Lessons of running a (semi) static, Indieweb-friendly site for 2 years

It's not possible to run fully static sites with dynamic features, such as webmention handling - you can get close to it, but you do need to embrace external services.

In 2016, I decided to leave WordPress behind. Some of their philosophy, mostly the "decisions, not options" started to leave the trail I thought to be the right one, but on it's own, that wouldn't have been enough: I had a painful experience with media handling hooks, which were respected on the frontend, and not on the backend, at which point, after staring at the backend code for days, I made up my mind: let's write a static generator.

This was strictly scratching my own itches[^1]: I wanted to learn Python, but keep using tools, like exiftool and Pandoc, so instead of getting an off the shelf solution, I did actually write my own "static generator" - in the end, it's a glorified script.

Since the initial idea, I rewrote that script nearly 4 times, mainly to try out language features, async workers for processing, etc, and I've learnt a few things in the process. It is called NASG - short for 'not another static generator', and it lives on Github[^2], if anyone wants to see it.

Here are my learnings.

Learning to embrace "buying in"

webmentions

I made a small Python daemon to handle certain requests; one of these routings was to handle incoming webmentions[^3]. It merely put the requests in a queue - apart from some initial sanity checks on the POST request itself -, but it still needed a dynamic part.

This approach also required parsing the source websites on build. After countless iterations - changing parsing libraries, first within Python, then using XRay[^4] - I had a completely unrelated talk with a fellow sysadmin on how bad we are when in comes to "buying into" a solution. Basically if you feel like you can do it yourself it's rather hard for us to pay someone - instead we tend to learn it and just do it, let it be piping in the house of sensor automation.

None of these - webmentions, syndication, websub - are vital for my site. Do I really need to handle all of them myself? If I make it sure I can replace them, if the service goes out of business, why not use them?

With that in mind, I decided to use webmention.io[^5] as my incoming webmention (*it even gave pingback support back*) handler. I ask the service for any new comments on build, save them as YAML + Markdown, so the next time I only need to parse the new ones.

To send webmentions, Telegraph[^6] is nice, simple service, that offers an API access, so you don't have to deal with webmention endpoint discovery. I put down a text file, with slugified names of the source and target URLs to prevent sending the mention any time.

websub

In case of websub[^7] superfeedr[^8] does the job quite well.

~~For syndication, I decided to go with IFTTT~~[^9] ~~brid.gy publish~~[^10]~~. IFTTT reads my RSS feed(s) and either creates link-only posts on WordPress~~[^11] ~~and Tumblr~~[^12]~~, or sends webmentions to brid.gy to publish to links Twitter~~[^13] ~~and complete photos to Flickr~~[^14]

IFTTT didn't work. Well, it worked right after setup, and syndicated one single article properly - since then, it decided to stop looking at my RSS update. I found Zapier[^15] instead. While it can do way more sophisticated, chained actions, that comes at a hefty, 50$/m price. Their free tier includes only 5, simple actions, to that is enough to send updates to WordPress.com, Twitter, Tumblr, Google Groups, and Flickr through brid.gy[^16].

I ended up outsourcing my newsletter as well. Years ago I sent a mail around to friends to ask them if they want updates from my site in mail; a few of them did. Unfortunately Google started putting these in either Spam or Promitions, so it never reached people; the very same happened with Blogtrottr[^17] mails. To overcome this, I set up a Google Group, where only my Gmail account can post, but anyone can subscribe, and another IFTTT hook[^18] that sends mails to that group with the contents of anything new in my RSS feed.

Search: keep it server side

I spent days looking for a way to integrate JavaScript based search (lunr.js or elasticlunr.js) in my site. I went as far as embedding JS in Python to pre-populate a search index - but to my horror, that index was 7.8MB at it's smallest size.

It turns out that the simplest solution is what I already had: SQLite, but it needed some alterations.

The initial solution required a small Python daemon to run in the background and spit extremely simple results back for a query. Besides the trouble of running another daemon, it needed the copy of the nasg git tree for the templates, a virtualenv for sanic (the HTTP server engine I used), and Jinja2 (templating), and a few other bits.

However, there is a simpler, yet uglier solution. Nearly every webserver out in the wild has PHP support these days, including mine, because I'm still running WordPress for friends and family.

To overcome the problem, I made a Jinja2 template, that creates a PHP file, which read-only reads the SQLite file I pre-populate with the search corpus during build. Unfortunately it's PHP 7.0, so instead of the FTS5 engine, I had to step back and use the FTS4 - still good enough. Apart from a plain, dead simple PHP engine that has SQLite support, there is no need for anything else, and because the SQLite file is read-only, there's no lock-collision issue either.

About those markup languages...

YAML can get messy

I went with the most common post format for static sites: YAML metadata

- Markdown. Soon I started seeing weird errors with ' and " characters, so I dug into the YAML specification - don't do it, it's a hell dimension. There is a subset of YAML, title StrictYAML[^19] to address some of these problems, but the short summary is: YAML or not, try to use as simple markup as possible, and be consistent.

```yaml
title: post title
summary: single-line long summary
published: 2018-08-07T10:00:00+00:00
tags:
- indieweb
syndicate:
- https://something.com/xyz
```

If one decides to use lists by newline and `-`, stick to that. No inline `[]` lists, no spaced `-` prefix; be consistent.

