GitBook 学习笔记

书栈(BookStack.CN)

目　录

致谢

当前文档 《GitBook 学习笔记》 由 进击的皇虫 使用 书栈(BookStack.CN) 进行构建，生成于 2019-05-13。

书栈(BookStack.CN) 仅提供文档编写、整理、归类等功能，以及对文档内容的生成和导出工具。

文档内容由网友们编写和整理，书栈(BookStack.CN) 难以确认文档内容知识点是否错漏。如果您在阅读文档获取知识的时候，发现文档内容有不恰当的地方，请向我们反馈，让我们共同携手，将知识准确、高效且有效地传递给每一个人。

同时，如果您在日常工作、生活和学习中遇到有价值有营养的知识文档，欢迎分享到 书栈(BookStack.CN) ，为知识的传承献上您的一份力量！

如果当前文档生成时间太久，请到 书栈(BookStack.CN) 获取最新的文档，以跟上知识更新换代的步伐。

内容来源：yangzh https://yangjh.oschina.io/gitbook/

文档地址：http://www.bookstack.cn/books/yangzh-gitbook

书栈官网：http://www.bookstack.cn

书栈开源：https://github.com/TruthHun

分享，让知识传承更久远！ 感谢知识的创造者，感谢知识的分享者，也感谢每一位阅读到此处的读者，因为我们都将成为知识的传承者。

GitBook 学习笔记

这本电子书主要记录GitBook工具使用过程中遇到的疑问及解决办法。虽然网络中已经有很多GitBook的资料，但学习技能的最好办法还是实践，而不仅仅是阅读。

这本小书假定读者已经具有操作系统的基本知识，尤其是命令行（终端）和文本编辑器的相关知识。

什么是GitBook

GitBook 有两个含义：一个是基于Node.js的制作电子书的命令行工具，其作品如ReduxJS documentation，一个是提供书写、托管服务的在线平台GitBook。本书是对GitBook命令行工具学习过程的记录。

主要参考资源

- GitBook主页
- Github地址
- GitBook编辑器
- GitBook Toolchain Documentation
- GitBook Documentation
- GitBook使用教程

使用GitBook命令行工具制作文档的步骤

安装 Node.js

GitBook 依赖 Node.js 环境。如果您的系统中还未安装 Node.js，请到Node.js官方网站，根据你所使用的系统下载对应的版本。如果已安装则略过本步骤。

安装 GitBook 命令行工具

打开“命令提示符”（Mac 系统打开“终端”）输入以下命令安装 GitBook：

```
1. npm install gitbook-cli -g
```

GitBook命令行工具一旦安装后，就无需重复安装，利用该工具可以创建多个电子书。

这条命令还可以用来更新gitbook命令行工具。

新建GitBook项目

新建一个目录，并进入该目录使用 gitbook 命令初始化电子书项目。举个例子，现在要创建一个名为“MyBook”的空白电子书项目，如下所示：

```
1. mkdir MyBook #新建目录
2. cd MyBook    #进入目录
3. gitbook init #初始化目录
```

初始化后的目录中会出现“README.md”和“SUMMARY.md”两个基本文件。

编辑电子书内容

首先，GitBook使用 `SUMMARY.md` 文件组织整个内容的目录，如：

```
1. # Summary
2.
3. * [简介](README.md)
4. * [常见问题](Faq.md)
```

如上面目录中“常见问题”章节的内容，就存放在名为 `Faq.md` 的纯文本文件中，md文件使用markdown语法编辑，规则非常简单，几分钟就可以学会。

预览电子书

当内容书写完毕后，可以在终端中输入如下命令，实现实时预览：

```
1. gitbook serve
```

`gitbook serve` 命令实际上会首先调用 `gitbook build` 编译书籍，完成以后会打开一个 web 服务器，监听本地 4000 端口，在浏览器中输入 `http://localhost:4000` ，即可打开电子书。

发布电子书

当电子书内容制作好后，可以使用如下命令，生成html版本的电子书：

```
1. gitbook build
```

该命令会在当前文件夹中生成 `_book` 文件夹，用户可以将这个文件夹内容托管到网上，从而实现内容的发布。

如何制作多级目录

GitBook使用 `SUMMURY.md` 文件实现目录结构的设定，在该文件中，可以通过缩进实现多级目录的效果，如：

```
* [第一章](section1/README.md)
    * [第一节](section1/example1.md)
    * [第二节](section1/example2.md)
* [第二章](section2/README.md)
    * [第一节](section2/example1.md)
```

