MarkDown Thesis FrameWork

Jan Sobeslav

February 2018

Proclamation

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla aliquet erat justo, sed finibus nisi interdum tempor. Sed sit amet ullamcorper mi. Suspendisse purus est, vehicula et dapibus nec, gravida non odio. Aenean tincidunt, dui at ultrices vulputate, turpis felis elementum nulla, et mollis enim nisl ut odio.

Thanks

Nunc est lacus, rutrum quis sodales ut, scelerisque ut nibh. Fusce a risus vel nisi condimentum euismod. Curabitur porta, dolor sit amet congue gravida, est elit ultricies odio, sit amet gravida sem ex sit amet arcu.

Abstract

Lorem Ipsum dolor sit amet

Keywords

markdown

Table of contents

List of tables

List of figures

Overview and philoshophy

As stated in the readme, this framework is a tool that utilizes Pandoc[1] to create beautiful standalone ebooks and documents from multiple MarkDown[2] markup files. The text contents are written in plain text, and can therefore be tracked by version control system, like Git, and before exporting to pdf or epub, the MD code is translated to, and interpreted as LaTeX[3].

The main reasons, why I decided to try this approach, instead of continuing to use any office document processing tool, were following:

- Modular nature: I wanted to divide a lengthy document into more managable set of files - chapters. That way, I can see two chapters side by side and modify them parallelly without scrolling all the time (I figured out, that such feature would be invaluable, only after writing a 13k words bachelor thesis)
- Version control: I thought it would be useful to see different versions of the document, and changes made between them. Moreover, remote VCS makes perfect backups, and even makes possible collaboration way easier
- Clearer style: Since MD markup is quite limited, I like to think of it as of a benefit: I hope it will give the document more unified, more readable look

Features checklist

There are certain features this framework in general, and other tools I use, must provide for me. Some features must be satisfied by recommended text processing program, other by the MD-to-PDF compilation tool. It's important to note, that theese points are my personal requirements, and I present them only to give an idea on the philosophy behind the project, and what could it give you.

[1] https://pandoc.org

[2] https://daringfireball.net/projects/markdown/

[3] https://www.latex-project.org

As noted further in the docs, the framework relies on Pandoc[4] tool, and I personally chosed Haroopad[5] as text processor, which suits me the most.

My text processor requirements:

- [x] Live compilation result preview
  - I need to see the result of MD processing as I write. I prefer two-pane layout, instead of WYSIWIG, and this requirement is satisfied by Haroopad
- [x] Ability to view multiple files simoutanously
  - Haroopad works in windowed mode, and therefore it is possible to arrange the files for convenience
- [x] Multiple monitor support
  - Again, windowed mode enables spreading the work on multiple monitors. This is feature, which might not be able to accomplish with a single-window IDE (which could, however, at least support viewing multiple files at the same time)
- [ ] Tabbed interface of opened documents
  - Haroopad does not have this feature. Instead, it works in windowed mode, which is not perfect to me personally, but it is satisfactory enough
- [x] Diacritics support (UTF8)
  - Haroopad works well with UTF-8

Framework requirements and goals:

- [x] Git support
  - MD files are basically plain-text and therefore perfectly compatible with any VCS
- [x] Export: Export to PDF (EPUB), merge of multiple files
  - Pandoc does this, and much more
- [x] Content: images, code preview
  - Works like a charm, as seen in example chapter
- [ ] Content: tables
  - It's difficult to maintain the tables in plain text, but it's possible, as seen in example chapter. I intend to add CSV or XLS import function.
- [x] Organization: Table of contents, list of tables, list of figures, bibliography (References)
  - Another few of nice features Pandoc$\LaTeX$provides
- [ ] Organization: pagebreaks, heading numbering
  - Hasn't been tested yet
- [x] Style: Template support
  - There are templates for both document and chapter
- [ ] Custom formatting / style
  - Templates are working, but they yet need some tweaking and cleaning. Once that

[4] https://pandoc.org

[5] http://pad.haroopress.com/user.html

had been done, it should be easier to add more formating options.

- [ ] Formalized FW
  - In the end, I want to present a publishable version of the project, with accurate docs and all features working and tested
- [ ] Generalize the script
  - I want to make it possible to use the script with multiple documents. Perheaps, some GUI, or file watchers may be usefull, too.

News

I made it possible to declare a variable within chapter file itself, which overwrites global declarations, and can be used in that very file (as can global variables be used, naturally). I will update the docs with next commit.

I need that functionality because of the new `template_chapter` file, a template that wrapps every chapter to get more control over the result formating. I want to use variables in this template as I do in `template_document`.

