Programming your blog

Ben Klemens

20 July 2009

First, back up to the six-part series on Why Word is a Terrible Program, which you can read on this here blog, or at fluff.info/terrible[1]. There's a PDF version[2] linked from there.

One of the key themes of that series was the importance of having output and formatting independent of the content, and you can see by the many forms of the same text—episodic blog, long essay, paper—that I do practice what I rant.

Here, I'll mention some technical details of this site that may help you to shunt around your own writing. It's not just me talking about me, but using this site as an example of how the Web is assembled. The summary sentence: Web software was written by programmers, so it pays to think like a programmer when putting together beautiful Web sites.

Content and its management The *content management system* is sold to people as *blogging software* and to businesses via the value-add acronym *CMS*. It's all the same: these systems take the principles of structured code and apply it to human-oriented text.

In well-structured code, you first produce a set of small, modular functions. One function reads input, the next searches any text for a given word, another puts any blob of text on the screeen in blue, and given all that, it's trivial to string together these three functions to write a new function to display the results of a lookup of *Steven*.

Blogging software [I can't stand the managementspeak name] asks you to turn your webpage into small units, and then puts those units together for you. You provide a style sheet, some text for the sidebar, a series of blog entries, and the software produces pages accordingly. If it's sold to a business, then employees, product descriptions, and so on are also reduced to a single unit of content each.

Now that the computer has uniform, modular blocks of content, it can string them together via master templates (i.e., the parent functions).

In that respect, CMSes are not particularly high tech, at all. There are many important bells and whistles to be had (comment forms, search boxes, spam filtering), but the core of it is simply specifying a format for blocks of content and helping you paste them together via a template.

[1] `http://fluff.info/terrible`

[2] `http://fluff.info/terrible/terrible.pdf`

This here blog The needs of this blog are not great. For the most part, every page has three blocks of content: The header plus sidebar (which are a unit), the blog entry/entries, the comment form.

You'll notice that the header and sidebar change depending on the page. This is done by the web server, Apache, which has a very standard Server Side Include (SSI) plugin that could potentially serve as blogging software in its own right. Here's the actual text of `about_the_author.html` (minus the actual content), with discussion to follow:

```
<!--#set var="isabout" value="true" -->
<!--#set var="isauthor" value="true" -->
<!--#include file="head.html" -->

<div id="maintext">
<div id="content">

<h3>About the author</h3>
<p> The actual content goes here.  </p>

</div>
</div>
</body></html>
```

The interesting part here is the first three lines, which are directives to the SSI. I can think of no better evidence for my thesis that web software is written by traditional programmers in a relatively traditional mindset than the format of these SSI directives. Compare the web server's `#include file="head.html"` with the C preprocessor's `#include "head.h"`.

You can picture the thought process of the person who first wrote the SSI package: 'Gee, when I write code, I have this nice C preprocessor that lets me include files. I'd like to do that when I post stuff online, but HTML doesn't let me. Maybe I should write a preprocessor for the Web. After all, unlike those people who just post Web pages about ponies, I have the prerequisites to actually write new software, which the pony-posters will eventually be forced to use.'

`head.html` has the usual code which you can almost inspect via your browser's *view source* option for this page (The web page version, I mean). But you won't quite see what I wrote because the server did some editing using the above variables. Here's the code for a single button in the bar across the top of the screen:

```
<li><a href="http://modelingwithdata.org/about_the_book.html"
<!--#if expr="${isabout}" -->class="active"<!--#endif -->
>About the Book</a></li>
```

The variable set on the page itself advises the system to add `class="active"` if the variable `isabout` is true. Otherwise, that blob of text doesn't appear. In coder-speak, this is the most natural thing in the world: the `head.html` macro has some if-then statements that change output depending on the input variables.

Getting back to the structure of this site, there are still some blog-type jobs to be done, like producing archives, the previous/next links, the search box, the main page with the amalgam of several subpages. I'm using a lightly hacked version of Greymatter, which is one of the original blogging systems, and is so unsupported that it seems that the author has disappeared from the 'Net.[3]

But the demands are so simple that it keeps working anyway. Blocks of text are as just as easy to paste together almost a decade later.

I'd hacked it to use tags, but realized that I never use the tags on anybody else's blogs, and stats to which I have access indicate limited use of the tag pages. The search box and the index of titles seems to be preferred.

The content itself The other hack is about how I write the actual content. All blogging software seems to have some form of pidgin HTML that lets you set boldface and easily add links, but it's never quite sufficient out of the box, especially for technical prose like the stuff on this site.

Raw HTML is a pain to write by hand. (Almost) every tag needs an end tag, the syntax for many details like linking are verbose, you have to mark paragraph breaks, and so on. Worse are the XML variants, which are virtually impossible to get right when writing by hand.

'By hand', of course, is a relative term. After all, I'm using a text editor, not a quill, and it has various conveniences. There are HTML- or XML-oriented text editors that take care of all the above garbage for you. My own preference is to do things in a more stripped-down, no-need-for-tools manner.

So I use LaTeX. It can be written without pain in any text editor, and as a bonus, can be compiled on any system to several types of output. That's how you get the PDF and the HTML version of every web page. If you're not a fan of LaTeX, see entry #021 and its comments for other systems that produce HTML with less hassle.

All that said, the process of producing several versions of a block of content is a repetitive script. As you can imagine, every step but the first is basically automatic:

- Write actual content with LaTeX tags, but no head.
- Paste on a head (`cat htmlhead content.tex > newblog.tex`)
- Have sed do some little search/replaces to get some details down
- `latex2html newblog.tex`
- Copy the HTML to the Web; notify Greymatter that it needs to make a new page using the standard template and the new content
- Repeat with `texhead` instead of `htmlhead` to produce the PDF version
- Add a reference in the book version; recompile and reship

[3] `http://compliticytheory.vox.com/library/post/noah-grey-is-leaving-the-internet.html`

Once you've written the actual content, and have headers specifying formatting, the rest is just logistics.

What's with that last item? If you were reading this online, I'd be pointing you to the link to the compilation. There are several reasons for this: first, you've surely noticed that several of these entries are a coherent thread, serialized over several entries. It ain't Dickens, but you might want to have the whole thing in one place. On my side, thinking about how today's stupid little post fits in to a larger scheme forces me out of bad blogging habits like repetitive prose or stream-of-consciousness writing that doesn't go anywhere.

Also, a lot of the world is still not wired, so there's a place for paper. And darn it, the format of paper looks nice; given a choice between reading an article on screen and via PDF (a choice many journal web sites offer), I always go for the PDF. I would *love* to see other blogs offer PDF editions and compilations.

But why rationalize. It's there, and after the setup of treating my content like code, took very little extra effort to implement.