如上所示，章节的md文件，还可以存放在子文件夹中。另外，GitBook的目录，限定为三级。

用户还可以通过使用标题或者水平分割线标识将GitBook目录分为几个不同的部分：

```
# Summary

### Part I

* [Introduction](README.md)
* [Writing is nice](part1/writing.md)
* [GitBook is nice](part1/gitbook.md)

### Part II

* [We love feedback](part2/feedback_please.md)
* [Better tools for authors](part2/better_tools.md)

----

* [Last part without title](part3/title.md)
```

如何插入图片

在GitBook中插入图片的方式有两种，使用绝对路径和相对路径。

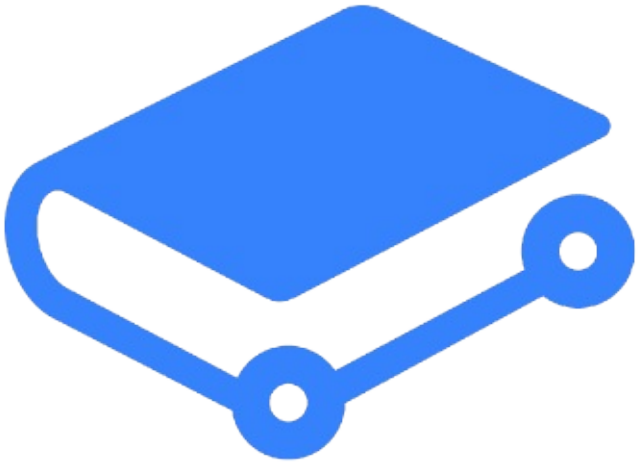
下面的这张图片存放在维基百科：

采用绝对路径的方式加以引用：

```
1. ![Markdown logo][1]
2.
   [1]: http://upload.wikimedia.org/wikipedia/commons/thumb/4/48/Markdown-mark.svg/208px-Markdown-mark.svg.png
3. "Markdown logo"
```

如果用户要自己上传图片并加以引用，就可以采用相对路径的方式，先在项目中创建存放图片的目录，如 `images` ，然后将图片保存至此目录，之后这样引用：

```
1. ![PNG](\images\gitbook.png)
```



如何在表格中使用竖线|

在markdown语法中，表格是使用竖线来组织的。但有时候，需要在表格的内容中加入竖线，如不做转换，则出现表格多出单元格的情况，查阅资料后，解决方案如下：

```
1. | $a &#124;&#124; $b | 逻辑或 | TRUE，如果 $a 或 $b 任一为 TRUE。 |
```

解决思路是使用HTML实体符号来表示竖线，这一思路还可用在其他特殊符号的输入上。

显示效果如下：

例子	名称	结果
$a || $b	逻辑或	TRUE，如果 $a 或 $b 任一为 TRUE。

如何在url中使用括号

markdown标记url时，需要使用方括号和圆括号，语法如下：

```
1. [文字信息](http://xxxx)
```

如果url中本身包含圆括号，就会导致链接地址出错，如：

```
2. [Journal of Communication](http://onlinelibrary.wiley.com/journal/10.1111/(ISSN)1460-2466) 传播学刊。影响因子
1. 3.843[^2]。
```

效果是这样：Journal of Communication1460-2466)

解决办法是将行内式的链接地址，改写成参考式的地址：

```
1. [Journal of Communication][1]
2.
3. [1]: http://onlinelibrary.wiley.com/journal/10.1111/(ISSN)1460-2466
```

效果如下：Journal of Communication1460-2466)

如何安装插件

插件是GitBook拓展功能的主要机制，插件的使用步骤如下：

在book.json文件中添加插件名称

book.json文件中，plugins选项用来添加、关闭插件，如下面的内容表示关闭sharing插件，使用sharing-plus插件：

```
"plugins": ["-sharing", "sharing-plus"],
```

所有插件名称可从https://plugins.gitbook.com/获取。

安装插件

执行下面的命令，gitbook会自动安装所需插件：

```
gitbook install
```

设置插件配置

一般而言，插件都有默认配置，如果需要的话，在book.json中加入插件配置内容即可。例如：

```
    "pluginsConfig": {
        "sharing": {
            "douban": false,
            "facebook": false,
            "google": false,
            "hatenaBookmark": false,
            "instapaper": false,
            "line": false,
            "linkedin": false,
            "messenger": false,
            "pocket": false,
            "qq": true,
            "qzone": true,
            "stumbleupon": false,
            "twitter": false,
            "viber": false,
            "vk": false,
            "weibo": true,
            "whatsapp": false,
            "all": [
                "facebook", "google", "twitter",
                "weibo", "qq", "linkedin",
```

```
                "qzone", "douban"
            ]
        }
    }
```

