Principles of SSG design

Why yet another static site generator? Pandoc provides nearly everything and gives the desired functionality (code in Haskell, markdown as primary text input, backup with git).

August 4, 2023

Contents

Goals

Goals

- Separate the presentation from the content.
- The structure of the site should map directly to the directory and file structure of the stored content.
- All content should be ordinary text (UTF-8) files, which gives a choice of tools for editing, version management, backup and guarantees long term viability.
- Connect the tools with a full programming language.
- Build a SSG from available packages in Haskell, to reduce the code which requires maintenance.

- Demonstrate integration using the "uniform" approach to wrap packages in integrable interfaces.

Packages available in Haskell

Packages available in Haskell

A number of tools are available from Hackage:

- Pandoc
- shake
- twich
- git
- citeproc
- doctemplates

They should be used as far as possible!

Well designe tools combine cleanly

Well designe tools combine cleanly

Tools which survive the test of time provide abstractions which combine without unexpected interactions and confusions.

Programmable

Programmable

It is my experience when adaptation is needed anything short of a complete (and well designed) programming language leads to an infinite sequence of special case additions bolted on with some ugly screws; I have the impression that this is a imitation of e.g. Sprinkles.

Uniform interfaces for packages used

Uniform interfaces for packages used

Wrap packages into a small interface layer to locally hide differences between packages which hinder integration. In Haskell, such differences tend to show up as multiple options to achieve more or less the same ends. I tend to think this is preferably over the use of a `special prelude` which makes collaboration with others difficult.

My **uniform** approach

- use `Text` as the primary representation and use `uniform-strings` to convert to and from other representations (with local deviations from the rule)
- all functions are pure or in the `ErrIO` monad (`ErrorT Text a IO`); operations in other monads are wrapped.
- represent path to files as with `Path`[1], use `uniform-fileio` for all operations to use the same interface for different file types and use `TypedFiles` to connect file extensions to semantics (i.e. code to transform from the external to the internal format),

[1] Avoid the more general `FilePath` type!

- write top level code in a basic form of Haskell and eschew use of special features, especially not relying on `Template Haskell`(see Haskell style).

Separate Theme *and* Content

Separate *Theme* and *Content*

The data should be separated from the style of presentation, to

- allow changes in the data without entanglement in the typically tricky descriptions of style,
- changes in the style must be applied regularly on all content and should not require editing already existing data.

The theme (templates, css etc.) and the content should be separated, with a documented interface. Default locations for theme and content can be adapted to needs.

Performance

Performance

SSG is mostly a proof of concept and demonstration for small academic homepages, it is not optimized for humongous web pages for large organizations. Performance is not designed in[2]; if performance is too slow for a specific use, localize the issue and tweak the code were performance is an issue.

The use of twich and shake gives a nearly dynamic behavior: changes are reflected quickly in locally served pages, when `ssgbake` is run with the `-w` switch.[3]

The conversion of the markdown to latex is, as currently implemented, slow; it requires separate processes for different steps in the conversion in a suboptimal fashion and could be improved.

[2] The caching mechanism of `shake`, however, should make even large web sites viable

[3] The mechanism works well for local changes in a page but requires some tweaking when an index page is produced during `bake` by collecting data from multiple other pages in a directory; typically delete at least the `index.html` page in the baked homepage and re-run ssgbake.

Character file based to facilitate backup and version management

Character file based to facilitate backup and version management

The storage of content should be in text files which can be edited with any editor[4], versioned with `git` and backed up with ordinary tools[5].

[4] my preference is currently `VScode`, but no special features are relied on

[5] Git operations can conflict with other tools to synchronize file content between installations (e.g. `syncthing`).

Let the directory structure reflect the structure of the site

Let the directory structure reflect the structure of the site

The primary structure of a site should be reflected in the directory structure where the files for the pages are stored.

It is acceptable that the file names can be restricted (e.g. must not include spaces) and facilities to translate the file names into readable titles must be provided.

Many site generator, especially flexible content management systems, use databases like SQL for storage of date; for small sites,

full function databases are too complex and notoriously difficult to integrate, backup and vulnerable to attacks from hacker.

Customization

Customization

A web site is infinitely customizable; the difficulty is to decide which customizations are provided at a *common user* level, which others are accessible only through tweaking the underlying technology and how accessibly such tweaks are.

I have opted for a minimal set of options to customize:

- a banner image with two lines of text,
- a ribbon of links to directories,

and most everything else is simply text in sources for web pages.

Any further adaption relies on some understanding of the underlying technology; accessible is

- the way text from web pages is arranged as `html` file in a template,
- layout is changeable in the `css` files.

Customization beyond will likely require forking the source code and changes in it.

*Documentation **not yet done***

Documentation **not yet done**

Desirable features

Desirable features

Support for the development of a web site

It should be possible to add new pages without them immediately going live.

Allow some parts of a web site to be protected from the public

No everything in a web site should be automatically visible to everybody; it should be possible to protect some pages or groups of pages with passwords or some similar means.

Produced with 'daino' (Version versionBranch = [0,1,5,3,3], versionTags = []) from /home/frank/Desktop/myHomepage/Essays/SSGdesign/06rationale/010Principles.md with latexTufte81.dtpl
arguments