目錄

> GitBook 是一个基于 Node.js 的命令行工具，可使用 Github/Git 和 Markdown 来制作精美的电子书。

本文基于GitBook使用 整理并根据实际情况改写部分，将简单介绍如何安装、编写、生成、发布一本在线图书。

支持格式

GitBook支持输出多种文档格式，如：

- 静态站点：GitBook默认输出该种格式，生成的静态站点可直接托管搭载Github Pages服务上；
- PDF：需要安装gitbook-pdf依赖；
- eBook：需要安装ebook-convert；
- 单HTML网页：支持将内容输出为单页的HTML，不过一般用在将电子书格式转换为PDF或eBook的中间过程；
- JSON：一般用于电子书的调试或元数据提取。

Gitbook项目地址:

- GitBook项目官网：http://www.gitbook.io
- GitBook Github地址：https://github.com/GitbookIO/gitbook

基本安装

本章内容讲述Gitbook的安装及命令行的功能快速预览说明

Node.js安装

> Node.js 是一个基于Chrome JavaScript 运行时建立的一个平台， 用来方便地搭建快速的， 易于扩展的网络应用· Node.js 借助事件驱动， 非阻塞 I/O 模型变得轻量和高效， 非常适合 run across distributed devices 的 data-intensive 的实时应用·

Node.js的安装，请移步这里：

> 在 Windows、Mac OS X 與 Linux 中安裝 Node.js 網頁應用程式開發環境

安装完成这后，你可以在终端模式下检验一下:

```
$node -v
v8.9.3
```

看到些提示，就表示你已成功安装上了Node.js。

Gitbook安装

Gitbook是从NMP安装的，命令行：

```
Note: Windows/Linux 下使用如下命令， 不然命令行下不支持：
$npm install -g gitbook-cli
```

安装完之后，你可以检验下是否安装成功：

```
$gitbook -V
CLI version: 2.3.2
GitBook version: 3.2.3
```

如果你看到了与上面类似的版本信息，则表示你已成功完装上了Gitbook，安装了CLI 后，运行gitbook 命令，gitbook会自动安装。

图书项目结构

README.md和SUMMARY.md是Gitbook项目必备的两个文件，也就是一本最简单的gitbook也必须含有这两个文件，它们在一本Gitbook中具有不同的用处。

README.md 与 SUMMARY编写

README.md

这个文件相当于一本Gitbook的简介。

SUMMARY.md

这个文件是一本书的目录结构，使用Markdown语法，新建一个测试用的示例书目录：

```
* [简介](README.md)
* [第一章](chapter1/README.md)
 - [第一节](chapter1/section1.md)
 - [第二节](chapter1/section2.md)
* [第二章](chapter2/README.md)
 - [第一节](chapter2/section1.md)
 - [第二节](chapter2/section2.md)
* [结束](end/README.md)
```

目录初始化

当这个目录文件创建好之后，我们可以使用Gitbook的命令行工具将这个目录结构生成相应的目录及文件：

```
D:\gitbook\mybook>gitbook init
info: create chapter1/README.md
info: create chapter1/section1.md
info: create chapter1/section2.md
info: create chapter2/README.md
info: create chapter2/section1.md
info: create chapter2/section2.md
info: create end/README.md
info: create SUMMARY.md
info: initialization is finished

D:\gitbook\mybook>tree .
文件夹 PATH 列表
卷序列号为 00000002 0236:A566
D:\GITBOOK\MYBOOK
├─chapter1
├─chapter2
└─end
```

我们可以看到，gitbook给我们生成了与SUMMARY.md所对应的目录及文件。 每个目录中，都有一个README.md文件，相当于一章的说明。

图书输出

本章内容讲述如何将编写好的图书输出为我们所需要的格式。目前为止，Gitbook支持如下输出： 静态HTML，可以看作一个静态网站

- PDF格式
- eBook格式
- 单个HTML文件
- JSON格式

我们这里着重说下如何输出静态的HTML和PDF文件。

输出为静态网站

你有两种方式输出一个静态网站： 本地预览时自动生成 当你在自己的电脑上编辑好图书之后，你可以使用Gitbook的命令行进行本地预览：

```
$gitbook serve ./图书目录
```

这里会启动一个端口为4000用于预览的服务器：

```
D:\gitbook\mybook>gitbook serve .
Live reload server started on port: 35729
Press CTRL+C to quit ...

info: 7 plugins are installed
info: loading plugin "livereload"... OK
info: loading plugin "highlight"... OK
info: loading plugin "search"... OK
info: loading plugin "lunr"... OK
info: loading plugin "sharing"... OK
info: loading plugin "fontsettings"... OK
info: loading plugin "theme-default"... OK
info: found 8 pages
info: found 7 asset files
info: >> generation finished with success in 1.6s !

Starting server ...
Serving book on http://localhost:4000
```