常用的插件都有哪些？

与Latex制作的pdf相比，GitBook的最大优势在于其能使用js脚本，这使得GitBook在功能上可以不断拓展，比如在文档中嵌入视频、嵌入动画、嵌入js练习，嵌入测试。

常用的插件

- exercises，在文档中增加交互练习内容，目前只支持js语言。
- quiz，在文档中增加测验内容，支持单选、多选、排序。
- include-codeblock，使得GitBook能引用外部独立文档。
- localized-footer，为GitBook的每个页面添加页脚内容。
- page-footer-ex，为文档添加修改时间和版权声明页脚。
- search-pro，为GitBook添加多字节字符搜索，实现中文搜索（默认只能搜索英文）。
- sharing-plus，GitBook默认分享工具的增强版，加入了中国常用的社交网站。
- changyan，为GitBook页面添加畅言评论框。
- iframely，在页面中嵌入常见视频网站内容。
- bibtex-indexed-cite，使用bibtex格式，自动生成参考文献。

本文档用到的插件及配置

{ "title": "GitBook学习笔记", "author": "yangjh", "links": { "sidebar": { "Home": "http://yangzh.cn" } }, "plugins": ["-sharing", "sharing-plus", "include-codeblock", "ace", "quiz", "-lunr", "-search", "search-pro", "collapsible-menu", "bibtex-indexed-cite", "-livereload", "prism", "-highlight", "url-embed", "page-footer-ex"], "pluginsConfig": { "theme-default": { "showLevel": true }, "sharing": { "douban": false, "facebook": false, "google": false, "hatenaBookmark": false, "instapaper": false, "line": false, "linkedin": false, "messenger": false, "pocket": false, "qq": true, "qzone": true, "stumbleupon": false, "twitter": false, "viber": false, "vk": false, "weibo": true, "whatsapp": false, "all": ["facebook", "google", "twitter", "weibo", "qq", "linkedin", "qzone", "douban"] }, "include-codeblock": { "template": "ace", "unindent": true, "edit": false }, "quiz": { "labels": { "check": "提交答案", "showExplanation": "显示解释", "showCorrect": "显示正确答案", "explanationHeader": "释义" }, "text": { "noChosen": "Choose at least one answer", "incomplete": "Some correct answers are missing" }, "buttons": { "showCorrect": true, "showExplanation": true } }, "bibtex-indexed-cite": { "path": "/" }, "prism": { "css": ["prismjs/themes/prism-solarizedlight.css"] }, "page-footer-ex": { "copyright": "By yangzh，使用知识共享 署名-相同方式共享 4.0协议发布", "markdown": true, "update_label": "此页面修订于：", "update_format": "YYYY-MM-DD HH:mm:ss" } }}

如何实现代码的高亮功能

在GitBook生成的静态文档中，默认的代码样式没有高亮，不够醒目。这是由于在书写md时，没有指定代码语言。指定特定语言后，即可实现高亮：

```
<p>test</p>
```

上述效果的代码如下：

````
```html
<p>test</p>
```
````

highlight插件支持的语法

`gitbook-plugin-highlight` 使用 `highlight.js` 作为后台渲染引擎， `highlight.js` 支持的语言及简写可在官方手册中查看。

使用prism插件

系统自带插件的高亮功能并不完善，可使用prism插件增强，该插件需要先禁用 `highlight` 插件。

在配置文件中增加如下信息：

```
{
   "plugins": ["-highlight", "prism", "prism-themes"]
}
"pluginsConfig": {
  "prism": {
    "css": [
      "prismjs/themes/prism-solarizedlight.css"
    ]
  }
}
```

使用 `prism` 高亮插件的优点在于，可以使用不同的配色方案，且语法关键词识别度比 `highlight` 插件高。 `prism` 支持的语言可在其官方网站查询。

如何引入外部文件

在有些情景下，我们需要将外部独立的文件加入到文档中。include codeblock插件可以满足这个需求。

安装并配置插件

include Codeblock插件需要和ace插件配合，因此，在book.json文件中，加入如下内容：

```
"plugins": [
        "include-codeblock",
        "ace"
    ],
"include-codeblock": {
    "template": "ace",
    "unindent": true,
    "edit": true
    }
```