Once this feature has been done, I may move some variables (notably the ones related to table of contents, and bibliography) to separate chapters, so that `metadata_content` and `metadata_style` would only contain variables, that affects the whole document, regardless of its content. I also want to remove the `header-includes` variables, since they contain TeX declarations that may be incorporated into the document template, having only its control variables present in the YAML (I would be user friendly, having variables like `break-lines-after-chapter: true` instead `after-includes: \pagebreak`).

Finally, another feature, that I have in mind, is using CSV or XLS tables as resource included in the file much like an image. That's important, since noone wants to write table in plain text, does someone?

Another minor change that had been done is rewritting the `templates` folder structure in a way, that makes it possible to store multiple custom templates for different document types. Another step in that direction will be generalizing the script for use with multiple documents (it is possible to simply copy the framework for each document, but that does not seem right).

- v0.0.4.0, March 2018

I realized, that compiling the MD template to LaTeX may not be optimal, and I will probably try to tweak the LaTeX tamplate instead. My goal is to make all the settings done by YAML variable declarations, so that I wouldn't have to touch LaTeX every time.

Moreover, when writing the example section, I realized, that writing tables in MD isn't user friendly at all, and I would like to seek another solution.

- v0.0.3.0, February 2018

I have just finished the first working verion of the framework, and will test it on a real example soon. When that happens, I will probably update the tool with some fixes, or additional options.

I am quite happy with the results so far, with the only exception being the template format. I wanted the template to be written in MD as well and to be translated to LaTeX during the compillation, but there were errors, and that's why I left the template in LaTeX. I will try to look into it later.

- v0.0.1.0, February 2018

Docs

Dependencies

The framework requires Pandoc[1] tool installed. If You need to export to PDF, you will also need MiKTeX[2], as Pandoc docs recommend.

I also had an issue with following error: `pdfTeX error (font expansion): auto expansion is only possible with scalable fonts`. I found solution in this latextemplates issue thread[3], where user by the name koppor recommended installing `cm-super` MiKTeX plugin, which has worked for me.

MarkDown processing

I use Haroopad[4] to process MD files, since I like the two-pane interface more, than any WYSIWIG, like what Typora[5] offers. Unfortunatelly, Haroopad has not been updated for a long time, and might have some issues, especially in the future. Nevertheless it still works for me, which is why I am leaving it here as a recommendation.

A decent list of other recommendations is available in this SitePoint article[6].

Moreover: You can use any IDE with MD highlight support: NetBeans, for example, offer optional MarkDown plugin (with live preview functionality), and enables user to split the window area, so that only issue I have had with it, was that the IDE does not break lines, which could be fixed in a few ways (with `Toggle line wrap` plugin, for example).

[1] https://pandoc.org

[2] https://miktex.org

[3] https://github.com/latextemplates/scientific-thesis-template/issues/28

[4] http://pad.haroopress.com/user.html

[5] https://typora.io

[6] https://www.sitepoint.com/best-markdown-editors-windows/

Directory structure:

- `chapters`: The document content itself. Contains MD files.
- `images`: Optional folder required only if you embed any image in the chapters. You may create any folder like this one
- `metadata`: The document meta information declarations, containing data like author name, or list of sources
  - `metadata.yaml`: Content-related metadata; a set of data like author name, in YAML[7] format
  - `bibliography.bib`: Bibliography, a list of references in BibTeX[8] format
- `template`: The general output document markup, style and meta data
  - `template.tex`: A template file, containing intro, tables of contents, figures, and tables, and other document parts, in Pandoc LaTeX template format You can replace it with any of user contributed templates[9]. If you do so, many metadata entries won't probably be working and other might have to be added. In other words: the meteadata files, and template file, are closely related.
  - `style_metadata.yaml`: Style-related metadata, containing data like paper size, or page margins
  - `citation_style.csl`: A citation style declaration. Default file is IEEE standard, but feel free to replace it with any other. You can find thousands of CSL files on Zotero Style Repository[10]
  - `epub_style.css`: A stylesheet declaration for epub format
- `_script`: Operating system batch command files that compile chapters and template into an output in requested format
  - `windows_cmd.bat`: Windows batch file; tested on Windows 10. Check the script variables section bellow
  - Other OS batch command files are missing; If you are writing one on your own, please see the `windows_cmd.bat` comments, and, please, contribute to the project with your solution
  - `_temp`: Temporary files required for compillation process

Use

First of all modify all the content files, and the metadata as well. Once You want to compile the results, navigate to the `_script` folder and run whatever script your OS need. You can

[7] http://yaml.org

[8] http://www.bibtex.org

[9] https://github.com/jgm/pandoc/wiki/User-contributed-templates

[10] https://www.zotero.org/styles

run it ither from CLI, or GUI, if your OS allows to run the script from there, but if the script fails, it's better to go to CLI as you will be able to see the errors.

