Blogging Beyond Jekyll

Martin Fenner, Gobbledygook

March 23, 2015

This blog has been on four different platforms since starting in 2007: a custom blogging engine and then Movable Type on Nature Network 2007-2010, Wordpress on the PLOS Blogs Network 2010-2013, and the static blogging engine Jekyll hosted on Github Pages since 2013. It might be time for yet another blogging platform change.

The main reason to switch from Wordpress to Jekyll was the concept of a static site generator: write posts in markdown format, store them in a Github repository, and then have Jekyll automatically generate the HTML pages hosted on Github Pages. The main attraction was the blog posts in markdown format stored in git version control without the need of a database. Jekyll is the glue to make all this work, and I was able to customize Jekyll to my needs, e.g. by using Pandoc for the markdown to html conversion.

While this workflow still makes sense for this blog, there are a number of shortcomings:

- Jekyll needs to rebuild the entire site every time I publish a new post. While this isn't much of a problem for the size of this blog, it doesn't scale well for larger sites. And the process is more complex if you use custom jekyll plugins like this blog, as you can't use the automatic Jekyll pipeline provided by Github (hint: use a Travis continous integration server to build the site)
- the web is moving to increasingly sophisticated javascript frontends, using frameworks such as Angular.js, Ember.js, or frontend libraries for scholarly documents such as Lens. While they can be used together with Jekyll, that is not a typical use case.
- the tight integration between the code to generate the website and the content (Wordpress and other blogging engines have the same approach) is not always the best solution, e.g. when you want to want to generate the pages for something that is not a blog (e.g. a book).

What could we do instead?

> Build a Javascript frontend where the content is served via an API built around markdown documents, stored in git version control.

API

The blog posts are still written in markdown, stored (and version-controlled in a Github repository), but we would now access the content via API. The easiest solution is to use the Github Contents API and either do the markdown to html conversion in javascript yourself, or let the Github API do the conversion to HTML for you. Alternatively we could build our own API, e.g. because we want to control the markdown to html conversion, or need additional functionality such as fulltext search. And of course the two approaches can be combined, e.g. via a Github webhook that triggers the markdown to html coversion every time a document is added or updated, and stores the converted documents in the same repo.

Frontend

The frontend should be written as a one-page javascript application, not requiring a server backend. In contrast to the Jekyll workflow the frontend code doesn't need to be updated every time we post a blog post. Since this is a very common scenario, there are probably several solutions out there already. Please mention them in the comments if you have suggestions. One candidate is Lens mentioned above - a beautiful frontend for scholarly documents. Lens displays documents in the JATS XML format, so your API would have to provide that format.

Conclusions

The separation into API and frontend is of course old news. But for blogs this seems to still be a fairly new concept, in particular when combined with a backend using documents stored in git version control rather than in a database. Wordpress added a REST API Plugin in 2014, and the Ghost blogging framework (which uses a database backend) also seems to go into that general direction. Please ping me if you like the idea and want to contribute, or have implemented something like this already.