在文档中引入外部文件

如何引入外部文件在有些情景下，我们需要将外部独立的文件加入到文档中。[include codeblock插件][1]可以满足这个需求。## 安装并配置插件include Codeblock插件需要和ace插件配合，因此，在book.json文件中，加入如下内容：`json&#34;plugins&#34;: [ &#34;include-codeblock&#34;, &#34;ace&#34; ],&#34;include-codeblock&#34;: { &#34;template&#34;: &#34;ace&#34;, &#34;unindent&#34;: true, &#34;edit&#34;: true }` ## 在文档中引入外部文件import更加详细的用法，可以查看[include codeblock插件手册][1]。[1]: https://plugins.gitbook.com/plugin/include-codeblock## 引入并渲染外部文件### 引入并渲染本地外部文件Gitbook还提供了引入并渲染外部文件的功能，使用方法如下：`markdown{% include &#34;../SUMMARY.md&#34; %}` ### 引入并渲染git仓库中的外部文件Gitbook还可以引入并渲染Git仓库中的文件内容片段：`markdown{% include &#34;git+https://github.com/GitbookIO/documentation.git/README.md#0.0.1&#34; %}` 该命令会将同级目录中的文件内容，插入到当前位置，并进行解析渲染。效

果如下: {% include "git+https://github.com/GitbookIO/documentation.git/README.md#0.0.1" %}

更加详细的用法，可以查看include codeblock插件手册。

引入并渲染外部文件

引入并渲染本地外部文件

Gitbook还提供了引入并渲染外部文件的功能，使用方法如下:

```
1. {% include "../SUMMARY.md" %}
```

引入并渲染git仓库中的外部文件

Gitbook还可以引入并渲染Git仓库中的文件内容片段:

```
1. {% include "git+https://github.com/GitbookIO/documentation.git/README.md#0.0.1" %}
```

该命令会将同级目录中的文件内容，插入到当前位置，并进行解析渲染。效果如下:

Documentation

This book contains the whole documentation for GitBook and gitbook.io.

You an contribute to improve it on **GitHub**: GitbookIO/documentation.

在页面中添加测验功能

添加quiz插件

在book.json中添加如下内容:

```
{
    "plugins": ["quiz"]
}
```

运行 `gitbook install` 。

设置插件配置信息

```
        "quiz": {
            "labels": {
                "check": "提交答案",
                "showExplanation": "显示解释",
                "showCorrect": "显示正确答案",
                "explanationHeader": "释义"
            },
            "text": {
                "noChosen": "Choose at least one answer",
                "incomplete": "Some correct answers are missing"
            },
            "buttons": {
                "showCorrect": true,
                "showExplanation": true
            }
        }
```

在md文件中插入quiz

What is gitbook used for?

To read booksTo book hotel named gitTo write and publish beautiful booksGitBook.com lets you write, publish and manage your books online as a service.
Is it quiz?

YesNo
This is multiple dropdown quiz, in each dropdown select a correct number corresponding to the dropdown's order

如何添加参考文献

GitBook项目创建时间不长，有些功能可能还没有插件，或者有些插件的功能并不完善，比如参考文献的插件。GitBook有多个使用bibtex生成参考文献的格式，多数很少更新，且功能过于简单。

安装和配置

- 在 `book.json` 中添加如下内容后运行 `gitbook install` ：

```
1.  "plugins": ["bibtex-indexed-cite"],
2.
3.
4.  "bibtex-indexed-cite": {
5.             "path": "/"
6.         }
```

其中path用来指定参考文献库“literature.bib”所在的路径。

- 在项目根目录，新增literature.bib和References.md两个文件，其中literature.bib用来存放参考文献数据，References.md文件中写入如下内容：

```
1. {% references %} {% endreferences %}
```

用法

在需要引用参考文献的地方使用如下命令： `{{"GitBook.com-2017"|cite}}`

```
1. {{ "TLW" | cite }}
```

需要注意，第一、引用名中不能有中文,如 `{{"GitBook.com-2017"|cite}}` 不能为 `{{"中文名称-2017"|cite}}` ；第二、参考文献需要单独用一个文件生成，文件名为“References.md”。

设定参考文献样式

`bibtex-indexed-cite` 插件，目前只支持IEEE的引文格式，且引用没有上标，可通过自定义样式表实现上标效果：

```
1. a[href*="#cite"] {
2.     vertical-align: super;
3.     font-size: 0.8em;
4. }
```

如何插入外部页面

有时候，需要在页面中插入外部的独立页面。如演示页面等。可使用 `url-embed` 插件达到上述目的：

安装

在book.json中加入如下内容，然后运行 `gitbook install` ：

```
{
    "plugins": ["url-embed"]
}
```

用法

这个插件将以iframe的形式，在当前页面插入页面内容。在页面中，使用如下内容指定内容所在地址：

```
{% urlembed %}
https://website.org/stuff/this-is-the-path-name
{% endurlembed %}
```

效果

如何更改电子书标题

在项目根目录新建 `book.json` 的配置文件，写入诸如如下信息：

```
{
    "title": "GitBook学习笔记",
    "author": "yangjh"
}
```

