Migrating to GitHub Pages and Jekyll

Thursday, 1 May 2014

Background

I've always been a fan of using Markdown to create web content. Several years ago, I created MDEngine, a small PHP script to render Markdown files in HTML dynamically. For a while, it was responsible for much of the content on my website. In October 2013, I began work on a fresh design. I decided to use a custom Node.js app deployed on Heroku for processing the Markdown. While this worked effectively, I always had some reservations.

While my site was decently fast, there was no real reason that it needed to be dynamically generated. I was particularly concerned with the performance of the two list pages, whose backend logic consisted of parsing an entire directory of Markdown files each time it was loaded. Though there was no noticeable performance impact, it was not inconceivable that the page generation time would increase substantially as content grew.

In late April 2014, I made some design updates to the site running on Heroku. I decided to take the opportunity to address my performance concerns as well. While my original intent was to simply clean up the server logic I had written, I realized that it would be more sustainable in the long term to migrate to a true static site using Jekyll.

The Setup

Installing Jekyll locally was a piece of cake; simply running `gem install jekyll` did the trick. I already had a placeholder page in my benburwell.github.io repo, so I `cd`'d to the parent directory and ran `jekyll new benburwell.github.io` to overwrite the old content.

For those unfamiliar with GitHub Pages, anything that you put in a repo named `[your username].github.io` will automatically be served from that URL. You can also create branches named `gh-pages` in your other repos to serve project-specific sites. In addition to serving static content, GitHub Pages will automatically compile sites generated with Jekyll.

Porting Content

Next came what was probably the most time-consuming part of the whole process: converting the Jade layout into pure HTML with Liquid markup. Luckily, this wasn't too painful, and I came out with two layouts, page structure and navigation, and the other for displaying Posts.

My next challenge was to maintain my link structure so nothing would be broken. The one exception I

conceded to was my résumé, a PDF file that I had been serving from `/resume/` using Express (admittedly a pretty poor idea). After exploring the Jekyll documentation, I discovered that an easy way to separate out my content into Writing and Projects as I've done on my site was to use the built-in category functionality. I would simply create two category pages at `/writing/index.html` and `/projects/index.html` to render a list of posts from their respective categories, and tag each Markdown document with the appropriate category. The final step was to define my permalink structure in `_config.yml` which I did by adding `permalink: /:categories/:title/` to the file.

I next had the pleasure of renaming all of my content files to adhere to Jekyll's naming convention (`YYYY-MM-DD-hyphen-separated-title.markdown`) and adding/modifying the front matter as necessary.

Additional Configuration

I decided to enable the `jekyll-sitemap` plugin by adding `jekyll-sitemap` as a gem to `_config.yml`. This plugin will generate an XML sitemap that can be used by crawlers such as those run by search engines to help determine what content needs to be indexed.

I moved my error page over and quickly translated the Jade to Markdown by following the instructions provided by GitHub for creating a custom 404 page. The only remaining issue was my stylesheet problem. In my Express app, I used Less for writing my stylesheets. As of this writing, Jekyll does not support compiled stylesheet languages like Less, though there is the suggestion of future support for Sass and CoffeeScript.

For now, I'm keeping my stylesheets in `/assets/less/` and compiling them down to a CSS file locally after making changes with `lessc --clean-css style.less ../css/style.css`. While this certainly isn't perfect, it allows me to keep my Less files intact and to serve minified CSS.

Conclusion

All in all, the process went very smoothly. I made the first Jekyll commit at 18:52 and changed my DNS records from Heroku at 21:20, spending about two and a half hours learning Jekyll and converting my site over. This is a pretty rapid deployment — kudos to Jekyll for building such an easy tool.

As far as the future goes, I'd like to see GitHub pages provide native support for a stylesheet language, be it Less, Sass, or some other one. Additionally, I'd like to see an HTML minification plugin (a minor optimization, but not unreasonable). For the time being, I'm quite happily serving this site with GitHub Pages.