This site.

BV

[2014-03-02 Sun 15:02]

This web site has evolved over time. It is currently written with Emacs org-mode. It's Org source is exported with OrgOnPy and Pelican. This topic contains notes on how this is done.

1 Generalities

I use Emacs for producing pretty much all of my information "output". Except for Email (and IRC), I use Org mode to manage pretty much all my text-based (non-code) output. The main content that this web site exposes are "topics" which are formal(ish) notes (like this one) which I produce to remind myself what I did and for the sake of anyone else that may be interested. I will sometimes revise topics after their initial "publication".

1.1 History

This site went through many iterations. Below we go through them in reverse chronological order.

2 Emacs and Org

My Emacs config is online. The Org initialization adds to the `org-capture-templates` list so I can start a "capture" of a topic with `C-c c b` or revisit an existing topic with `C-c c B`.

Because a topic may include multiple files, I have this start a sub-directory for each topic under my topics storage area (`~/org-pub/topics`).

Each new topic is created using a template which will cause Org to prompt for various items used later including title, subtitle, category, tags (note, not same as Org headline tags) as well as automatically fill some things

like the date stamp (interpreted as a creation date). It also sets a `SETUPFILE` which contains some minimal, global Org setup.

During authoring I can get a preview of how things look by doing a local export (eg, `C-c C-e l p` for PDF, `C-c C-e h h` for HTML). This export is not what eventually turns into the web page.

3 OrgOnPy

Unfortunately, I have never been able to fully grok Elisp and Python is my preferred scripting language. So that I may process content in Org in non-trivial ways I have developed OrgOnPy. It works by running some Elisp which I did manage to write with a lot of help from the Org mailing list to get the Org document as an `org-element` tree. It then breaks some of the circular references that exist in that tree before using Emacs's `json` module to convert it to JSON. OrgOnPy provides an `org-element`-like tree representation which can then be used to provide Python objects which are somewhat reminiscent of their Elisp counterparts.

4 Pelican

Pelican is a static site generator. It is well designed with fairly good layering and a plugin system. HTML is generated from two paths: the overall structure of the pages is determined by Jinja templates driven by metadata while the "payload" content is directly converted from the content source files.

It is the job of the "reader" layer to provide metadata and HTML content. The "reader" is an explicit layer in the Pelican design and it comes with readers that support Markdown, rST and AsciiDoc. One may also provide the "reader" as a plugin which is what OrgOnPy does to add support for source files in Org markup. The OrgOnPy Pelican reader is heavily inspired by `org_reader` which is part of the Pelican plugin collection. OrgOnPy also provides a modified version of the `extract_toc` plugin also from the Pelican plugin collection.

Pelican has a number of "themes" which provide for HTML structure and CSS styling. For the most part a theme is defined at a layer boundary so that most themes can be interchanged without changes to the source content files. I made a a survey of Pelican themes ultimately settling on elegant as being closest in structure and functionality to what I wanted. Inevitably I wanted to tweak so I have lightly forked elegant.

4.1 Machinations

Create an environment for building the web site

```
$ virtualenv venv
$ source venv/bin/activate
$ pip install pelican beautifulsoup4
```

Build the site

```
$ cd ~/org-pub/pelican/site
$ pelican
$ (cd output && python -m pelican.server)
```

Content is held in `site/content/` and exists as either immediate files or relative symlinks to sub-directories of `topics/`.

That build locally. Deployment is done with the help of the Fabric file and goes like:

```
$ pip install fabric
$
```

5 Links

There are other ways to use Org to make web pages. Here are some I've looked at.

- `http://www.seas.upenn.edu/~heqin/academic/sitecreation.html`
- `http://doc.norang.ca/org-mode.html`
- `http://www.nicolas-petton.fr/blog/blogging-with-org-mode.html`
- `http://stevenbagley.net/blog/blog-with-emacs-org-mode.html`
- `http://kerunix.com/blog-using-orgmode-and-pelican.html`
- `http://steckerhalter.co.vu/posts/blogging-with-org-mode.html`
- `http://justinlilly.com/emacs/orgmode_static_site_generator.html`