Writing Books Using Open Source Software

Wesley J. Chun
wescpy@gmail.com
OSCON, 2009 Jul 23
San Jose, CA, USA

About The Talk

- We can write software using open source tools...
- Can the same be said of writing books?
- Hypothesis
  - What do you think?!? :-)
- Proof?
  - Give examples of open source tools used for software
  - Show some of those same tools used for book-writing
  - Discuss some authors and their tools
About The Speaker

- Software engineer by profession, also book author
- Skipped generations of writing tools
  - *troff/nroff* user but not LaTeX
  - FrameMaker user but not Word
  - Markdown, *wiki* user but not DocBook nor Textile
  - reST is next...
- vi (now vim) user since 1985
  - Not totally naive; know some *emacs* too ;)
  - Prefer plain text editing (wonder why?)
    - XML doesn't count... :P

About the Development Tools

- Text editor(s)
- Syntax, formatting, and layout
- Versioning and backup
- Running tests
- Team communication
- Issue-tracking
- Generate for production
**Text Editors: Source Code**

```python
import tkinter

top = tkinter.Tk()

hello = tkinter.Label(top, text='Hello World!')
hello.pack()

quit = tkinter.Button(top, text='QUIT', 
                      command=top.quit, bg='red', fg='white')
quit.pack(fill=tkinter.X, expand=1)

top.mainloop()
```

**Text Editors: Manuscript**

If you were developing on a Win32 platform, opening an Explorer window to that folder will look something like Figure 2.1, if we created a folder named `\django` with the intention of putting our project there:

![Figure 2.1: 'mysite' folder on Win32](../screenshots/2-1-mysite.png)

**Figure 2.1: 'mysite' folder on Win32**

*NOTE*: As you probably know if you are an advanced Python user, that `__init__.py` file makes this project directory a Python `package` -- a collection of related Python modules. Its status as a package allows us to use Python's dot-notation to address individual pieces of our project, like `mysite.urls`. (You can read more about them in the Python introductory chapter.)

Besides `__init__.py`, the 'startproject' command has created three other files:

* `manage.py` is a utility for working with this Django project. You can see from its permissions flags in the directory listing that it is executable. We'll be running it in a moment.
* `settings.py` is a file containing default settings for your project. These in...
Syntax, Formatting, Layout (SFL)

- Writing software
  - Programming language syntax
    - Python, Perl, Ruby, VB
    - Java, C, C++; PHP

- Writing books
  - Markup Syntax
    - reST, Markdown, *wiki, Textile
    - DocBook-XML, SGML, LaTeX, *roff

SFL: Source Code

```python
import Tkinter
top = Tkinter.Tk()
labels = Tkinter.Label(top, text='Hello World!')
labels.pack()
button = Tkinter.Button(top, text='QUIT', command=top.quit, bg='red', fg='white')
button.pack(fill=Tkinter.X, expand=1)
top.mainloop()
```
SFL: Manuscript "Source"

If you were developing on a Win32 platform, opening an Explorer window to that folder will look something like Figure 2.1, if we created a folder named "C:\py\django" with the intention of putting our project there:

![screenshot](../screenshots/2-1-mysite.png)

**Figure 2.1: 'mysite' folder on Win32**

**[NOTE]:** As you probably know if you are an advanced Python user, that "__init__.py" file makes this project directory a Python *package* -- a collection of related Python modules. Its status as a package allows us to use Python's dot-notation to address individual pieces of our project, like "mysite.urls". (You can read more about them in the Python introductory chapter.)

Besides `__init__.py`, the `startproject` command has created three other files:

- `manage.py` is a utility for working with this Django project. You can see from its permissions flags in the directory listing that it is executable. We'll be running it in a moment.
- `settings.py` is a file containing default settings for your project. These in

Versioning and Backup (V&B)

- Repository system
  - SCCS, RCS, CVS, SVN (Subversion)
  - git, hg (Mercurial), bzr (Bazaar)
- Backup and recovery
  - cp, rcp, scp/rsync, tar, cpio, dump
  - Amanda, Bacula, Mondo; DirSync, Unison; FlyBack, TimeVault
- Code vs. Manuscript
  - Both sources are in like formats (plain text)
    - Identical: managing files from repository
    - Backup & recover: works the same for both
Running Tests

