

Table of Contents

Introduction	1.1
基本安装和使用	1.2
Node.js 安装	1.2.1
Gitbook-cli 安装	1.2.2
Gitbook-cli 基本使用	1.2.3
图书项目结构	1.3
配置说明	1.3.1
图书输出	1.4
输出为静态网站	1.4.1
输出PDF	1.4.2
发布	1.5
上线发布	1.5.1
结束	1.6

Gitbook 使用入门

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。

本书将简单介绍如何安装、编写、生成、发布一本在线图书。gitbook 生成电子书主要有三种方式：

gitbook-cli 命令行操作，简洁高效，适合从事软件开发的相关人员。
gitbook-editor 编辑器操作，可视化编辑，适合无编程经验的文学创作者。但是 **gitbook editor** 的注册和登录需要科学上网。**gitbook.com** 官网操作，在线编辑实时发布，适合无本地环境且科学上网的体验者。

基本安装

包括node和gitbook-cli的安装 以及gitbook的基本使用和一些参数如何配置

Node.js 安装

node.js 是 js 在服务端运行的环境基础,从而使得 js 从浏览器端延伸到服务端领域,而 gitbook 则是运行在 node.js 基础之上的命令行工具,因此必须先安装好 node.js 开发环境.

如果打印出 node.js 版本信息,则表示本机已安装 node.js 环境,跳过此步骤.

```
node --version
```

安装node

Node.js 安装包及源码下载地址为: <https://nodejs.org/en/download/>。

可以根据不同平台系统选择你需要的 Node.js 安装包。Node.js 历史版本下载地址: <https://nodejs.org/dist/>

Linux 上安装 Node.js

Ubuntu apt-get 命令安装（推荐）

```
sudo apt-get install nodejs
sudo apt-get install npm
```

直接使用已编译好的包

Node 官网已经把 linux 下载版本更改为已编译好的版本了,我们可以直接下载解压后使用:

```
wget https://nodejs.org/dist/v10.9.0/node-v10.9.0-linux-x64.tar.xz
tar xf node-v10.9.0-linux-x64.tar.xz          // 解压
cd node-v10.9.0-linux-x64/                    // 进入解压目录
./bin/node -v                                  // 执行node命令 查看版本
v10.9.0
```

解压文件的 bin 目录下包含了 node、npm 等命令,我们可以使用 ln 命令来设置软连接:

```
ln -s /usr/software/nodejs/bin/npm  /usr/local/bin/
ln -s /usr/software/nodejs/bin/node  /usr/local/bin/
```

Ubuntu 源码安装 Node.js

以下部分我们将介绍在 Ubuntu Linux 下使用源码安装 Node.js 。其他的 Linux 系统，如 Centos 等类似如下安装步骤。

在 Github 上获取 Node.js 源码：

```
sudo git clone https://github.com/nodejs/node.git
Cloning into 'node'...
```

修改目录权限：

```
sudo chmod -R 755 node
```

使用 ./configure 创建编译文件，并按照如下顺序执行：

```
cd node
sudo ./configure
sudo make
sudo make install
```

Mac OS 上安装

可以通过以下两种方式在 Mac OS 上来安装 node： 1、在官方下载网站下载 pkg 安装包，直接点击安装即可。 2、使用 brew 命令来安装：

```
brew install node
```

Windows 上安装 Node.js

可以采用以下两种方式来安装。安装完成后可以将node加入环境变量方便使用

1、Windows 安装包(.msi)