You might want to look into the script, and modify following variables:

- `extension` (default: `pdf`): Result output format; I will test the results for `pdf` and `epub` only, but other values are possible, as seen in Pandoc docs
- `template` (default: `template.tex`): A chosen LaTeX template file from `template` folder

Again: the script is built on Pandoc[11], so you should check their docs as well.

[11] https://pandoc.org

Functions example

For further reference, see GitHub MarkDown guide[1].

Chapter One, basics

Headings and pagebreaks

Note the two different heading styles used in this document. Both are valid according to original MD specification.

To create a horizontal line content separator, just write a line of three dashes (`-`), or asterisks (`*`). To force pagebreak, use LaTeX `\pagebreak` command.

Text highlighting

This text is bold. Nulla ligula velit, nec ornare felis placerat sed. Morbi accumsan, ligula commodo varius viverra, lectus sem interdum sapien, eget hendrerit velit eros sit amet mi. Fusce convallis est pulvinar, sollicitudin sem eget, suscipit turpis. Vivamus euismod fringilla mauris, vel porta risus porta quis. **This will also be bold**.

This text is italics. Praesent ultrices auctor urna, ut scelerisque arcu euismod at. Vivamus odio elit, tempor quis pellentesque at, sodales non dolor. Suspendisse et lobortis urna. *This text is also italics.*

[1] https://guides.github.com/features/mastering-markdown/

This text is both bold and italics Vestibulum ullamcorper purus nibh, vitae dapibus nisl fermentum et. ***So is this one***.

You can use backticks to emphasize some text, that `ought to be absolutely literal` (mostly in code example comments). a orci bibendum aliquam sit amet nec ex. Nam orci sapien, porttitor a orci eu, hendrerit sodales sem. Quisque pellentesque accumsan nisl id tincidunt. Ut sit amet diam non arcu accumsan gravida non eget erat. Nam sit amet rutrum enim. Nulla mattis ex auctor augue condimentum pharetra.

~~This text is crossed out~~ et hendrerit magna. Maecenas vel elit at est sagittis mollis. In hac habitasse platea dictumst. Pellentesque in bibendum justo, sed aliquam sem. Vestibulum sollicitudin, tellus ac aliquam hendrerit, dolor tortor tempor eros, nec scelerisque quam mauris id ex.

Lists

Morbi nec turpis placerat, consectetur sapien nec, consectetur leo.

This is ordered list example:

1. Sed id tempus turpis
2. Ac sodales risus
3. Pellentesque lobortis eu nibh eget accumsan

It's important to note that the actual numbers you use to mark the list have no effect on the output Markdown produces. It's only about the number-dot sequence. It's also worth noting that it's possible to trigger an ordered list by accident, for example when beginning a line with a date, in which case you may escape the first dot with a backslash (\):

12.8. something.

Donec id justo et eros facilisis maximus. Mauris sed nisl sed ligula malesuada aliquam at sed sem. Praesent ultricies urna non facilisis egestas. Nunc egestas dolor dictum turpis tincidunt lacinia eu eu neque. Donec gravida feugiat velit, eu bibendum ipsum hendrerit porta.

There are three ways how to mark an unordered list:

- Integer tristique risus nisi

- Id tincidunt magna hendrerit nec

- Fusce posuere molestie velit, in pulvinar nibh faucibus nec.

- Pellentesque finibus placerat dapibus.

- Integer blandit urna justo

Fusce rhoncus risus vel sem fringilla euismod. Vivamus eu leo rutrum, porta orci quis, sollicitudin lectus. Donec eget mollis elit. Fusce fringilla consequat sapien id cursus. Aenean volutpat, lacus vitae consectetur rutrum, libero enim suscipit tellus.

This is mixed and nested list example:

1. Phasellus at mollis lectus
    - Egestas fermentum elit
    - Donec aliquam pretium ipsum vitae dignissim.
2. Fusce commodo ante in dolor commodo eleifend.
    1. Maecenas iaculis euismod risus
    2. Hendrerit ultrices nisl porta sit amet.
    3. Interdum et malesuada fames ac ante ipsum primis in faucibus.
        - Nam laoreet magna vel tincidunt porta. Phasellus scelerisque efficitur risus sit amet viverra. Nulla ornare a arcu in varius. Fusce at gravida ex. Quisque at ullamcorper leo, a pulvinar nibh. Etiam orci magna, viverra eu mi in, ultricies facilisis metus.
        - Integer tristique risus nisi
3. Id tincidunt magna hendrerit nec. Fusce posuere molestie velit, in pulvinar nibh faucibus nec. Pellentesque finibus placerat dapibus. Integer

Quisque luctus cursus mattis. Nunc nec ipsum convallis, vulputate felis quis, pharetra nisl. Phasellus at mollis lectus, egestas fermentum elit. Donec aliquam pretium ipsum vitae dignissim. Proin vel eleifend ante. Fusce commodo ante in dolor commodo eleifend. Maecenas iaculis.

