Static Site Generator

A static site generator from pandoc and other available packages on Hackage (e.g. shake, twitch, scotty), influenced by Chris Penner's slick(todo: look at Ema by Sridhar Ratnakumar).

SSG uses files to manage data to permit version management with git. Page appearances are directed with YAML and internally data is structured with JSON. Unlike other Site Generators, for each page a PDF file is produced to gurantee well formated prints.

Index pages are automatically formated, but minimal content must be provided initially.

Layout

The code includes an example site in the `docs/site` directory. It contains a file `settingsNN.yaml` which describes the layout of the site. A correponding file must be created for a new site.

in the `docs` directgory is the separated `theme` directory, which determines the apparence of the site. There must be a link to the theme directory in the dough directory (TODO remove the needs).

Starting with `-t` for `test` selects the settingsfile in the example test site.

Test site

Test with the included example site (in the `docs/site` directory) and the provided `theme` can be extended to include all troublesome cases. The tests are executed with `-t` switch (e.g. `cabal run ssgbake -- -t`). A html server (scotty) is started; the result can be viewed in the broswer at `localhost:3000` (the port can be selected in the `settings` file.

Alternatively, the resulting site can be tested in the browser with - SimpleServer (must be installed from Hackage with `cabal install` - not yet on 8.10 with base 4.14) with `simpleserver -p <portnumber>` or - `python3 -m http.server <portnumber>` running in `ssg/docs/site/baked`.

Defaults

The markdonw file for each page included in the site must contain in the yaml header values for title, author, date etc. Missing values are replaced with defaults, which are stored in the same format in a file.
Missing title and author is replaced by `FILL` - which can be searched for and corrected!

TODO

Processing

The design is based on Shake which is sort of lazy:

Each markdown file produces a page (correlate: for each page expected include a markdown file, even for the index pages!). A markdown page starts Shake with a `need` for the html page. To produce html page, a panrep file must be produced, which then ask for a docrep file which is produced from the markdown file. Shake caches the intermediate files and recreates files only if the source changed, which achieves very fast udates and allow dynamic uupdates of pages.

From each markdown page a pdf is produced. The progression is from the TODO

In this sense, the conversion/transformation progresses in **Waves** and code is kept in modules which each cater for a wave.

Organising Shake:

- main: ssgbake (from app/ssgBake.hs)
- StartSSGprocess missing upload automatically TODO
- shakeAll from Shake2.hs
- convertFiles

The code is in the subdir `ShakeBake`.

Waves: Transformations of pages for the site

- `md`: The each page shown on the site starts as an markdown file with yaml meta information.
- `docrep`: the pandoc format of the page plus the completed metadata (DocrecJSON meta in json form, Docrep as record) DROP ?? docrep and produce directly panrep
  - bakeOneMD2docrep
    - readMarkdown2docrepJSON (the result from pandoc)
    - completeDocRep (complete with defaults, hardcode TODO)
    - addRefs
- `panrep`: Input format for pandoc with metadata as record; same format as docrep, but index completed
- `html`: a page for the browser to show
- `texsnip`: intermediate format of a part of a page
- `tex`: a tex file for a page
- `pdf`: a printable page

Wave MD -> Docrep: md2docrep

The md page is translated by `bakeOneMD2docrep`.

The md files are - read md file with `readMarkdown2` to pandoc (processing of Metadata in Yaml, conversion of text to pandoc) - `pandoc2docrep` produces

`MetaPage` with all meta data, collects the data for the index - `addRefs` adds and transforms the bibliographic data

Wave Docrep2html

Docrep -> Panrep: docrep2panrep conversion of record (but same content), then complete index.

`bakeOneDocrep2panrep` to docrep file by #### Panrep -> html: panrep2html

Wave Panrep2pdf

Panrep -> Texsnip: panrep2texsnip

– problem the details of the call to pandoc – gives latex code, but not a full file

Texsnip -> Tex: tex2latex

– wraps the head and tail around the a list of latex snips peamble code is in ProcessPDF

Tex -> PDF: writePDF1

conversions between file

hint for problems !tests/*, !unusedHaskellSources/*

changed to fourmolu for formatting

problems are - building the refs

Testing strategy for conversions

Each transformation step identified in Shake2 is used to organize the tests for the conversions.

The tests are indexed by the transformation AtoB

Compilation - Build - Run

Compile

With *cabal build* in the ssg directory *cabal install*

```
uses
- ssg.cabal
- *cabal.project*, updated to use the uBase as moved to Hackage

check with *ghcup tui* which version is installed - currently 8.10.4 (ghcup -h)
```

```
There are changes
    - uniformBase
    - uniform-strings
```

test with testsite

- move to ssg
- cabal run ssgbake – -t
- cd docs/site/baked
- python3 -m http.server 3000
- localhost:3000

run on myhomepage (in the folder)

- there must be a file settingsN.yaml
- in LayoutFlags.hs is the current name as "settings3"
- ssgbake