Pelican Setup

Pelican can be installed as follows

Within your python virtual environment:

```
source ${HOME}/venv_generic/bin/activate
pip install pelican
```

Or install it using the ubuntu package.

```
sudo apt-get install pelican pelican-doc
```

For our own site projects we use our own script "compile_rst" which generates lists of links to include in our site's pages. It can also be used to build pages which reference our compilation of music tracks.

That script relies on installing these packages.

```
sudo apt-get install python3-mutagen
sudo apt-get install python-mutagen-doc
```

In each of our pelicanconf.py configurations we select a theme for the given site. We assume the theme exists under ${HOME}/pelican-themes.

To get a collection of themes to choose from, we cloned as follows:

```
git clone --recursive https://github.com/getpelican/pelican-themes ${HOME}/pelican-themes
```

Start up

I used 'pelican-quickstart' to generate initial static website content using pelican.

pelican-quickstart creates the wherewithall to generate a website in the current empty directory. Afterwards you create .rst files in content/articles as blog content and/or you create .rst files in content/pages as web pages.

I have created a script to generate pages for audio albums, and to create an include file listing links to tarballs.

The include file can be included in a .rst file under content/pages to list the links.

Our www.bernatchez.net test site is an example using all those features.

compilation

To generate the static site files, once we have all our source files the way we want them we only need to use the Makefile. The following will compile emit some generated .rst files among the manually edited ones under content/.. and then from that emit all the web site assets to output/ using pelican.

```
make html
```

server test

Once we have our generated output/ we can check out what we have serving it locally also using pelican

```
pelican -p 8000 -l --relative-urls
```

To stop serving type <ctl>c

View the site while it is being served using any browser pointed to this url:

```
http://127.0.0.1:8000/index.html
```

reminder

Our proof of concept mp3 files needed to be massaged so that the script compile_rst could extract the information it needs from them to build the album pages correctly. Some of these tools were used to do so. None of this business of audio compilation and prepping mp3 files is robust enough to use in any kind of real production context.

Once our compilations directory contains sane enough mp3 files, the audio features of compile_rst can be used to generate pages.

audio file processing tools I used:

- exfalso

  GUI for inspecting and editing tags. We use this to massage our raw audio content into something sane enough to be admitted to our web page compilations directory.

  I use it to fixup audio files. When I get a cd full of mp3 files I give them all the same album name, and the same date (a year like 2015). I try to arrange them in the same order as appears on the cd label and give them sequential track numbers starting from 1. I label them all 'artist' 'unknown', and give them all the same genre (i.e. latin). Then I go through them and try to replace the title and artist with something useful. Then I rename all the files to names like YYYY-albumname-track.mp3
- id3info

  Display id3 tag information.
- id3cp

  Copies tags from one file to another.
- id3convert

  Converts between id3v1 and id3v2 tags of an mp3 file.
- id3tag

  Tags an mp3 file with id3v1 and/or id3v2 tags.
- mid3iconv

  Convert ID3 tag encodings: better replacement for above supposedly
- mid3v2

  Audio tag editor similar to 'id3v2'
- mp3rename

  Rename mp3 files based on id3tags
- mp3tag

  View and manipulate ID3v1 tags

To get a list of genres:

```
mp3tag -g  list
```