Chapter Two, slightly advanced

Links, notes, and references

You have already seen the link example, when linking to the GitHub MarkDown guide[2]. Another way is using this syntax: http://example.com/.

This is how you insert a footnote reference[3]. Pellentesque enim turpis, tincidunt vel nisl nec, consequat molestie arcu. Donec porta risus risus, vestibulum vestibulum ipsum iaculis vel.

This is how you link to bibliography[1] Curabitur congue interdum ex a ornare. Some erat volutpat. In nec words orci, ac dapibus eros. Sed mauris mi, ullamcorper eget egestas eu, suscipit vel libero. In augue sapien, ultrices in congue vitae, commodo a purus.

Images

Nulla lobortis sem non varius pretium. Etiam luctus urna eget lorem posuere, auctor gravida mauris imperdiet. Praesent in porta nibh, eu feugiat mi. Maecenas et risus risus. Maecenas convallis, risus vel cursus varius.

This is, how you input an image. The filepath is relative to this MD file and it should also be notet, that the image does not always stay on the declared position, but instead LaTeX moves it to separate page, if necessary due to its size.

Donec euismod massa a magna blandit convallis. Ut vulputate risus augue, ut dictum diam feugiat non. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

Text blocks

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras auctor ut libero in venenatis. Curabitur maximus dui sit amet aliquet gravida. Aliquam consequat mauris sem, ac elementum arcu vehicula sed. Phasellus non posuere sem.

> **This is how you make a blockquote**
>
> Etiam sagittis ut nisl ut cursus. Fusce id dapibus lorem, in venenatis nibh. Mauris interdum congue dolor sit amet mollis. Morbi pellentesque tellus et libero pulvinar
>
> - Author

[2] https://guides.github.com/features/mastering-markdown/

[3] This is how you declare footnote contents. Etiam luctus urna.

Figure 7.1: A magnificent animal

> Yesteday:
>
> > **Also, you can nest a blockquote**
> >
> > Vestibulum consectetur tempor vestibulum
> >
> > - Also author
>
> A day before yesterday:
>
> > **Also, yo can nest a blockquote**
> >
> > Vestibulum consectetur tempor vestibulum
> >
> > - Also author

Sed sed scelerisque nibh. Etiam nec iaculis nisl, quis fringilla metus. Maecenas vel justo id lectus consequat scelerisque. Phasellus ex nibh, congue nec leo sit amet, consequat elementum dui.

```
// You can also input a preformated code block:
main() {
    printf("hello, world");
}
```

Sed mauris mi, ullamcorper eget egestas eu, suscipit vel libero. In augue sapien, ultrices in congue vitae, commodo a purus.

```
// on github, you can specify the language highlighting
if (isAwesome){
  return true
}
```

Tables

Tables are controlled by pipe character (`|`) denoting a cell boundary, and a line of dashes (`-`) separating the heading. Also, colon (`:`) may denote a column alignment.

MarkDown	Table	Example	
Lorem	Ipsum dolorInteger tristique risus nisi, id tincidunt magna hendrerit nec. Fusce posuere molestie velit, in pulvinar nibh faucibus nec. Pellentesque finibus placerat dapibus.	true	1000
Sit	Nulla ornare a arcu in varius. Fusce at gravida ex. Quisque at ullamcorper leo, a pulvinar nibh. Etiam orci magna, viverra eu mi in, ultricies facilisis metus	false	1500

It's difficult to write tables like this in hand; that's why there is Markdown Tables Generator[4]. It's also quite possible, that you might need to constuct more complicated tables, or control their style more, in which case, you'll probably have to write it in LaTeX itself.

[4] https://www.tablesgenerator.com/markdown_tables

Aenean congue risus ante, in malesuada elit sagittis et. Pellentesque ex ex, viverra quis nisl at, tempor venenatis mi. Aliquam erat volutpat. Nunc quis dictum lectus. Nam lacinia pellentesque elit.

[1] J. Sobeslav, “ MarkDown Thesis FrameWork,” 2018.