在官网找到适合自己的系统的msi文件下载到本地 然后双击运行一路next就行

2、Windows 二进制文件 (.exe)安装

在官网找到适合自己的系统的二进制文件下载到本地 双击运行 点击run 出现命令行界面表示成功

参考链接

<https://www.runoob.com/nodejs/nodejs-install-setup.html>

Gitbook安装

安装gitbook

执行以下命名 耐心等待一会

```
npm install -g gitbook-cli
```

查看gitbook版本

```
gitbook --version
```

正常返回

```
root@hwsrv-941495:~# gitbook --version
CLI version: 2.3.2
Installing GitBook 3.2.3
gitbook@3.2.3 ../tmp/tmp-1993479K7L3rLoIFAVi/node_modules/gitbook
├─ escape-string-regexp@1.0.5
├─ escape-html@1.0.3
├─ destroy@1.0.4
├─ ignore@3.1.2
├─ bash-color@0.0.4
├─ gitbook-plugin-livereload@0.0.1
├─ cp@0.2.0
├─ graceful-fs@4.1.4
├─ nunjucks-do@1.0.0
├─ github-slugid@1.0.1
├─ direction@0.1.5
├─ q@1.4.1
└─ spawn-cmd@0.0.2
```

错误解决

polyfills错误

```
root@hwsrv-941495:~# gitbook --version
CLI version: 2.3.2
Installing GitBook 3.2.3
/usr/lib/node_modules/gitbook-cli/node_modules/npm/node_modules/graceful-fs/polyfills.js:287
    if (cb) cb.apply(this, arguments)
               ^
TypeError: cb.apply is not a function
    at /usr/lib/node_modules/gitbook-cli/node_modules/npm/node_modules/graceful-fs/polyfills.js:287:18
    at FSReqCallback.oncomplete (node:fs:199:5)
```

注释掉对应文件中如下代码

查看是否解决可以再次通过gitbook --version查看

```
fs.chownSync = chownFixSync(fs.chownSync)
fs.fchownSync = chownFixSync(fs.fchownSync)
fs.lchownSync = chownFixSync(fs.lchownSync)

fs.chmodSync = chmodFixSync(fs.chmodSync)
fs.fchmodSync = chmodFixSync(fs.fchmodSync)
fs.lchmodSync = chmodFixSync(fs.lchmodSync)

//fs.stat = statFix(fs.stat)
//fs.fstat = statFix(fs.fstat)
//fs.lstat = statFix(fs.lstat)

fs.statSync = statFixSync(fs.statSync)
fs.fstatSync = statFixSync(fs.fstatSync)
fs.lstatSync = statFixSync(fs.lstatSync)
```


Gitbook命令行速览

查看gitbook版本

```
gitbook --version
```

正常返回

```
root@hwsrv-941495:~# gitbook --version
CLI version: 2.3.2
Installing GitBook 3.2.3
gitbook@3.2.3 ../tmp/tmp-1993479K7L3rLoIFAVi/node_modules/gitbook
├─ escape-string-regexp@1.0.5
├─ escape-html@1.0.3
├─ destroy@1.0.4
├─ ignore@3.1.2
├─ bash-color@0.0.4
├─ gitbook-plugin-livereload@0.0.1
├─ cp@0.2.0
├─ graceful-fs@4.1.4
├─ nunjucks-do@1.0.0
├─ github-slugid@1.0.1
├─ direction@0.1.5
├─ q@1.4.1
└─ spawn-cmd@0.0.2
```

初始化

新建一个文件夹，然后直接执行下面的初始化命令会报错

```
gitbook init
```

```
root@hwsrv-941495:~/gitbook_test# gitbook init
warn: no summary file in this book
info: create README.md
info: create SUMMARY.md
TypeError [ERR_INVALID_ARG_TYPE]: The "data" argument must be of type string or an instance of Buffer, TypedArray, or DataView. Received an instance of Promise
root@hwsrv-941495:~/gitbook_test# ls
README.md
root@hwsrv-941495:~/gitbook_test#
```

建议的方式还是自己先新建好两个文件README.md和SUMMARY.md
README.md是说明文档，SUMMARY.md是电子书的章节目录

启动

执行以下命令启动gitbook服务

```
gitbook serve
```

下图为成功执行

```
root@hwsrv-941495:~/gitbook_test# ls
README.md SUMMARY.md
root@hwsrv-941495:~/gitbook_test# 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 1 pages
info: found 0 asset files
info: >> generation finished with success in 0.7s !

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

打包静态文件

执行以下命令打包静态文件，默认输出到_book/目录。当然输出目录是可配置的,暂不涉及,见高级部分. 打包完成后就可以自行部署到服务器或者github等网站进行托管

