The use of daino, a Static Site Generator (SSG)

Design of a Static Site Generator (SSG).

August 4, 2023

Contents

The design of my Static Site Generator is explained here and gives a detailed account of its functioning.

Architecture of the bake process

The bake process gradually converts the source texts into texts a html server can use (primarily HTML, PDF, JPG) and adds the supplementary files (mostly CSS to describe appearances.)

The architecture, i.e. the combination of implementations of functions to achieve the overall functionality of SSG, can be seen as steps and each step processing an input into some formats which are used by the next.

Command line processing:

Command line processing:

The standard Unix-style command line analyzes the CLI input and passes it to the program. It establishes the directory in which the command was issued.

Processing of Layout (file `settingsN.yaml`)

Processing of Layout (file `settingsN.yaml`)

List of the directory names and locations - to give flexibility on different distribution of the relevant directories. It is possible to have the code, the content (dough) and the directory where the served files are stored in three different locations.

Watching for changes

Watching for changes

The use of `twitch` to watch for changes in the directories where input files exist and triggering the shake organized rebuilding process removes all tests for file changes in one point. If a change is detected, shake is called.

Given that shake is only redoing what is strictly necessary and caches older results, makes false positives — alerts to changes which are not substantiated — not dangerous and can be ignored.

Shake for rebuilding

Shake for rebuilding

Shake is checking for changes in the needed input files with precision and starts redoing what is necessary to update the result - filtering out false alerts from watching for changes.

Shake relies on filenames and specifically extension. It is important that files with different semantics have different extensions; for example, templates must be separated by extension for the specific processor.
Shake is managing all filenames and calls functions in the next (sub-layer). It checks for existence of files and produces error messages when a file is not found — no further error processing for missing files neded.

In cases where files with the same extension (e.g. `html` or `pdf`) are given (in the dough directory) and are produced for some other files given as e.g. `md` files, the processing checks wether a file is given and if not, tries to produce it.

Transformation of filepath to Path

Transformation of filepath to Path

The FilePath typed files are translated to Path type, which differentiates relative and absoulute path to files or directories.

Processing

Processing layers are split again in two: a layer to read or write files (using typed files and typed content) before it is passed to the operations actually manipulating the data.

Issues : how to organize regression tests

In general, testing algebraic properties is difficult for complex data; I have a method to organize regression tests. Results from operations are stored and used for input later. The input and output of the test functions are typed to avoid problems with confusion in types between data written to disk and read from disk.

The construction of a test for a function is limited as another tested function must produce the input data.

Command Line Interface (CLI) for bake

The main program of SSG ssgbake has a command line interface (CLI) which includes switches to direct

The main program is `ssgbake` and it includes some swiches to taylor the run:

- continuous update (`watch`) when filecontent on disk changes update the current produced content to reflect changes applied to the files on disk (`w`)
- start a web server to serve the produced homepage (`s`)
- update the test homepage (`t`), which is included in the code and distributed with it,
- quick run without producing the `pdf` files, which slows down the conversion (`q`)
- help. The swiches include in specific version of SSG are shown (`h`).

30. PackagesUsed.md

The Packages from Hackage used. Primarily pandoc, pandoc-citeproc, doctemplates, but also twich, shake, scotty and aeson, lens, and aeson-lens.

Pandoc

The central component of any modern site generator seems to be Pandoc. At the moment only markdown is used for content and output is html, additionally, `pdf` files for print output.

Pandoc-citeproc allows the inclusion of references and reformat references based on a BibTex file, which includes the details.

Templates

Pandoc includes a template system, [doctemplates] (http://hackage.haskell.org/package/doctemplates). Injects text values from a JSON record (based on labels); it allows conditionals ('$if(label)$ .. $endif$) and loops.

Watching file change : Twitch

Twitch uses FSnotify to connect programmed actions to activities with files. It can be used to notify the process which bakes files about changes in file content.

Caching

Shake is a Haskell version of `make` and can be used to convert a static site (idee in Slick

JSON

The aeson Haskell implementation of JSON is used, together with aeson-lens for getting and setting values in JSON records.

40 use of pandoc . md

The transformation uses Pandoc, in four steps: - read the file into the pandoc structure - extract from the pandoc file all content into a context - convert the context into the target format - fill the context into a template to produce the result (respective a .tex file to process by lualatex)

Read Markdown

Read Markdown

Reads the YAML header and the text content into a `Pandoc` data type. The formating, in the header and the content, is converted from the input format (e.g. markdown) into the internal Pandoc encoding.

This first step could read essentially any format, Pandoc accepts - likely with minimal or no changes in other steps.

Extract all information into `Context`

Extract all information into `Context`

Extract the information in the `MetaValue` type into a `Context MetaValue`; preserves the formating in the Pandoc format, but separated into pieces.

Convert the Data to target format

Convert the Data to target format

The Pandoc structured formatted data are converted to the target format (either Latex encoded as Text or HTML encoded as Text) - each individual piece.

Fill the converted pieces into template.

Fill the converted pieces into template.

The specific templates for `Daino` must be compiled and are then filled with converted pieces - separately to produce the HTML file to be served and the `.tex` file to be processed by `lualatex`, which produces the final `.pdf`.

Produced with 'daino' (Version versionBranch = [0,1,5,3,3], versionTags = []) from /home/frank/Desktop/myHomepage/Essays/SSGdesign/09technical/index.md with latexTufte81.dtpl
arguments booklet