A MuseWiki

Marco Pessotto
Rationale (second part)

amusewiki.org

Rationale (second part)

Marco Pessotto

```
<idiot_on_irc> You built an albatross for no rea-
son, just to serve a directory of static files
```

A year ago, I got this, let's call it so, "criticism" over a conversation on an IRC channel. Unfortunately the logs got lost, or they never existed, so those could be not the exact words, but the meaning is pretty much intact. Oh, yes, the nickname has been amended.

It's worth nothing that the above IRC user was the same one which, a bunch of years ago, had the idea of building a digital library, gave me a Drupal installation, supervised for a while, then got busy and disappeared, but now and then chimes in, mostly ranting because the said library runs amusewiki and not the Drupal he gave me.

Yes, running amusewiki means basically serving texts. It shouldn't be too complicated, right? We don't even have a commenting system. It turns out that things are never so simple.

Amusewiki is built around the texts. The idea is that each muse file which gets inserted in the archive is in a format which is both human and machine readable, it includes all the needed meta information and can be used locally, without the need of a running site.

Systems running sites sometimes die, get abandoned, etc. if you think over the long term. If I put a lot of work on a text, writing it, correcting it, caring about that, I would love to have a copy of that at least in a file, *with all the logical information intact*, possibly in a revision control system, so I know what I did and when.

At the beginning, to keep track of the work on the Drupal site, I started writing scripts to help me out, and things got out of hand pretty fast, because the syncing back and forward wasn't going to work out, unsurprisingly.

So the first step was to abandon Drupal, take the loot of the texts, and build something around it. It all started with a couple of CGI scripts which, even if not nice to see, was working better (for the editors) than Drupal. At this point the "filtered HTML" was still in use and was a pain to edit (and too error prone).

Of course I started looking around. MoinMoin and ikiwiki were considered, but still weren't the solution (particularly ikiwiki was quite near to get used), because I wanted some features (like the LaTeX PDF and the bookbuilder) which are hard to achieve without having a backend daemon which does the heavy lifting.

The first version of Amusewiki was also written in Perl, using the Dancer framework. The Git backend was integrated, the Bookbuilder worked, but the application didn't scale for multiple sites and for different language. However, the file system structure behind it was exactly the same which amusewiki uses nowadays.

The current version of Amusewiki uses another Perl framework, Catalyst, with a full database behind that, a real localization engine, a full test suite (which takes about 20 minutes to run), but the concept is still intact. You could download the whole archive, and build something else with that. Please note that every time a text on amusewiki is published, a HTML index of the whole archive is built. When you run the backup of the amusewiki working directory, you have also a version of the site which can work both locally without any webserver, you can access the PDF, the EPUB and the sources. This means you can also have sites which runs as mirrors, just syncing that directory and serving static files.

The point here is that even if at some point you won't be running amusewiki anymore, or if I disappear for whatever reason (this is mostly a one-man project), you will still have the whole archive *intact* and in a useful format, and if you use git to do the backups, with the full history of the archive, and without the layout elements which usually make the migrations a pain, and you can do whatever you want with that.

So, to conclude and to reply to the user, I built this system for a real reason, not out of boredom: preserve the texts (without coupling them with the code and the database) and make the editing experience pleasant (read: off-line editing). All the other features (which are not vital) are of course

nice in my opinion, but that's a bonus which comes from having someone actually working on the codebase.