April 26, 2023

Hugo-theme-vivliocli

User Guide

mochimo

v0.17.0

Table of Contents

Chapter 1　Getting Started

1.1　Target Environment

This guide is for Windows.

1.2　Installation

Installing Hugo

Install Hugo `v0.94.0` or higher by following the instructions on the official website. After installing, confirm that the version of Hugo is displayed with the following command:

```
hugo version
:: Hugo Static Site Generator vx.xx.x ...
```

Installing Node.js and Vivliostyle

Install the LTS version of Node.js. After installing, confirm the versions of Node.js and npm are displayed with the following command:

```
node -v
:: vX.X.X
:: note: vivliostyle-cli is not working in Node v14.0.0

npm -v
:: X.X.X
```

Next, install vivliostyle-cli with npm. After installing, confirm the version of Vivliostyle with the following command:

```
npm install -g @vivliostyle/cli
vivliostyle -v
:: cli: X.X.X
:: core: X.X.X
```

Creating a Site and Installing a Theme

Create a new site with Hugo. In this example, the site name is `myPDF`.

```
hugo new site myPDF
cd myPDF
```

Download and extract the latest `Source code (zip)` from the releases page, and place it in `/themes/hugo-theme-vivliocli`.

Note

If using Git, it is possible to install it as a submodule.

```
git init
git submodule add https://github.com/mochimochiki/hugo-theme-vivliocli themes/hugo-theme-vivliocli
```

Copying Files from exampleSite

The theme includes an exampleSite that can be used as a template. Copy the necessary files from the exampleSite. Also, delete the default `config.toml`.

```
xcopy /s themes\hugo-theme-vivliocli\exampleSite\config config\
xcopy /s themes\hugo-theme-vivliocli\exampleSite\CI CI\
del config.toml
```

1.3 Creating the First Site

Editing config.toml

Open `/config/default.toml` and comment out or delete the following line:

```
themesdir = "../.."
```

Also, edit `baseURL` and `title`.

```
baseURL = ""
title = "My PDF Site"
```

Creating a Cover

Create `ja/firstpdf` and `en/firstpdf` directories under `content`.

```
mkdir content\ja\firstpdf
mkdir content\en\firstpdf
```

Next, create a `_pdf.md` file under `ja/firstpdf` and edit it as follows. Save it in UTF-8 format. This file will be the cover/preface/table of contents of the PDF. The directory where `_pdf.md` is located will be the unit of PDF output.

```
---
pdfname: 'firstpdf'
doctitle: 'My first pdf'
subtitle: 'subtitle'
header: '2023/1/1'
author: 'auther'
pagesize: 'letter'
book: true
cover: true
toc: true
sectionNumberLevel: 2
colophon: false
outputs:
- vivlio_cover
- vivlio_config
---
```

Note

For details, please refer to _pdf.md.

Creating articles

Next, create articles. Create `_index.md` in `/content/en/firstpdf` and edit it as follows:

```
---
title: "firstpdf"
weight: 10
---

`_index.md` will become the section page on the site and will not be included in the PDF.
```

Also, create `first.md` and `second.md` in `/content/en/firstpdf` and edit and save them as follows:

```
---
title: "First article"
weight: 10
---

## First article

Hello. This is the first article.
```

```
---
title: "Next article"
weight: 20
---
```

```
## Next article

Hello. This is the next article.
```

Confirm that the created articles are displayed by selecting `English` from the `Languages` menu in the Hugo preview displayed with the following command. When you edit and save an article, LiveReload takes effect and the preview is updated.

```
hugo server --config config\default.toml
```

Building

Exit the Hugo preview with `Ctrl+C` and run the following command to build the Hugo site:

```
cd CI\windows
build_pdf.bat
```

> If an error is output, check if the PDF is still open. Overwriting the PDF will fail if it is still open.

The result is `/public_default/en/firstPDF.pdf`. Let's check it.