如何自定义样式

由于MarkDown语法本身是HTML语言的一个子集，按照Web标准的精神（内容、表现和行为三者分离），MarkDown语法中，没有和内容表现形式相关的语法规则。因此，要实现自定义样式，必须从CSS或者插件入手。

据GitBook官方手册介绍，我们可以通过在项目文件夹中创建特定样式表的方式达到自定义样式的目的：

> You can specify CSS files to be included in your book's website or PDF builds by creating files:styles/website.css: will apply only to the websitestyles/pdf.css: will apply only to the PDFstyles/ebook.css: will apply only to ebook formats (PDF, Mobi, ePub)

如何为目录添加序号

默认情况下，GitBook的目录是没有序号的，若想为目录编号，需要在book.json中添加如下配置：

```
"theme-default": {
            "showLevel": true
        },
```

为左侧导航栏添加链接信息

在 `book.json` 文件中，添加 `links` 配置项：

```
"links" : {
    "sidebar" : {
        "Home" : "http://yangzh.cn"
    }
}
```

将_book发布到pages上

Pages 是github类网站提供的免费的静态网页托管服务，既然GitBook能生成基于HTTML的静态电子书籍，那自然而然，我们就会有将GitBook静态页面发布到pages服务的需求。

除了能够将书籍发布到GitBook.com外，用户还可以将电子书发布到提供Pages服务的GitHub类站点上，如国内的oschina、coding.net等等。这样做有个好处，就是访问速度比gitbook.com快很多。

这个需求情景实际上是git的应用，即将书籍的源码存放到仓库的master分支，而将书籍的html版本存放到仓库的pages分支，

将书籍源码托管到git远端仓库

这一环节的任务就是将书籍的源码（不含gitbook build生成的内容）推送到远端仓库。

新建仓库

在GitBook项目目录，如 `mybook` 中，执行如下命令，创建本地git仓库：

```
1. git init
```

添加忽略文件

使用文本编辑器，创建名为 `.gitignore` 的文件，内容如下：

```
1. *~
2. _book
3. .DS_Store
```

通过 `.gitignore` 文件，本地仓库将忽略临时文件和 `_book` 文件夹，达到只保存书籍源码的目的。

添加文件

现在可以将本地书籍源码添加到本地git仓库中了：

```
1. git add .
```

添加更新说明

```
1. git commit -m '更新说明文字'
```

建立本地仓库与远端仓库的对应关系

```
git remote add origin https://远程仓库地址.git
```

推送

将本地仓库内容同步到远端仓库：

```
git push -u origin master
```

至此，就完成了将gitbook源码推送到远程仓库的任务，之后书籍内容修改后，执行如下操作即可：

```
git add .
git commit -m '更新说明文字'
git push -u origin master
```

使用pages服务展示gitbook书籍

接下来，需要在原有本地仓库新建一个分支，在这个分支中，只保留_book文件夹中的内容，然后将这些内容推送到远程仓库的pages分支，启用pages服务，最终达到免费发布电子书的目的。

新建分支

```
git checkout --orphan pages
```

新建名为pages的分支，分支名称随意，但最好能反映出分支的用途。

删除不需要的文件

切换到pages分支后，我们需要将_books目录之外的文件都清理掉：

```
git rm --cached -r .
git clean -df
rm -rf *~
```

添加忽略文件

使用文本编辑器，创建名为.gitignore的文件，内容如下：

```
*~
_book
.DS_Store
```

通过.gitignore文件，本地仓库将忽略临时文件和_book文件夹。

复制_book文件夹到分支根目录

```
cp -r _book/* .
```

添加文件

```
git add .
```

添加更新说明

```
git commit -m '更新说明'
```

推送

```
git push -u origin pages
```

现在开启git托管网站的pages服务即可。

上述任务的自动化脚本

命令行的精髓在于可以自动执行，如下面的脚本，可以完成同时更新master分支和pages分支的目的。

```
git checkout master
git add .
git commit -m $1
git push -u origin master
git checkout pages
cp -r _book/* .
git add .
git commit -m $1
git push -u origin pages
git checkout master
```

在需要更新的时候，执行如下命令：

```
sh gitbook.sh '更新说明'
```

多人协作

由于使用了git工具，所以从根本上说，GitBook的多人协作问题，实际上就是git的多人协作。Git的多人协作流程如下：

- 团队成员使用git pull 命令获得最新内容
- 团队成员各自修改本地内容后，执行推送操作
- 如果遇到冲突，再使用pull合并冲突
- 使用push推送内容

参考文献

1	GitBook.com, GitBook Toolchain Documentation, 2017.