- Code: Unit and regression tests
- Question: can you "test" a document?
- Sure! (to a certain extent)
  - For software, you're testing the code itself
  - For a book, you can test...
    - The code in your book if it is a technical book
    - "Validity" if markup system is "tagged" (XML, etc.)
- Python "docstrings" really help

Python Docstrings & doctest Module

"doctestDemo.py - demo doctest module"

def foo(x):
    """foo(x): display argument 'x'

    >>> foo(123)
    123
    """
    print x
"Testing" a Manuscript

Team Communication

- Similar for software or book development
- Email
  - various
- Mailing List and Archive
  - Listserv, Majordomo, Mailman
  - Listproc, Lyris, SmartList
Email, Mailing List, Archive

- Ph. W. J. Chun
  - Re: [Django-Book] explicit imports in urls.py
    - Mon, 5/25/08 19 KB
- Ph. Paul Bissex
  - Re: [Django-Book] trying out Chapter 2
    - Mon, 5/26/08 6 KB
- W. J. Chun
  - [Django-Book] CMS (was: Re: explicit imports in urls.py)
    - Mon, 5/26/08 5 KB
- Ph. W. J. Chun
  - Re: [Django-Book] trying out Chapter 2
    - Mon, 5/26/08 8 KB
- Practical Django Development
  - [Django-Book] [Practical Django Development] best
    - Mon, 5/26/08 4 KB
- W. J. Chun
  - Re: [Django-Book] trying out Chapter 2
    - Mon, 5/26/08 9 KB
- W. J. Chun
  - Re: [Django-Book] [Practical Django Development] #17
    - Mon, 5/26/08 6 KB
- Practical Django Development
  - Re: [Django-Book] [Practical Django Development] #20
    - Mon, 5/26/08 4 KB
- Practical Django Development
  - [Django-Book] [Practical Django Development] #59
    - Mon, 5/27/08 4 KB
- Jeff Forliz
  - Re: [Django-Book] [Practical Django Development] #59
    - Tue, 5/27/08 5 KB
- Jeffrey Forliz
  - [Django-Book] read: More chapters?
    - Thu, 5/29/08 8 KB
- Paul Bissex
  - [Django-Book] status update
    - Thu, 5/29/08 4 KB

Issue-Tracking

- trac, RT, Bugzilla, Mantis, Redmine
- Code issues
  - New features
  - Bugs
  - Feature enhancements (or removals)
  - Milestones
- Manuscript "issues"
  - New material
  - "Bugs:" typos, errata
  - Future or outdated material
  - Milestones
Generate for Production

- For end-user consumption; software & books...
  - Undergo some production/release process
  - More similar than you would think
- Code
  - (Possible compilation process)
  - Install on production server
  - Shrinkwrapped software package
  - (Possible printing, boxing, distributing, media)
- Manuscript
  - HTML, PDF®, XML or other formatting "compilation"
  - Proofreading/editing(*) and layout
  - Printing, binding, distributing, perhaps media

Just run make (!)

```
myMac$ make allzip
rm -rf *0706.zip
markdown introduction.txt -x utables >> introduction.html
markdown 1_practical_python_for_django.txt -x utables >> 1_practical_python_for_django.html
markdown 2_django_for_the_impatient.txt -x utables >> 2_django_for_the_impatient.html

cd .. & zip chapters/html0706.zip chapters/introduction.html chapters/1_practical_python_for_django.html chapters/2_django_for_the_impatient.html [...]
  adding: chapters/introduction.html (deflated 54%)
  adding: chapters/1_practical_python_for_django.html (deflated 67%)

html2ps introduction.html | ps2pdf - Introduction.pdf
html2ps 1_practical_python_for_django.html | ps2pdf - 1_practical_python_for_django.pdf


cd .. & zip chapters/pdf0706.zip chapters/introduction.pdf chapters/1_practical_python_for_django.pdf chapters/2_django_for_the_impatient.pdf [...]
  adding: chapters/introduction.pdf (deflated 30%)
  adding: chapters/1_practical_python_for_django.pdf (deflated 20%)

  adding: chapters/matter/clopphon.pdf (deflated 42%)
myMac$
```
Case Studies