Customizing section numbers

The output level of chapter numbers such as `1.1` can be customized in `_pdf.md`. Open `_pdf.md` and change the following section:

```
sectionNumberLevel: 2 # -> change to 1
```

In addition, the top-level output format and delimiter can be set in `/config/default.toml`.

```
  [languages.en.params]
    sectionDelimiter = "."
    sectionTopFormat = "Chapter %s" # -> change to "Lesson %s"
```

> For other config parameters, please refer to the config.toml and the Hugo documentation, among others.

Check the following in `/public_default/en/firstpdf.pdf`:

- With `sectionNumberLevel = 1`, only the top-level section number is displayed
- At the same time, only the top-level is displayed in the table of contents of the PDF
- By specifying `sectionTopFormat` as “Lesson %s”, it is displayed as “Lesson 1” instead of “Chapter 1”

1.4 NextStep

- Check the Markdown and Shortcodes that can be used in the manuscript
  - Markdown and Shortcodes
- Customize the cover and colophon of the PDF
  - config.toml
  - _pdf.md
- Create multiple editions
  - Editions

Chapter 2　Markdown and Shortcodes

This is a showcase of Markdown and Shortcodes that can be used in this theme.

2.1　Markdown

2.2　Header 2

Header 3

Header 4

Header 5

Header 6

Text

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Bulleted Lists

- bullet
    - bullet
    - bullet
        - bullet
- bullet
    - bullet

Numbered Lists

1. AAA
2. BBB
3. CCC
    1. AAA
    2. BBB
    3. CCC
        - AAA
        - BBB

4. DDD
5. EEE

Links

Link to Google.

Table

Please also refer to include for more advanced table expressions.

No.	item	note
1	AAA	noteA
2	BBB	noteB
3	CCC	noteC

Quote

> This is a quote.

Inline Code

`inlinecode`.

Code Block

```
Formatted text.
```

```
Formatted text.
```

```
// comment
if (a == b)
{
  return true;
}
```

Footnote

This is a reference to Google[1]. Here is a footnote reference,[2]

Horizontal Rule

Italic

italic

Bold

bold

Strikethrough

~~strikethrough~~

MathJax

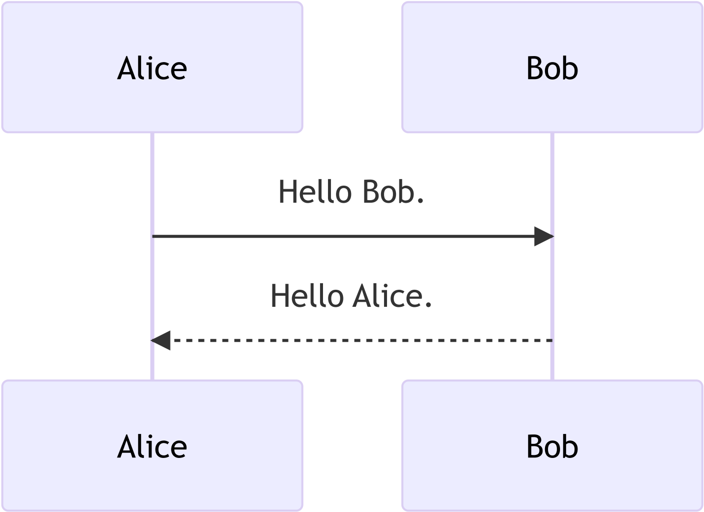
Inline math expressions are written as $y = ax^2 + \frac{b}{c}$. To enable inline math expressions, `math: true` needs to be added in the front matter.

Block math expressions are written as `math` code blocks.

$$\frac{\pi}{2} = \left(\int_0^\infty \frac{\sin x}{\sqrt{x}} dx\right)^2 = \sum_{k=0}^\infty \frac{(2k)!}{2^{2k}(k!)^2}\frac{1}{2k+1} = \prod_{k=1}^\infty \frac{4k^2}{4k^2-1}$$

Mermaid

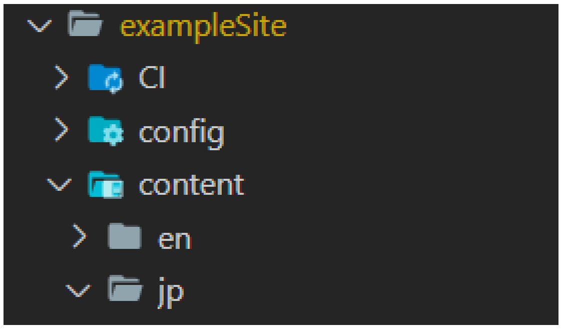
`mermaid` code blocks are rendered as diagrams written in Mermaid.