```
gitbook build
```

下图为成功执行

```
root@hwsrv-941495:~/gitbook_test# gitbook build
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 1 pages
info: found 0 asset files
info: >> generation finished with success in 0.7s !
root@hwsrv-941495:~/gitbook_test# ls
README.md SUMMARY.md _book
root@hwsrv-941495:~/gitbook_test#
```

图书项目结构

配置说明

gitbook 的目录结构说明

既然要书写一本电子书,那么起码的章节介绍和章节详情自然是必不可少的。

当然还有标题,作者和联系方式等个性化信息需要指定,如果不指定的话,一旦采用默认配合,八成不符合我们的预期,说不定都会变成匿名电子书?所以配置文件一般也是需要手动设置的!

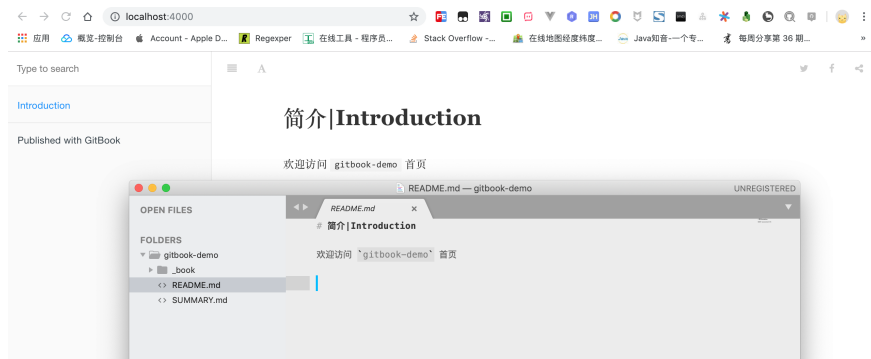
真正可选的文件要数词汇表了,毕竟不是每一本电子书都有专业词汇需要去解释说明.如果在章节详情顺便解释下涉及到的专业词汇,那么自然也就不需要词汇表文件了。

简单解释下各个文件的作用:

- `README.md` 是默认首页文件,相当于网站的首页 `index.html`,一般是介绍文字或相关导航链接。
- `SUMMARY.md` 是默认概括文件,主要是根据该文件内容生成相应的目录结构,同 `README.md` 一样都是被 `gitbook init` 初始化默认创建的重要文件。
- `_book` 是默认的输出目录,存放着原始 `markdown` 渲染完毕后的 `html` 文件,可以直接打包到服务器充当静态网站使用.一般是执行 `gitbook build` 或 `gitbook serve` 自动生成的。
- `book.json` 是配置文件,用于个性化调整 `gitbook` 的相关配置,如定义电子书的标题,封面,作者等信息.虽然是手动创建但一般是必选的。
- `GLOSSARY.md` 是默认的词汇表,主要说明专业词汇的详细解释,这样阅读到专业词汇时就会有相应提示信息,也是手动创建但是可选的。
- `LANGS.md` 是默认的语言文件,用于国际化版本翻译,和 `GLOSSARY.md` 一样是手动创建但是可选的。

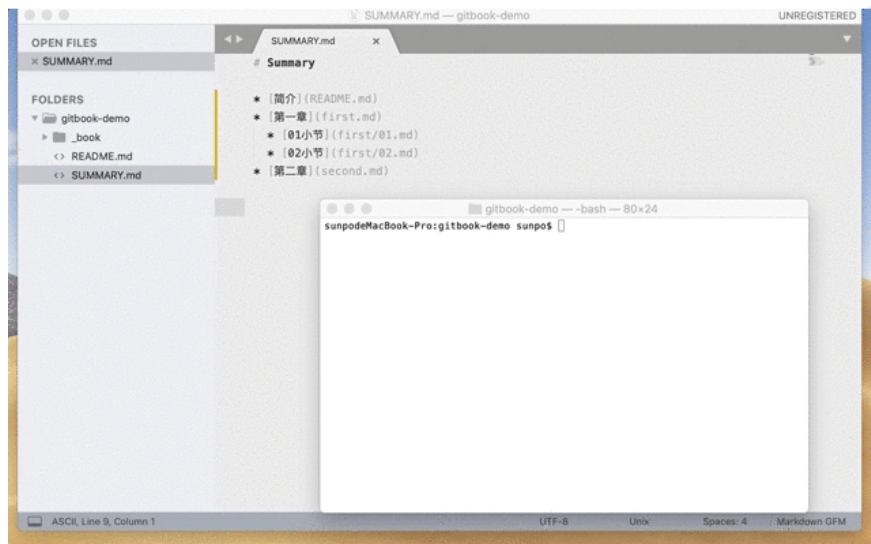
README.md 首页文件[必须]

编辑 `README.md` 文件,随便写点内容并启动本地服务(`gitbook serve`)实时预览效果。



SUMMARY.md 概括文件[必须]

先停止本地服务,编辑章节目录结构,然后重新再初始化(`gitbook init`)
自动创建相应目录.



_book 输出目录[可选]

执行 `gitbook build` 或 `gitbook serve` 命令后会自动生成静态网页.