- Forcier, Bissex, Chun
- Ramm, Dangoor, Sayfan
- Mertz
- Martelli
- Summerfield
- others

J. Forcier, P. Bissex, W. Chun
**Python Web Development w/Django**

- Editing: vim, emacs, TextMate
- Format: Markdown
- Repository: svn
- Issue-tracking: trac
- Mailing list: Mailman
- Conversion: HTML (Markdown) => PS (html2ps) => PDF (ps2pdf)
- Build: make
- Miscellaneous: "try_excerpt" tool cuts-n-pastes code snippets; "test_snippets" tool tests code execution in manuscript
- FUTURE: reST (format), Sphinx (conversion), Redmine (issue-tracking), hg (repository)

---

**I Irony: publisher imported into Word**
Our makefile

```
M. Ramm, K. Dangoor, G. Sayfan
```

```
Rapid Web Applications with TurboGears

MARK RAMM
KEVIN DANGOOR
GIGI SAYFAN
```
Rapid Web Apps with TurboGears

- Format: Markdown
- Conversion: HTML (Markdown) => PDF (Adobe InDesign) -- all subsequent edits in InDesign
- Repository: svn
- Issue-tracking: trac
- FUTURE: reST (format), Sphinx (conversion), hg (repository)

David Mertz
Text Processing in Python

- Preferences
  - Using a plain text editor
  - Extremely minimal markup language
- Editing: Boxer (OS/2), jEdit (plain text editors)
- Markup: "Smart ASCII"
- Conversion: LaTeX (Python+shell scripts) => PDF (LaTeX tools)
- FUTURE: same as above but add repository

Alex Martelli

![Python in a Nutshell](image1)

![Python Cookbook](image2)
Alex Martelli

- *Python in a Nutshell*
  - Editing: Word
    - "EEK, what a nightmare: never EVER again"
- *Python Cookbook*
  - Markup: Docbook (XML)
  - Editing: Oxygen (XML editor), vim and gvim
  - Conversion: lots of Python scripts
  - Repository: svn
- FUTURE: XMLmind (editing), hg (repository)

Mark Summerfield

*Programming in Python 3*
A Complete Introduction to the Python Language

*Rapid GUI Programming with Python and Qt*
The Definitive Guide to PyQt Programming
Mark Summerfield

- *Rapid GUI Programming with Python and Qt*
- *Programming in Python 3*
  - Markup: lout
    - Similar to LaTeX but only does PS... no HTML, etc.
  - Editing: vim and gvim
  - Conversion: PS (lout) => PDF (ps2pdf)
  - Miscellaneous: "snip" tool that cuts-n-pastes code snippets

Leon Atkinson

*core PHP programming*  
*core MySQL*  
*TTHIRD EDITION*  
*The authoritative guide to real-world development with the new PHP 5*  
*Brand-new coverage of PHP design patterns, client techniques, and XML*  
*Updated coverage of servers, functions, algorithms, and PHP software engineering*  
*Includes 650+ downloadable code examples*  
*Contributed by Zeev Suraski, PHP 5 contributor and Zend Engine 2 co-creator*  

Reviewed and enhanced by Michael Watin, the developer of MySQL
Conclusion?

- Introduced our hypothesis
- Demonstrated tools, usage for writing software & books
- Gave numerous examples of what authors use today
- Major outstanding issue
  - Manuscript files edited by publisher
Disconnect

- Edits/corrections not propagated back to source files
  - Generated HTML imported elsewhere (e.g., into Word)
    - All further editing with new master
  - Authors may do it themselves only...
    - If they insist and there is time
    - If they have time
- Not applicable for developing software

The Future and You

- Using and/or supporting open source?
- Considering writing a book?
- Afraid you can't stick with the tools you're familiar with?
- Think you can't publish unless you submit files in a proprietary word processor format?
Think Again

- Industry Changes
  - Hardware: proprietary *ix OSs to Linux, *BSD, etc.
  - Software: proprietary tools to open source
- Book writing seems to follow the same trend
- May have to force publishers to be more flexible
  - Else produce camera-ready PDFs on your own
- For coders, writing a book in plain text makes you a happier person
  - Spend less time as a fish out of water

FINIS