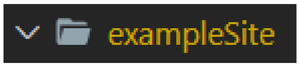
Images

Regular Image

Image without alternate text.



Inline

Display a small image inline. 

Figure

When an alternate text is provided, a figure number and caption will be attached.

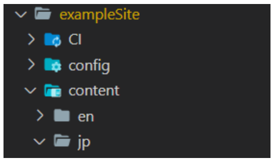


Fig. 2.1: Screenshot of exampleSite

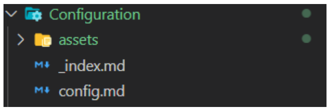
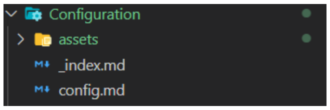


Fig. 2.2: Screenshot of Configuration directory

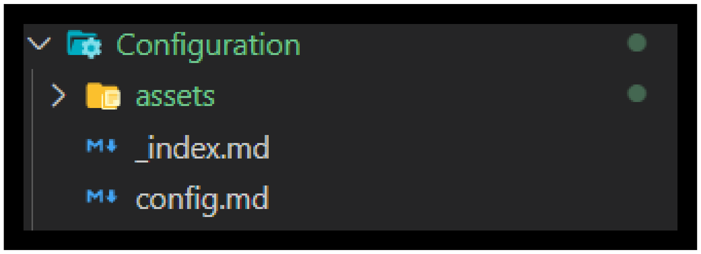
If an alternate text contains only whitespace, no figure number or caption will be attached, and it will be centered.



Changing image size (adding attributes)

By adding attributes as URL query parameters, you can resize images or add borders.

```
![border=10 & width=50%](assets/2021-02-28-22-21-35.png?border=10&width=50%)
```



Configuration
assets
_index.md
config.md

Fig. 2.3: border=10 & width=50%

2.3 Shortcodes

ShowIf

Specify the part to be rendered when it is enumerated in `showIfs` in `config.toml`. The following block is rendered when `showIfs = ["edition1"]`.

```
{{% ShowIf edition1 %}}
Content to be displayed if supporting xxx is written here.
{{% /ShowIf %}}
```

For more information, see Editions.

HideIf

Specify the part that is “not” rendered when it is enumerated in `showIfs` in `config.toml`. The following block will not be rendered when `showIfs = ["edition1"]`.

```
{{% HideIf edition1 %}}
This part will only be hidden when it is edition1.
{{% /HIdeIf %}}
```

For more information, see Editions.

note

The part surrounded by the `note` shortcode is rendered as a note.

```
{{% note %}}
This is where the note article is written.
{{% /note %}}
```

Note

This is where the note article is written.

You can also specify a title as an argument in the format of `note (title)`. You can also write Markdown inside the note.

```
{{% note "Warning!" %}}
This is where the note article is written.

* Markdown is also possible
  * List level 2
* List level 1
{{% /note %}}
```

Warning!

This is where the note article is written.