```
# 构建电子书
$ gitbook build
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 5 pages
info: found 0 asset files
info: >> generation finished with success in 0.7s !
```

```
# 查看输出目录
$ tree _book/
_book/
├── first
│   ├── 01.html
│   └── 02.html
├── first.html
├── gitbook
│   ├── fonts
│   │   └── fontawesome
│   │       ├── FontAwesome.otf
│   │       ├── fontawesome-webfont.eot
│   │       ├── fontawesome-webfont.svg
│   │       ├── fontawesome-webfont.ttf
│   │       ├── fontawesome-webfont.woff
│   │       └── fontawesome-webfont.woff2
│   ├── gitbook-plugin-fontsettings
│   │   ├── fontsettings.js
│   │   └── website.css
│   ├── gitbook-plugin-highlight
│   │   ├── ebook.css
│   │   └── website.css
│   ├── gitbook-plugin-lunr
│   │   ├── lunr.min.js
│   │   └── search-lunr.js
│   ├── gitbook-plugin-search
│   │   ├── lunr.min.js
│   │   ├── search-engine.js
│   │   ├── search.css
│   │   └── search.js
│   ├── gitbook-plugin-sharing
│   │   └── buttons.js
│   ├── gitbook.js
│   └── images
```

```
| | | └─ apple-touch-icon-precomposed-152.png
| | | └─ favicon.ico
| | └─ style.css
| | └─ theme.js
└─ index.html
└─ search_index.json
└─ second.html

10 directories, 28 files
$
```

book.json 配置文件[可选]

在根目录下新建 `book.json` 配置文件,完整的支持项请参考[官方文档](#),下面仅列举常用的一些配置项.

Getting Started

About this documentation

Installation and Setup

Your Content

Directory structure

Pages and Summary

Configuration

Glossary

Multi-Lingual

> Markdown

AsciiDoc

eBook and PDF

Customization

> Templating

> Plugins

Configuration

GitBook allows you to customize your book using a flexible configuration. These options are specified in a `book.json` file. For authors unfamiliar with the JSON syntax, you can validate the syntax using tools such as [JSONlint](#).

General Settings

Variable	Description
root	Path to the root folder containing all the book's files, except <code>book.json</code>
structure	To specify paths for Readme, Summary, Glossary etc. See Structure paragraph .
title	Title of your book, default value is extracted from the README. On legacy.gitbook.com this field is pre-filled.
description	Description of your book, default value is extracted from the README. On legacy.gitbook.com this field is pre-filled.
author	Name of the author. On legacy.gitbook.com this field is pre-filled.
isbn	ISBN of the book
language	ISO code of the book's language, default value is <code>en</code>
direction	Text's direction. Can be <code>rtl</code> or <code>ltr</code> , the default value depends on the value of <code>language</code>
gitbook	Version of GitBook that should be used. Uses the SemVer specification and accepts conditions like <code>">= 3.0.0"</code>

title 标题

书籍的标题

示例:

