Daino

This ReadMe lists the principles which guided the design of daino

Design of `daino` was guided by some principles which are explained in this `ReadMe`.

Part I.

No proprietary file formats

> The source of the web site is formatted with common, non-proprietary formats.

The source of the web site should be stored in open file formats, which can be exchanged and read by many programs. It should be easy to take a site organized with one SSG and put it into another one. Proprietary formats make it typically hard to extract content, store it in an open format, and to move it to another program --- effectively locking users in.

The principle speaks against use of databases to store content (so called Content Management Systems) which are probably justified for large, very high traffic sites, a use case, `daino` is not design for.

`Daino` organized the source for a web site in text files written in Markdown; they can be edited with any simple text editor[1].

Part II.

Daino organizes a site as a tree

> The web pages are structured as a tree and collected in a directory tree.

[1] Using a *intelligent* editor like Word is inconvenient; `VS code` however works well.

1. Principle: The structure of the site and the structure of it is stored representation should correspond

A web site is presented as pages of hyper-text with links between the pages(Berners-Lee, Hendler, and Lassila, 2001). This logical structure is represented as files and the whole site is collected under a root directory.

The mapping between rendered web pages and the files representing them is crucial in the design:

Each web page is stored as a markdown file.[2]

Each web page in a site is written as a markdown file, which the generator transforms to a HTML file which can be rendered. The structure of the source (`dough`) of the web page is parallel to the directory structure of the `baked` homepage, which can be served by a web server and rendered by a browser.

A markdown page can call for **additional material** and link to other renderable pages not produced from a markdown page.

1.1. Tree structure

The web site starts with a single page[3] from which all other pages can reached in a tree structure.

The web pages are stored as files in directories. The directory tree starts with the root (here `dainoSite/dough`) which contains all the source text for the web pages[4].

Directories store only files and additional information for the presentation of the directory as web page is necessary. For each directory an `index.md` file is added which comments on the directories content and the list of directories is rendered.

Additional content can be stored in `resources` directories[5]

1.2. Correspondence between presentation and storage

The source for web pages, and the web pages in `HTML` formate are stored in a parallel directory structure and correspond to the structure of the web site visible to the user.

Part III.

Pandoc converts from markdown to HTML

> The sources of the web pages are (primarily) written as markdown and converted by Pandoc to HTML.

[2] Additional material can be stored in files in a `resources` directory.
[3] Often called `landing page`.
[4] It contains an additional file `settingsNN.yaml`, currently `settings3.yaml` for the site.
[5] Which must be called `resources`, allother directories are assumed to be conent directories!

1.3. Source files are converted to HTML using Pandoc

The web page sources are translated using Pandoc to HTML and PDF. Pandoc is equally used to convert the markdown sources to latex and then to PDF.

Pandoc would allow three dozens of input formats. At the moment, page sources must be written in the Pandoc markdown language, but essentially any other input Pandoc can read could be used (e.g. `latex`).

2. `Shake` controls the conversion

Shake is an improved `make` producing a desired set of files from sources and rules. It is driven by the correspondence between the `md` files which must be converted to `HTML` and draws in additional files as necessary. It converts files only if changed; files can be watched for changes and automatically converted.

Part IV.

Help with language specific input

> Various conventions to speed up textual input exist and can be supported; currently support for german text input is built in.

Various conventions to type text in non-english languages exist. For example, it is customary to type combinations of American keyboard characters to stand in for those which are not found on the standard American keyboard. For example, when typing a German text, often the umlaut ä, ö and ü are written as ”ae”, ”oe” and ”ue”. Unfortunately, it is not possible to just use a global replace, because some German words contain some such combinations (e.g. Koeffizienten) which must not be written as Köffizienten! Similar conventions exist in other languages to type, e.g. email on standard keyboards. Italians, for example, replace an accented character with a appended apostrophe (italianita').

Daino includes a support program for german language writing which is automatically applied to German texts and replaces umlauts when acceptable. It uses a small list to guide the process, which is not perfect. Omissions can be edited manually and are not affected by later replacements. Commissions must be - on a file by file base - collected in type YAML header as `doNotReplace` list[6]. Such a list remains with the file and need only updated when text is added and the replacement process produces undesired changes.

[6] Specifically useful to allow some english words, like ”blue”, in a German text!

Part V.

Markdown as primary input format

> The web pages are written as markdown text, which allows emphasis, titles, references, images, footnotes etc.

Markdown is an easy to learn and versatile. The list possible formatting is quite comprehensive:

"Markdown may not be the right format for you if you find these elements not enough for your writing:

- paragraphs,
- (section) headers,
- block quotations,
- code blocks,
- (numbered and unnumbered) lists,
- horizontal rules,
- tables,
- inline formatting (emphasis, strikeout, superscripts, subscripts, verbatim, and small caps text),
- LaTeX math expressions,
- equations,
- links,
- images,
- footnotes,
- citations,
- theorems,
- proofs, and
- examples."[7]

In exceptional circumstances additional formatting tricks can be pulled in as HTML code.

3. YAML header

Markdown allows headers to pass metadata about a file (e.g. title, author) in a YAML to processes working on the source text; the format is flexible[8].

[7] Bookdown.
[8] But beware of colons, quotes etc.!

4. Markdown can include images, reference etc.

Markdown allow the inclusion of images, bibliographic references etc. These additional files are stored in `resources` directories[9].

References are always `absolute` with respect to the root[10] or `relative` to the current page[11].

Part VI.

Separate content and presentation (aka theme)

The content of the web pages should be indpendent of the presentation.

5. Presentation can be changed

A change in the presentation style should not affect the content. It must bepossible to move from fixed-width presentation to a presentation which adapts to different screen sized and later to use a Tufte-inspired style without touching the web page content.

Markdown allows to structure the content with hints for the presentation (e.g. title, footnote) but not fixing how these are rendered.

6. Theme directory

The instructions for presentation, the so called `theme` is in a separate directory (here 'dainoSite/theme). It is linked automatically into the baked site.

The theme consists of

- fonts, preferably in the `woff` format,
- images
- cascading style sheets (CSS) in `static` folder.

The elements are brought together with the content using the Pandoc templates and a template to construct a `LaTeX` input file to produce the PDF.[12]

[9] `resources` is a reserved word; all other directory names are treated as content directories
[10] starting with a "/"
[11] not starting with "/"
[12] Currently `master7tufte.dtpl` for HTML output and `latex7.dtpl` for PDF output.

Part VII.

Produced web site is self-contained

> The Static Site generated is selfcontained. It can be served by any web server.

The files in the `baked` directory includes everything a web server needs to access and is relocatable. It can be copied to become the web root of a server.

Any web server to which a user can upload files to the web root can be used.[13]

References

Berners-Lee, Tim, James Hendler, and Ora Lassila (2001). "The Semantic Web". In: *Scientific American* 284.5, pp. 28–37.

Produced with 'daino' (version Version versionBranch = [0,1,5,3], versionTags = []) from /home/frank/Workspace11/dainoSite/ReadMe/index.md

[13] I currently use a service giving my a `cpanel` to which I can upload with `ftp`; perhaps not the most convenient solution but sufficient.