- Markdown is also possible
  - List level 2
- List level 1

include

You can prepare “components” of Markdown files and csv files in advance and “insert” them anywhere in the article by specifying the relative path starting `.` for loading.

```
{{< include "./test.csv"  >}} # /(md file dir path)/test.csv
```

In addition, if you place the component files in `/content/<language>/_include`, you can include them as follows:

```
{{< include "test_en.md"  >}} # /content/en/_include/test_en.md
{{< include "/sample/sample_en.md" >}} # /content/en/_include/sample/sample_en.md
{{< include "test_en.csv"  >}} # /content/en/_include/test_en.csv
```

- There is no front matter in the Markdown files in the _include directory.
- Write the include shortcode in `{{< >}}` style (no Markdown rendering). If you write it in `{{% %}}` style (with Markdown rendering), the csv loading may not work properly.

include csv

When including a csv file, you can draw a more advanced table than Markdown.

```
{{< include
      src="./rich.csv" # path to source file
      caption="Sample Table" # caption
      class="gray"      # class attribute added to the table. If unspecified, defaults to
"simple"
      markdown=true     # if true, renders markdown inside the table. Defaults to false
      head=1         # if true, treats the first row as a header. Defaults to true. If a number, treats that number of rows as the header.
      align="left"  # align the table left/center/right. left / center / right
      head_align="center" # align all columns in the header of the table left/center/right. left / center / right
      body_align="left" # align all columns in the body of the table left/center/right. left / center / right
      width="90%"       # sets the width attribute of the table.
>}}
```

Table 2.1: Sample Table

id	category	name	notes
001	fruit	apple	
002		orange	• setoka • kanpei
003		grape	delaware
004	snacks	potato chips	• Plain • Salt and Vineger
005		candy	• orange • grape

CSV Format

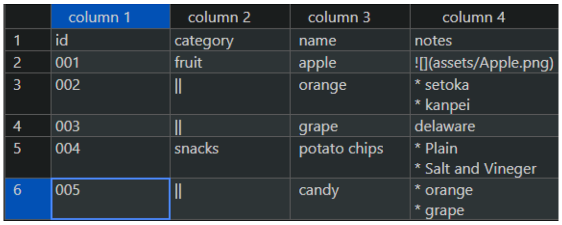
Merge Rows and Columns

You can merge rows vertically using `||` and horizontally using `->`.

ColumnCodes

You can embed ColumnCodes in the header (last row if multiple) to apply them to the column. ColumnCodes are codes in the format `[[@<identifier>=<value>]]`. They apply to the column by adding them to the end of the header cell. If you want to apply multiple codes, you can write them as `[[@<identifier>=<value> @<identifier>=<value>]]` separated by spaces. Here is a list of ColumnCodes.

ColumnCodes	Description
`[[@id=myId]]`	Embeds `myColumnId` as the `class` attribute in all `<td>` tags in this column.
`[[@w=20%]]`	Sets the width of this column to 20% of the table. Columns without this code are treated as `auto`. If any of these codes exist, the `width_ratio` option in the `include` shortcode is ignored.
`[[@h:--]]` / `[[@h:--:]]` / `[[@h--:]]`	Aligns header text to the left/center/right (the number of `-` is arbitrary).
`[[@:--]]` / `[[@:--:]]` / `[[@--:]]`	Aligns body text to the left/center/right (the number of `-` is arbitrary).



	column 1	column 2	column 3	column 4
1	id	category	name	notes
2	001	fruit	apple	![](assets/Apple.png)
3	002	||	orange	* setoka * kanpei
4	003	||	grape	delaware
5	004	snacks	potato chips	* Plain * Salt and Vineger
6	005	||	candy	* orange * grape

Fig. 2.4: rich.csv

Reference Specific Values in CSV

You can also reference specific values in a CSV file by specifying a key to determine the row and a column name. The leftmost column is used as the key. If there are duplicate values, the first one found takes precedence.

```
{{< include "test_ja.csv" "003" "Name" >}} # Value in the "Name" column of the row matching
"003" in /content/en/_include/test_en.csv
```

In the above example, the value in the `Name` column of the row where the first column value is `003` is inserted into the shortcode's position.

1. This is a footnote text https://www.google.com/ ↩
2. Here is the footnote. ↩

Chapter 3 Frontmatter

Explanation of the settings for each section and article:

3.1 Front matter of `_index.md`

By default, `_index.md` is treated as a section at the same level as each article, but by setting `role`, it can be used as a larger framework for parts/chapters.