```
"title": "雪之梦技术驿站"
```

author 作者

书籍的作者

示例:

```
"author": "snowdreams1006"
```

description 描述

书籍的简要描述

示例:

```
"description": "雪之梦技术驿站又名snowdreams1006的技术小屋.主要分
```

isbn 国际标准书号

书籍的国际标准书号

示例:

```
"isbn": "978-0-13-601970-1"
```

选填,请参考 [ISBN Search](#)

language 语言

支持语言项: 默认英语(`en`),设置成简体中文(`zh-hans`)

```
en, ar, bn, cs, de, en, es, fa, fi, fr, he, it, ja, ko, no, pl,
```

示例:

```
"language": "zh-hans"
```

direction 阅读顺序

阅读顺序,支持从右到左(`rtl`)或从左到右(`ltr`),默认值取决于语言值.

示例:

```
"direction" : "ltr"
```

gitbook 版本

指定 `gitbook` 版本,支持[SemVer规范](#),接受类似于 `>=3.2.3` 的条件.

示例:

```
"gitbook": "3.2.3"
```

root 根目录

指定存放 `gitbook` 文件(除了 `book.json` 文件本身)的根目录

示例:

```
"root": "."
```

links 侧边栏链接

左侧导航栏添加链接,支持外链

示例;

```
"links": {
  "sidebar": {
    "我的网站": "https://snowdreams1006.cn/"
  }
}
```

styles 自定义样式

自定义全局样式

示例:

```
"styles": {
  "website": "styles/website.css",
  "ebook": "styles/ebook.css",
  "pdf": "styles/pdf.css",
  "mobi": "styles/mobi.css",
  "epub": "styles/epub.css"
}
```

plugins 插件

配置额外的插件列表,添加新插件项后需要运行 `gitbook install` 安装到当前项目。

`gitbook` 默认自带5个插件,分别是:

- `highlight` 语法高亮插件
- `search` 搜索插件

- `sharing` 分享插件
- `font-settings` 字体设置插件
- `livereload` 热加载插件

后续会介绍一些常用插件,如需获取更多插件请访问[官网插件市场](#)

示例:

```
"plugins": [  
  "github",  
  "pageview-count",  
  "mermaid-gb3",  
  "-lunr",  
  "-search",  
  "search-plus",  
  "splitter",  
  "-sharing",  
  "sharing-plus",  
  "expandable-chapters-small",  
  "anchor-navigation-ex",  
  "edit-link",  
  "copy-code-button",  
  "chart",  
  "favicon-plus",  
  "donate"  
]
```

`pluginsConfig` 插件配置

安装插件的相应配置项,具体有哪些配置项是由插件本身提供的,应访问插件官网进行查询.