Same applies for dates and times. While I thought the "correct" date format is ISO 8601, that turned out to be a subset of it, named RFC 3339[^20]. Unfortunately I started using `+0000` format instead of `+00:00` from the beginning, so I'll stick to that.

Markdown can also get messy

There are valid arguments against Markdown[^21], so before choosing that as my main format, I tested as many as I could[^22] - in the end, I decided to stick to an extended version of Markdown, because that is still the closest-to-plain-text for my eyes. I also found Typora, which is a very nice Markdown WYSIWYG editor[^23]. *Yes, unfortunately, it's electron based. I'll swallow this frog for now.*

The "extensions" I use with Markdown:

- footnotes - *my links are foonotes, so they can be printed*
- pipe_tables
- strikeout - ~~*cause it's useful for snarky lines*~~

- raw_html
- definition_lists - *they are useful, and they were also present on the very first website ever*
- backtick_code_blocks - ``` *type code blocks*
- fenced_code_attributes - *language tag for code blocks*
- lists_without_preceding_blankline
- autolink_bare_uris - *otherwise my URLs in the footnotes are mere text*

I've tried using the Python Markdown module; the end result was utterly broken HTML when I had code blocks with regexes that collided with the regexes Python Markdown was using. I tried the Python markdown2 module - worked better, didn't support language tag for code blocks.

In the end, I went back to where I started: Pandoc[^24]. The regeneration of the whole site is ~60 seconds instead of ~20s with markdown2, but it doesn't really matter - it's still fast.

```bash
pandoc --to=html5 --quiet --no-highlight --
from=markdown+footnotes+pipe_tables+strikeout+raw_html+definiti
on_lists+backtick_code_blocks+fenced_code_attributes+lists_with
out_preceding_blankline+autolink_bare_uris
```

The take away is the same with YAML: do your own ruleset and stick to it; don't mix other flavours in.

Syntax highlighting is really messy

Pandoc has a built-in syntax highlighting method; so does the Python Markdown module (via Codehilite).

I have some entries that can break both, and break them bad.

Besides broken, Codehilite is VERBOSE. At a certain point, it managed to add 60KB of HTML markup to my text.

A long while ago I tried to completely eliminate JavaScript from my site, because I'm tired of the current trends. However, JS has it's place, especially as a progessive enhancement[^25].

That in mind, I went back to the solution that worked the best so far: prism.js[^26] The difference this time I that I only add it when there is a code block with language property, and I inline the whole JS block in the code - the 'developer' version, supporting a lot of languages, weighs around 58KB, which is a lot, but it works very nice, and it very fast.

No JS only means no syntax highlight, but at least my HTML code is readable, unlike with CodeHilite.

Summary

Static sites come with compromises when it comes to interactions, let that be webmentions, search, pubsub. They need either external services, or some simple, dynamic parts.

If you do go with dynamic, try to keep it as simple as possible. If the webserver has PHP support avoid adding a Python daemon and use that PHP instead.

There are very good, completely free services out there, run by ~~mad scientists~~ enthusiasts, like webmention.io and brid.gy. It's perfectly fine to use them.

Keep your markup consistent and don't deviate from the feature set you really need.

JavaScript has it's place, and prism.js is potentially the nicest syntax highlighter currently available for the web.

Links

1. https://indieweb.org/scratch_your_own_itch
2. https://github.com/petermolnar/nasg/
3. http://indieweb.org/webmention
4. https://github.com/aaronpk/xray
5. https://webmention.io/
6. http://telegraph.p3k.io/
7. https://indieweb.org/websub
8. https://superfeedr.com/
9. http://ifttt.com/
10. https://brid.gy/about#publishing
11. https://ifttt.com/applets/83096071d-syndicate-to-wordpress-com
12. https://ifttt.com/applets/83095945d-syndicate-to-tumblr
13. https://ifttt.com/applets/83095698d-syndicate-to-brid-gy-twitter-publish
14. https://ifttt.com/applets/83095735d-syndicate-to-brid-gy-publish-flickr
15. https://zapier.com/
16. https://brid.gy/about#publishing
17. https://blogtrottr.com/
18. https://ifttt.com/applets/83095496d-syndicate-to-petermolnarnet-googlegroups-com
19. http://hitchdev.com/strictyaml/features-removed/
20. https://en.wikipedia.org/wiki/RFC_3339
21. https://indieweb.org/markdown#Criticism
22. https://en.wikipedia.org/wiki/List_of_lightweight_markup_languages

23. http://typora.io/
24. http://pandoc.org/MANUAL.html#pandocs-markdown
25. https://en.wikipedia.org/wiki/Progressive_enhancement
26. https://prismjs.com/

Created by Peter Molnar <mail@petermolnar.net>, published at 2018-08-07 18:33 UTC+01:00, last modified at 2021-05-11 11:49 UTC , to canonical URL https://petermolnar.net/journal/running-a-static-indieweb-site/ , licensed under CC-BY-4.0 .