

[*4] Windows は改めてマンドイと思いました。 `http://d.hatena.ne.jp/mkouhei/20110521/1305905297`

[*5] 拡張子不要です。

表 2　sphinx-quickstart の設定項目

設定項目	デフォルト値	設定例
Root path for the documentation	.	デフォルト
Separate source and build directories (y/N)	n	デフォルト
Name prefix for templates and static dir	_	デフォルト
Project name:		Tokyo Debian Meeting
Author name(s)		Debian JP Project
Project version		1.0
Project release	1.0	デフォルト
Source file suffix	.rst	デフォルト
Name of your master document (without suffix)	index	デフォルト
Do you want to use the epub builder (y/N)	n	デフォルト
autodoc: automatically insert docstrings from modules (y/N)	n	デフォルト
doctest: automatically test code snippets in doctest blocks (y/N)	n	デフォルト
intersphinx: link between Sphinx documentation of different projects (y/N)	n	デフォルト
todo: write “todo” entries that can be shown or hidden on build (y/N)	n	デフォルト
coverage: checks for documentation coverage (y/N)	n	デフォルト
pngmath: include math, rendered as PNG images (y/N)	n	デフォルト
jsmath: include math, rendered in the browser by JSMath (y/N)	n	デフォルト
ifconfig: conditional inclusion of content based on config values (y/N)	n	デフォルト
viewcode: include links to the source code of documented Python objects (y/N)	n	デフォルト
Create Makefile? (Y/n)	y	デフォルト
Create Windows command file? (Y/n)	y	デフォルト

```
$ sensible-editor index.rst
---
.. Tokyo Debian Meeting documentation master file, created by
   sphinx-quickstart on Fri Jun 17 13:39:53 2011.
   You can adapt this file completely to your liking, but it should at least
   contain the root `toctree` directive.

Welcome to Tokyo Debian Meeting's documentation!
================================================

Contents:

.. toctree::
   :maxdepth: 2

   tokyodebian  ←追加
   hackcafe  ←追加
   2011-06  ←追加

Indices and tables
==================

* :ref:`genindex`
* :ref:`modindex`
* :ref:`search`
```

コンパイルします。

```
$ make html
sphinx-build -b html -d _build/doctrees   . _build/html
Running Sphinx v1.0.7
loading pickled environment... done
building [html]: targets for 4 source files that are out of date
updating environment: 0 added, 4 changed, 0 removed
reading sources... [ 25%] 2011-06
reading sources... [ 50%] hackcafe
reading sources... [ 75%] index
reading sources... [100%] tokyodebian

looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [ 25%] 2011-06
writing output... [ 50%] hackcafe
writing output... [ 75%] index
writing output... [100%] tokyodebian

writing additional files... genindex search
copying static files... done
dumping search index... done
dumping object inventory... done
build succeeded.

Build finished. The HTML pages are in _build/html.
```

_build/html/ディレクトリの下に reST から生成された HTML ファイルができます。

6.4 Debian の日本語環境での状況

HTML の場合は日本語も問題なく表示できました。他のフォーマットはどうでしょうか。結果は下記のとおりです。(表 3)

表 3　フォーマット毎のビルド結果

フォーマット	結果
html	OK
epub	OK (ただし、 CSS は反映されない)
text	OK
man	OK
latex	OK
latexpdf	NG

上記のとおり、 LaTeX から PDF への生成がうまくできません。

```
(snip)
! PACKAGE INPUTENC ERROR: UNICODE CHAR \U8: NOT SET UP FOR USE WITH LATEX.

SEE THE INPUTENC PACKAGE DOCUMENTATION FOR EXPLANATION.
Type  H <return>  for immediate help.
 ...

l.119 \chapter{東京エリア Debian 勉強会}

?
(snip)
```

これは生成される LaTeX 文書が UTF-8 であるためです。 Debian JP Project での課題にもなっていますが、現状の Debian の TeX 系では日本語の UTF-8 は未対応です。

また、日本語を使っていなくても、 GIF イメージを".. image::" で読み込んでいる場合に PDF の生成に失敗するようです。

6.4.1 rst2pdf を使う方法

reST から PDF への生成には、 LaTeX 経由での方法以外に、 rst2pdf というツールを使う方法もあります。まず、rst2pdf パッケージをインストールします。

```
$ sudo apt-get install rst2pdf
```

インストール後、先ほど作った Sphinx のプロジェクトディレクトリの直下に conf.py という設定ファイルがあるので、この中の extensions に下記を追記します。

```
extensions = ['sphinx.ext.autodoc','rst2pdf.pdfbuilder']
```

PDF のオプションを追記します。

```
pdf_documents = [
        ('index',u'TokyoDebianMeeting', u'Tokyo Debian Meeting', u'Debian JP Project'),
]
pdf_stylesheets = ['sphinx','kerning','a4','ja']
pdf_font_path = ['/usr/share/fonts/']
pdf_language = 'ja_JP'
```

Makefile に下記を追記します。

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

ja.json ファイルを作ります。

```
{
  "fontsAlias" : {
    "stdFont": "ttf-japanese-gothic",
    "stdBold": "ttf-japanese-gothic",
    "stdItalic": "ttf-japanese-mincho",
    "stdBoldItalic": "ttf-japanese-mincho",
    "stdMono": "ttf-japanese-gothic"
  }
}
```

make pdf を実行すると、_build/pdf/TokyoDebianMeeting.pdf が生成されます。日本語の表示も問題ありません。詳細については、/usr/share/doc/rst2pdf/manual.pdf.gz にマニュアルがあるので、これの「Section 18 Sphinx」のページを参照してください。なお、この場合は make latexpdf ではうまくいかなかった Gif ファイルの読み込みは問題ありません。