```

"pluginsConfig": {
  "github": {
    "url": "https://github.com/snowdreams1006/snowdreams1006.g
  },
  "sharing": {
    "douban": true,
    "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", "instapaper", "linkedin",
      "pocket", "stumbleupon"
    ]
  },
  "edit-link": {
    "base": "https://github.com/snowdreams1006/snowdreams1006.
    "label": "编辑本页"
  },
  "chart": {
    "type": "c3"
  },
  "favicon": "/images/favicon.ico",
  "appleTouchIconPrecomposed152": "/images/apple-touch-icon-pr
  "output": "_book",
  "donate": {
    "wechat": "/images/wechat.jpg",
    "alipay": "/images/alipay.jpg",
    "title": "赏",
    "button": "捐赠",
    "alipayText": "支付宝",
    "wechatText": "微信"
  }
}

```

structure 目录结构配置

指定 `README.md` , `SUMMARY.md` , `GLOSSARY.md` 和 `LANGS.md` 文件名称.

配置项	描述
<code>structure.readme</code>	<code>readme</code> 文件名(默认值是 <code>README.md</code>)
<code>structure.summary</code>	<code>summary</code> 文件名(默认值是 <code>SUMMARY.md</code>)
<code>structure.glossary</code>	<code>glossary</code> 文件名(默认值是 <code>GLOSSARY.md</code>)
<code>structure.languages</code>	<code>languages</code> 文件名(默认值是 <code>LANGS.md</code>)

pdf 配置

定制 `pdf` 输出格式,可能需要安装 `ebook-convert` 等相关插件

配置项	描
<code>pdf.pageNumbers</code>	添加页码(默认值是 <code>true</code>)
<code>pdf.fontSize</code>	字体大小(默认值是 <code>12</code>)
<code>pdf.fontFamily</code>	字体集(默认值是 <code>Arial</code>)
<code>pdf.paperSize</code>	页面尺寸(默认值是 <code>a4</code>),支持 <code>a0</code> , <code>a1</code> , <code>a2</code> , <code>a3</code> , <code>a4</code> , <code>a5</code> , <code>a6</code> , <code>b0</code> , <code>b1</code>
<code>pdf.margin.top</code>	上边界(默认值是 <code>56</code>)
<code>pdf.margin.bottom</code>	下边界(默认值是 <code>56</code>)
<code>pdf.margin.left</code>	左边界(默认值是 <code>62</code>)
<code>pdf.margin.right</code>	右边界(默认值是 <code>62</code>)

电子书封面照片 `cover.jpg` 和 `cover_small.jpg` ,后续会详细说明.

GLOSSARY.md 词汇表文件[可选]

词汇表文件,用于全书的专业词汇解释说明,比如鼠标悬停在专业词汇上会有相应提示.

语法格式: `##` + ` + 专业词汇`

学习 `gitbook` 前最好先学习下 `markdown` 和 `git`,你知道他们的用途吗?

示例:

```
## markdown
```

简洁优雅的排版语言,简化版的`HTML`,加强版的`TXT`,详情请参考 [<https://snowdreams1006.github.io/git>]

```
## git
```

分布式版本控制系统,详情请参考 [<https://snowdreams1006.github.io/git>]

LANGS.md 语言文件[可选]

支持国际化编写图书,一种语言一个单独子目录,同样地,将语言文件放到根目录下.

示例:

```
* [English](en/)
```

```
* [French](fr/)
```

```
* [Español](es/)
```

图书输出

将图书输出为html或者pdf等其他格式

输出为静态网站

打包静态文件

执行以下命令打包静态文件，默认输出到`_book/`目录。当然输出目录是可配置的,暂不涉及,见高级部分. 打包完成后就可以自行部署到服务器或者github等网站进行托管

```
gitbook build
```

下图为成功执行

```
root@hwsrv-941495:~/gitbook_test# gitbook build
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 1 pages
info: found 0 asset files
info: >> generation finished with success in 0.7s !
root@hwsrv-941495:~/gitbook_test# ls
README.md  SUMMARY.md  _book
root@hwsrv-941495:~/gitbook_test#
```

发布

发布到gitpage或者通过自己服务器部署对外提供服务。

发布到线上

Github Pages

1. 在github新建一个仓库
2. 仓库下新建一个gh-pages分支 保证gh-pages分支为空 如果原本main分支有readme 需要删除gh-pages分支下的readme
3. 然后把打包好的_book下的所有文件提交到gh-pages分支等待一会
4. 通过访问<https://用户名.github.io/仓库名/>即可访问

自己通过服务器发布

1. 将打包好的静态html文件上传到服务器的特定位置。
2. 通过nginx设置反向代理即可。将以下代码修改放到nginx配置下的http下面，然后启动或重启nginx服务就好。

```
server {  
    listen      8089;#自定义端口 安全策略需要放通  
    server_name ip;#公网ip  
    location / {  
        root    /usr/_book;#项目地址  
        index   index.html;#首页  
        try_files $uri $uri/ /index.html;  
    }  
}
```

结束

到此你就完成了gitbook安装、书写、打包、部署等的流程，可以愉快的到互联网访问自己的图书了。