Python No Muerde

Capítulo: El Meta-Libro

"Escribir es un asunto privado."

Goldbarth

Una de las intenciones de este experimento escribir-un-libro fue hacerlo "en publico". ¿Porqué?

- Me gusta mucho el open source. Trato de aplicarlo en muchas cosas, aún en aquellas en las que no se hace habitualmente. Por ejemplo, si bien no acepto colaboraciones para el libro, si acepto parches.
- En mi experiencia, si hay gente que le interesa un proyecto mío, entonces es más probable que no lo deje pudrirse por abandono. Creí (aparentemente con razón) que a la gente de PyAr le interesaría este proyecto. Ergo, le vengo poniendo pilas.
- Los últimos quince años metido en proyectos open source y diez años de blog me han convertido en una especie de exhibicionista intelectual. Idea que me pasa por el bocho la tiro para afuera. O la hago código, o la hago blog, o algo. Este libro es algo así, tuve la idea, no la puedo contener en mi cabeza, la tengo que mostrar.

Y uno de los efectos de querer mostrar el libro mientras lo hacía es que *tengo que poder mostrarlo* y no tiene que ser algo demasiado vergonzoso estéticamente y tiene que poder leerse cómodamente.

Como ya es casi natural para mí escribir reStructured text (hasta los mails me suelen salir como reSt válido), busqué algo por ese lado.

Para generar PDFs, elegí rst2pdf porque es mío y si no hace exactamente lo que yo quiero... lo cambio para que lo haga [1]

1 De hecho, usarlo para este proyecto me ha permitido arreglar por lo menos cinco bugs :-)

Para el sitio, la solución obvia era Sphinx, pero... me molestan algunas cosas (menores) de incompatibilidad con docutils (especialmente la directiva `class`), que hacen que un documento Sphinx sólo se pueda procesar con Sphinx.

Entonces, buscando alternativas encontré rest2web de Michael Foord que es **muy** fácil de usar y flexible.

Al ser este un libro de programación, tiene algunos requerimientos particulares.

Código

Es necesario mostrar código fuente. Rst2pdf lo soporta nativamente con la directiva `code-block` pero no es parte del restructured text standard. En consecuencia, tuve que emparchar rest2web para que la use [2]

2 Por suerte la directiva es completamente genérica, funciona para HTML igual que para PDF. Esto es lo que tuve que agregar al principio de `r2w.py`:

```
from rst2pdf import pygments_code_block_directive
from docutils.parsers.rst import directives
directives.register_directive('code-block', \
    pygments_code_block_directive.code_block_directive)
```

Gráficos

Hay algunos diagramas. Los genero con la excelente herramienta Graphviz.
Los quiero generar en dos formatos, PNG para web PDF para el PDF, por suerte graphviz soporta ambos.

Build

Quiero que cuando cambia un listado se regeneren el sitio y los PDF. Quiero que cuando cambia el estilo del PDF se regenere este pero no el sitio. Quiero que todo eso se haga solo.
Sí, podría haber pensado en algo basado en Python pero, realmente para estas cosas, la respuesta es `make`. Será medio críptico de a ratos, pero hace lo que hace.
Por ejemplo, así se reconstruye el PDF de un diagrama:

```
%.graph.pdf: %.dot
	dot -Tpdf $< > $@ -Efontname="DejaVu Sans" \
	    -Nfontname="DejaVu Sans"
```

Y se ejecuta así:

```
$ make loop-n-y-medio.graph.pdf
dot -Tpdf loop-n-y-medio.dot > loop-n-y-medio.graph.pdf
-Efontname="DejaVu Sans" -Nfontname="DejaVu Sans"
```

Normalmente no hace falta hacerlo manualmente, pues se hace, de ser necesario, cuando se publica al sitio o a PDF.

Feedback

Como toda la idea es tener respuesta, hay que tener como dejarla. Comentarios en el sitio via disqus.

Tipografía

Es complicado encontrar un set de fuentes modernas, buenas, y coherentes. Necesito por lo menos bold, italic, bold italic para el texto y lo mismo en una variante monoespaciada.

Las únicas familias que encontré tan completas son las tipografías DejaVu y Vera. Inclusive hay una DejaVu Thin más decorativa que me gustó para los títulos.

HTML

Soy un queso para el HTML, así que tomé prestado un CSS llamado LSR de http://rst2a.com. Para que la estética quede similar a la del libro usé TypeKit (lamentablemente me limita a 2 tipografías, así que no pude usar Dejavu Thin en los títulos/citas).

Server

No espero que tenga mucho tráfico. Y aún si lo tuviera no sería problema: *es un sitio en HTML estático* por lo que probablemente un pentium 3 pueda saturar 1Mbps. Lo puse directamente en el mismo VPS que tiene mi blog.

Versionado

No hay mucho para discutir, cualquiera de los sitios de hosting libres para control de versiones serviría. Usé mercurial (porque quería aprenderlo mejor) sobre googlecode (porque es mi favorito).

Por supuesto que toda la infraestructura usada está en el mismo repositorio de mercurial que el resto del libro.

Licencia

La elección de licencia para un trabajo es un tema personal de cada uno. Creo que la que elegí es *suficientemente libre*, en el sentido de que prohíbe las cosas que no quiero que se hagan (editar el libro y venderlo) y permite las que me interesa permitir (copiarlo, cambiarlo).

Por supuesto, al ser yo el autor, siempre es posible obtener permisos especiales para cualquier cosa *pidiéndolo*. Tenés el 99% de probabilidad de que diga que sí.