Note

`_index.md` at the same level as `_pdf.md` will not be output to PDF.

role

`role: doc-part`: `_index.md` is output as the cover page for a part in a PDF.
`role: doc-chapter`: `_index.md` is output as the title of a chapter and the first article in a PDF.

3.2 Front matter of each article

ShowIf

By specifying the `ShowIf` front matter, an article can only be rendered if it matches the keywords listed in the `showIfs` of `config.toml`. The following front matter specifies an article that will only be rendered if `showIfs = ["edition1"]`:

```
---
title: Description of edition1
ShowIf: ["edition1"]
---
```

For more information, see Editions.

HideIf

By specifying the `HideIf` front matter, articles/sections that are hidden only in a specific edition can be created. For more information, refer to Editions.

math

By setting `math: true`, you can write inline equations on the page. Inline equations are written as `$E = mc^2$`. Note that block equations can always be written as follows, regardless of the value of this flag:

~~~
```math
y = ax^2 + bx + \frac{c}{d}
```
~~~

Chapter 4　config.toml

Here is an explanation of the main items in `config.toml`. Please also refer to the Hugo official documentation and the comments in `default.toml` for information about each setting.

4.1　settings under [params]

isPDF

Whether it is a PDF or not. If set to true, menu generation is suppressed on each html page. Set to true when building a PDF.

theme_css

Specify the css used for typesetting. If using `/static/css/yourtheme.css`, specify it as `/css/yourtheme.css`. By default, `style-main.css` in the theme's `static/css` directory is referenced.

showIfs

A list of conditions to switch between showing and hiding using ShowIf/HideIf shortcodes or front matter.

```
showIfs = ["edition1", "edition2"]
```

Please refer to Edition for more details.

sectionNumberLevel

Specify the level at which chapter and section numbers are added. If 0, chapter numbers will not be added.

sectionDelimiter

Specify the delimiter for chapter and section numbers when sectionNumberLevel >= 2. If not set, `.` is used as the delimiter.

sectionTopFormat

Specify the format of the top-level section numbers.
Specify one `%s` in the format, and the section number will be inserted at that position.

Example: `sectionTopFormat = "Chapter %s"`

Chapter 5　_pdf.md

In this theme, the following config files are used to generate the configuration file / cover page (table of contents) / colophon used in vivliostyle:

- _pdf.md
- _pdfcolophon.md

By placing these two files in the directory where you want to generate the PDF on your site, you can generate the configuration file for PDF generation (not just in the site root).

5.1　_pdf.md

`_pdf.md` is the config file and cover page template for vivliostyle. It should be described as follows:

```
---
pdfname:
  default: "pdf file name" # PDF name: must be unique within the site
  # editionA: "editionA file name"
doctitle:
  default: "Cover title" # Title: PDF cover title
  # editionA: "editionA cover title"
subtitle:
  default: "Cover subtitle" # Subtitle: PDF cover subtitle
  # editionA: "editionA cover subtitle"
doc_number:
  default: "Document number" # Document number: PDF cover document number
  # editionA: "editionA document number"
header: '2023/4/7' # Header 1: First line of PDF cover header on the upper right
header2: 'header2' # Header 2: Second line of PDF cover header on the upper right
footer-left: 'left' # Left footer: PDF cover left bottom footer
footer-center: 'v0.16.1' # Center footer: PDF cover center footer
footer-right: 'right' # Right footer: PDF cover right bottom footer
author: 'mochimo' # Author: PDF cover author name
company: 'Company' # Company name: PDF cover company name
logo: 'img/logo.png' # Logo: PDF cover upper left logo
pagesize: 'A5' # PDF size: letter / A5 / A4 / A3 / letter landscape / A4 landscape / A3 landscape
book: true # true: book style false: simple style
marks: false # true: style that outputs crop marks
cover: true # Whether or not to output the cover
toc: true # Whether or not to output the table of contents
sectionNumberLevel: 2 # Output level of chapter/section numbers
colophon: false # Whether or not to output the colophon: _colophon.md is required if true.
outputs: # Config output for VivlioCLI: Basically do not edit.
- vivlio_cover
- vivlio_config
```

```
---

# Introduction

(Description of what to display before the table of contents goes here)
```

