Using org2blog

Since moving from OSX to Linux, for reasons I will elaborate on in another post, the only real thing which I missed was the MarsEdit blogging application. The immediate effect is that both this blog and my cobra blog have not seen any posts since the switch.

Initially I did make an effort to create a VirtualBox image for MacOSX so I could potentially run OSX applications in my Ubuntu install, but Apple made a pretty good effort to prevent me from running a virtualized OSX, even if you have ticked all their boxes (read: spent enough money on both their hardware and software). So, for the moment, no OSX applications on my main machine anymore.

The next step was to decide on a new blogging solution, or perhaps publishing solution, as a replacement for MarsEdit. My requirements were as follows:

- emacs based solution, preferably integrated with org-mode;
- based on an api that wordpress supports;
- reasonably active project and responsive.

Finding a solution that satisfies all 3 of the above requirements basically left me with org2blog. I did look at a couple of other solutions but none of them met all 3 requirements, amazingly.

The use of org2blog is pretty simple, especially if you are already using orgmode to publish documents. The gist of it is to start an outline header and start writing the blog-post. Alternatively the command `M-x org2blog-new-entry` can be used to start a blog-entry in a new buffer (presumably to be saved to a file later on). Once finished writing a `C-c d` keyboard shortcut publishes a buffer as a draft, where `C-c p` publishes the buffer as a finalized post. (Posting as a page in wordpress is also possible).

I'm using the *post-per-buffer variety* for this post, so I can attach the source for this posting at the end more easily. I also found that working in a *one-posting-per-file* matter is easier. (For starters, the org2blog key shortcuts work properly).

The concept of org2blog is rather simple. It piggybacks on the excellent export options already present in org-mode and exports the relevant piece of text to html, takes that output and uses the wordpress xml-rpc api to publish it.

This post contains a small set of examples which should be enough to fullfill the vast majority of my blogging-needs. I need a way to chop up text into paragraphs, marking some of my words with some inline markup and I need a way to insert images in a variety of ways. Additionally I want to be able to attach/link to files, preferably automatically attached to the blog-posting.

So, let's have a look at the basic ingredients.

Headers

Outline headers in org-mode (the lines that start with one or more stars) are translated into html header elements. Here is an example of the first 5 levels of org-mode outline header levels:

Header 1

Header 2

Header 3

1. Header 4
   (a) Header 5

Headers (cont'd)

The limitation/feature of org-mode that a paragraph belongs to the immediate header above it is equally valid for the texts produced with the help of org2blog. In this example, had I not put the *Header (cont'd)* header above this paragraph, the text would have belonged to the *Header 5* header.

With these, basic structuring of text is achieved. Within normal text, inline markup like **bold**, *emphasized*, underlined, ~~striked through~~, and `monospace` and `ver  ba  tim` should work as well. (not sure what verbatim should produce, but typically a multiple of spaces is the way to distinguish it from normal monospaced paragraphs)

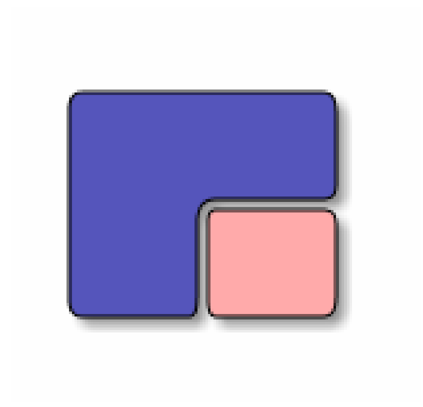
These two mechanisms should be enough to create readable blobs of texts, but obviously I would like to have some means to make my posts a bit more attractive; by including images.

Images

Images in org-mode are basically references / links to image locations. If such an image is a file on the local disk of the machine I'm working on, I can insert a link to that file in the usual org-mode way.

```
[[file:filename.png]]
```

Which results in this:



Users of org-mode may recognize this image as an example of the usage of the ditaa system. Processing of *begin$_{src}$* sections in the org file will be done prior to publishing, so the image does not even need to be linked explicitly, but can be an implicit result of code evaluation like in the example below:

```
#+begin_src ditaa :file filename.png :cmdline -r
+----------+
| cBLU     |
|          |
|    +-----+
|    |cPNK|
|    |    |
+----+----+
#+end_src
```

```
+----------+
| cBLU     |
|          |
|    +-----+
|    |cPNK|
|    |    |
+----+----+
```

In this example the ditaa code evaluated delivered a file blue2.png which was subsequently uploaded to the blog. (the problem here is that mentioning the filename translates it into the url on the server, the basename of the file is the local filename).

Using images which are already somewhere else on the web (I tend to use flickr.com for these[1]) inserting a link to their location should suffice to get the

[1]Not anymore, I've migrated most, if not all, image to a self-hosted instance of mediagoblin.

image in the web page.

Example:

So, with this basic knowledge I should be able to start publishing again.

Footnotes