しかし、この方法では sphinxcontrib.*diag を使うと、ビルドに失敗するという別の問題があります。

6.5 Doxygen とは

さて、今回のもう一つのドキュメント生成ツールである Doxygen について見てみます。 Doxygen はソースコードを解析してドキュメントを生成するツールです。対応する言語は C/C++、 Java、 Python、 C#、 Objective-C などをサポートし、 D や PHP も部分的にサポートしています。

一方、生成可能なフォーマットは、 HTML、 LaTeX、 RTF(MS-Word)、 PostScript、 PDF、 man などがあります。

6.5.1 Debian で使ってみる

今回は、 Debian 勉強会参加登録システムのソースコードからドキュメントを生成してみることにします。

Debian パッケージがあるので、 doxygen パッケージをインストールします。

```
$ sudo apt-get install doxygen
```

次に、ソースツリーのルートディレクトリに移動し、設定ファイルを生成します。

```
$ cd monthly-report/utils/gae/
$ doxygen -g .doxgen.conf
```

Debian 勉強会参加登録システムは Python なので、最低限次の設定項目の設定を行います。 (表 4)

表 4　Doxygen の設定項目

設定項目	デフォルト値	設定例
PROJECT_NAME		Tokyo Debian Meeting
PROJECT_NUMBER		1.0
OUTPUT_LANGUAGE	English	Japanese
TAB_SIZE	8	4
INPUT		.
FILTER_PATTERNS		*.py

doxygen コマンドを実行します。

```
$ doxygen .doxygen.conf
```

すると、 monthly-report/utils/gae/ディレクトリ以下に、 html, latex ディレクトリができます。 html ディレクトリ以下には HTML 形式で、 latex ディレクトリ以下には、 LaTeX 及び PDF 形式でドキュメントが生成されます。

w3m で html/index.html を見ると、以下のような画面が表示されます。

“クラス” リンクをクリックするとクラスの一覧が展開されます。

例えば、”admin_event::EditEvent” のリンクをクリックすると、 admin_event::EditEvent クラスについてのドキュメントを見ることができます。

Doxygen についての詳細は doxygen.jp[*6]のマニュアルを参照してください。

[*6] http://www.doxygen.jp/manual.html

6.5.2 Sphinx との連携

breathe[*7]というツールを使うと、 reST/Sphinx から Doxygen に連携できるようです[*8]。なお、 Debian パッケージにはなってません。

6.6 まとめ

ソースコードからドキュメントを作る Doxygen, またドキュメントの作成自体を簡単にする Sphinx を使うと、大変でなかなかやりたがらないドキュメントの作成の敷居を低くすることができます。また冒頭で紹介した Sphinx 拡張としても使える*diag シリーズや、まだ Doxygen と Sphinx を連携する Breathe を使うことによって、これらのドキュメント生成ツールの利用価値が上がります。

自分のドキュメント作成のモチベーションを上げる意味でも、*diag シリーズだけでなく、 Breathe についても、Debian パッケージ化を行おうと思います。

参考文献

[1] Georg Brandl, Shibukawa Yoshiki(Japanese), “Overview - Sphinx v1.0.6 documentation” 2007-2010. http://sphinx-users.jp/doc10/

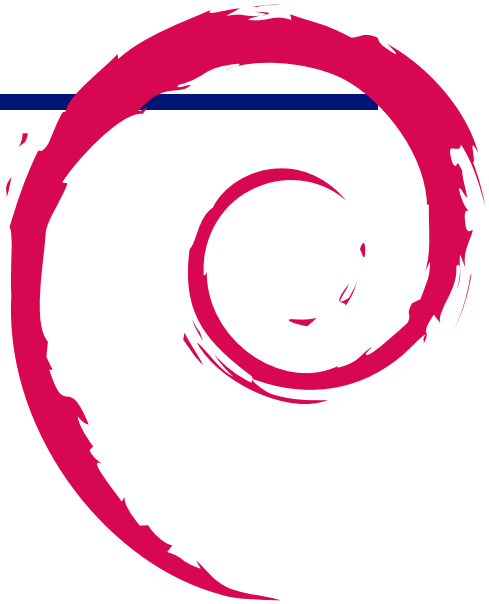
[2] MiCHiLU “Sphinx で日本語 PDF を生成する” 2009. http://d.hatena.ne.jp/MiCHiLU/20091009/

[3] Dimitri van Heesch 1997-2010, OKA Toshiyuki (Japanese translation) 2001, TSUJI Takahiro (Japanese translation) 2006-2011, TAKAGI Nobuhisa (Japanese translation) 2006-2011, “Doxygen マニュアル” http://www.doxygen.jp/manual.html

[4] OKA Toshiyuki, “Doxygen を使おう” 2002. http://www.fides.dti.ne.jp/~oka-t/doxygen.html

[*7] https://github.com/michaeljones/breathe

[*8] http://sphinx.shibu.jp/faq.html



7 索引

Debian 勉強会資料

2011 年 6 月 18 日　　初版第 1 刷発行

東京エリア Debian 勉強会（ 編集・ 印刷・ 発行）