Settings specified as `default` or `editionA` in list format can be set for each edition by specifying the edition name specified in `showIfs` in `config.toml`.

pdfname

PDF file name. Required. Make sure it is a unique name that does not conflict within `/content/<language>`.

doctitle

The title of the cover. Required. Can be switched for each edition.

subtitle

Subtitle displayed on the cover. Can be switched for each edition.

doc_number

The document number displayed on the cover. Can be switched for each edition.

Other

Other items cannot be switched for each edition. Refer to the comments in `_pdf.md` for explanations of each item.

5.2 _pdfcolophon.md

This is the template for colophon generation. This file is necessary when `colophon:true` is specified in the front matter of `_pdf.md`.

```
---
title: "Colophon"
outputs:
- vivlio_colophon
---

## Title of Colophon

(Text for the colophon)

<div role="doc-colophon">

## Title of Book
```

```
First edition published on xxxx-xx-xx

| | |
| -- | --  |
|Author| Author's name |
|Publisher| Publisher's name |
|Printing and Binding| Printing company's name |

© Copyright Description

</div>
```

title

The title of the colophon, which is not displayed in the body text. If the header displays a title, this title will be used. This field is required.

outputs

This is a necessary description for generating a colophon file. Do not change it. This field is required.

Body Text

The body text of the colophon page, which is the same as a regular file.

`<div role="doc-colophon">` Tag

The content inside this tag is displayed justified to the bottom. It is intended to describe the book's title and publication information in a table format.

Chapter 6 Edition

If you need to generate a family of PDFs with different contents, you can use the Configuration Directory mechanism in Hugo to create editions and switch configurations and article contents for building.

6.1 Creating a config for an edition

Create `/config/myedition.toml`, and specify it with `default.toml` at build time to easily create the `myedition` edition.

Switching the display of specific elements on and off

Use `showIfs` with `ShowIf` / `HideIf` shortcodes / front matter to switch the display on and off.

```
[params]
  showIfs = ["edition1"]
```

With this setting, elements written as follows in the markdown will be displayed.

```
{{% ShowIf edition1 %}}
Write content to be displayed when supporting edition1 here.
{{% /ShowIf %}}
```

Elements written as follows will be hidden only in edition1.

```
{{% HideIf edition1 %}}
Write content to be hidden in edition1 here.
{{% /HideIf %}}
```

Switching the display of specific articles / sections on and off

When writing front matter like this, the article / section will be included in the PDF. It must be written in list format (`["xxx", "yyy"]`).

```
[params]
  showIfs = ["edition1"]
```

If “edition1” is included in config.toml, the following pages will be included in the PDF.

```
---
title: edition1 description
ShowIf: ["edition1"]
---
```

If you use HideIf as follows, it will no longer be included in the PDF for edition1.

```
---
title: Displayed in editions other than edition1
HideIf: ["edition1"]
---
```

6.2 Building

To build with a specified edition, specify the edition as an argument. For example, to build the `other` edition, specify it as follows:

Hugo

```
set HUGO_PARAMS_ISPDF=true
hugo --config "config/default.toml","config/other.toml" -b ""
```

Batch

```
.\CI\build_pdf.bat other
```

Chapter 7　Tagging

You can tag your pages, which allows them to be categorized and linked to from a page displaying all tags.

7.1　Page displaying all tags

If there is at least one page that has been tagged, a link to the page displaying all tags will be displayed below the language selector in the menu.

7.2　How to tag

To tag a page, all you need to do is declare a list of tags to be added to the page in its front matter like so: `tags: ["tagA", "tagB"]`. This will enumerate the tags on the page displaying all tags.

```
---
title: Tagging Pages
weight: 70
---
```

This page is blank

Hugo-theme-vivliocli User Guide

April 26, 2023

Author	mochimo
Editor	mochimo