这里你会发现，你在你的图书项目的目录中多了一个名为_book的文件目录，而这个目录中的文件，即是生成的静态网站内容。

使用**build**参数生成到指定目录

与直接预览生成的静态网站文件不一样的是，使用这个命令，你可以将内容输入到你所想要的目录中去：

```
D:\gitbook\mybook>mkdir tmp\gitbook

D:\gitbook\mybook>gitbook build --output=tmp\gitbook
info: 7 plugins are installed
info: 6 explicitly listed
info: loading plugin "highlight"... OK
info: loading plugin "search"... OK
info: loading plugin "lunr"... OK
info: loading plugin "sharing"... OK
info: loading plugin "fontsettings"... OK
```

```
info: loading plugin "theme-default"... OK
info: found 8 pages
info: found 7 asset files
info: >> generation finished with success in 1.9s !
```

无论哪种方式，你都可以将这个文件打包，然后把你的书发给你的朋友们了！

输出PDF

输入为PDF文件，需要先使用NPM安装上gitbook pdf：

```
$npm install gitbook-pdf -g
```

> 笔者在安装这个模块的时候，当下载一个名为phantomjs的包时，出现请求未能响应的情况，这时，你可以到这个应用的网站上去下载: http://phantomjs.org/
> 这个包的安装方法，请看网站上的说明。

然后，使用下面的命令，可生成PDF文件了：

```
$gitbook pdf ./图书目录
```

如果你在图书目录内容，也可以这样做：

```
$gitbook pdf .
```

然后，你会发现你的目录里多了一个名为book.pdf的文件，就是它了！ 但是，笔者在测试的时候发现，生成的PDF文件在排版上不是合理，如页面的边距明显过小，如果是图文混排的话，整个版面感觉有些零乱。

发布

本章内容讲述如何使用Github Pages服务将我们的发布到网络上去。 我们假设你已经会使用Git命令和Github的仓库管理。

发布到github pages

将生编写好的格式为.md的文件通过Gitbook处理，然后再发布到Github Gages上去。我个人比较喜欢将源码，即.md文件与Github Pages静态文件存放在一个仓库中。.md文件为master分支，而hmtl文件为gh-pages分支。具体流程是这样的：

创建仓库与分支

- 登录到Github，创建一个新的仓库，名称我们就命令为testGitbook，这样我就就得到了一个testGitbook的空仓库。
- 克隆仓库到本地：git clone git@github.com:USER_NAME/testGitbook.git。
- 创建一个新分支：git checkout -b gh-pages，注意，分支名必须为gh-pages。
- 将分支push到仓库：git push -u origin gh-pages。
- 切换到主分支: git checkout master。 经过这一步处理，我们已经创建好gh-pages分支了，有了这个分支，Github会自动为你分配一个访问网址：

  > http://USERNAME.github.io/testGitbook

你可以在项目页面右下settings中看到：

当然，由于我们内容还没有上传所以你点开链接，也只是一个404页面。

- 同步静态网站代码到分支 下面我们就可以将build好的静态网站代码同步到gh-pages分支中去了：
- 切换出master分支目录。我们需要将gh-pages分支内容存放到另一个目录中去。
- 克隆gh-pages分支：git clone -b gh-pages git@github.com:USERNAME/book.git book-end
  这步我们只克隆了gh-pages分支，并存放在一个新的目录book-end里面。
- Copy静态多站代码到book-end目录中。
- Push到仓库。

然后，等十来分钟的样子，你就可以访问到你的在线图书了。而后，每次修改之后，都可以将生成的代码Copy到book-end目录，再Push一下就OK了。

当然，对于gh-pages存放问题，你也可以直接在master分支目录中直接git clone gh-pages分支，假如名称为book-end，然后修改一下.gitignore文件，将book-end/添加进去，这样主分支就不会理会book-end内容的修改了。

笔者曾试过直接使用gitbook build --output=/PATH/book-end这个方式输出到gh-pages分支目录中，但发现gitbook在build的时候，相当于删除存在的这个目录，然后再新建目录，再写入内容，原来的git信息会完全删除掉，显示这不是我们想要看到的，所以只能Copy才行。

结束

这是笔记在试用了一天的Gitbook之后记录下来的，觉得这真是一个伟大的发明。而后来我喜欢上了更简单的Markdown语法，却发现对于阅读类似于书籍的Markdown时，得自己一个个的文件点开才能进行阅读。

而gitbook则是将这些独立的文件组合起来，做成一个静态站或是生成PDF，且是真正精美的，一下便喜欢上了。

本书记录的内容是简单的，甚至有些地方是含糊的，由于水平尚浅，以至于出现错误，还请详解，如果你有好的想法，也欢迎Fork这个项目。