Table of Contents

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。

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

本项目地址

- 仓库：https://github.com/tonydeng/gitbook-zh/tree/master/gitbook-howtouse
- 在线阅读：http://tonydeng.github.io/gitbook-zh/gitbook-howtouse

基本安装

Gitbook的安装及命令行的快速功能预览说明。

Node.js安装

> Node.js是一个基于Chrome Javascript运行时建立的一个平台，用来方便的搭建快速的，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-intensivede 的实时应用。

Node.js的安装，请参考：

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

在 Windows、Mac OS X 与 Linux 中安装 Node.js 网页应用程式开发环境

安装完成之后，可以通过下面的命令来验证一下Node.js是否安装成功。

```
$ node -v
v0.10.33
```

Gitbook安装

Gitbook是使用NPM来进行安装的，可以在命令行中输入下面的命令进行安装：

```
npm install gitbook -g
```

安装完成之后，你可以使用下面的命令来检验是否安装成功

```
gitbook -V
1.5.0
```

如果你看到了上面类似的版本信息，则表示你已经安装成功了。

Gitbook命令行速览

Gitbook是一个命令行工具，使用方法：

本地预览

```
gitbook serve ./{book_name}
```

输出一个静态网站

```
gitbook build ./{book_name} --output=./{outputFolde}
```

查看帮助

```
gitbook -h

  Usage: gitbook [options] [command]

  Commands:

    build [options] [source_dir] Build a gitbook from a director
    serve [options] [source_dir] Build then serve a gitbook from
    install [options] [source_dir] Install plugins for a book
    pdf [options] [source_dir] Build a gitbook as a PDF
    epub [options] [source_dir] Build a gitbook as a ePub book
    mobi [options] [source_dir] Build a gitbook as a Mobi book
    init [source_dir]      Create files and folders based on con
    publish [source_dir]   Publish content to the associated git

    git:remote [source_dir] [book_id] Adds a git remote to a boo

  Options:

    -h, --help     output usage information
    -V, --version  output the version number
```

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

使用语法

在Gitbook中所有文字的编写都使用Markdown语法。

README.md

这个文件相对于是一本Gitbook的简介，比如我们这本书的README.md :

```
# 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 Mar

本书将简单介绍如何安装、编写、生成、发布一本在线图书。
```

SUMMARY.md

这个文件相对于是一本书的目录结构。比如我们这本书的SUMMARY.md :

```
# Summary

* [Introduction](README.md)
* [基本安装](howtouse/README.md)
    * [Node.js安装](howtouse/nodejsinstall.md)
    * [Gitbook安装](howtouse/gitbookinstall.md)
    * [Gitbook命令行速览](howtouse/gitbookcli.md)
* [图书项目结构](book/README.md)
    * [README.md 与 SUMMARY编写](book/file.md)
    * [目录初始化](book/prjinit.md)
* [图书输出](output/README.md)
    * [输出为静态网站](output/outfile.md)
    * [输出PDF](output/pdfandebook.md)
* [发布](publish/README.md)
    * [发布到Github Pages](publish/gitpages.md)
* [结束](end/README.md)
```

目录初始化

当SUMMARY.md创建完毕之后，我们可以使用Gitbook的命令行工具将这个目目录结构生成相应地目录及文件

```
$ gitbook init

$ ls
README.md    SUMMARY.md    book    end    howtouse    output

$tree
.
├── LICENSE
├── README.md
├── SUMMARY.md
├── book
│   ├── README.md
│   ├── file.md
│   └── prjinit.md
├── howtouse
│   ├── Nodejsinstall.md
│   ├── README.md
│   ├── gitbookcli.md
│   └── gitbookinstall.md
├── output
│   ├── README.md
│   ├── outfile.md
│   └── pdfandebook.md
└── publish
    ├── README.md
    └── gitpages.md
```

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，用于描述这一章的说明。

图书输出

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的网站：

本地预览是自动生成

当你编辑好gitbook文档之后，你可以使用gitbook的命令来进行本地预览。

```
$ gitbook serve ./{book_name}
```

gitbook会启动一个4000端口用于预览。

比如，通过 `gitbook serve` 来预览本文档：

```
$ gitbook serve gitbook-howtouse

Press CTRL+C to quit ...

Live reload server started on port: 35729
Starting build ...
Successfully built!

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

你可以你的浏览器中打开这个网址： http://localhost:4000

你会发现，在你编辑的gitbook项目的目录中会多一个 `_book` 目录，而这个目录中就是生成的本地预览的静态网站内容。

使用gitbook build参数

与直接预览生成的静态网站不一样的时，使用这个命令，你可以将内容输出到你想要的目录。

```
$ gitbook build --output=/tmp/gitbook
Starting build ...
Successfuly built !
$ ls /tmp/gitbook/
howtouse          search_index.json
book              imgs              output
gitbook           index.html        publish
```

无论哪种方式，你都可以将这个静态网站打包，发布到你想要发布的服务器上，或者直接将这个打包文件给阅读者。

输出PDF

输出为PDF文件，需要先安装gitbook pdf

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

> 如果在安装gitbook-pdf时，觉得下载phantomjs包太慢的话，你可以到phantomjs的官方网站上去下载。
>
> http://phantomjs.org/
>
> 这个包的安装方式，参考其官网的说明文档。

然后，用下面的命令就可以生成PDF文件了。

```
$ gitbook pdf {book_name}
```

如果，你已经在编写的gitbook当前目录，也可以使用相对路径。

```
$ gitbook pdf .
```

然后，你就会发现，你的目录中多了一个名为book.pdf的文件。

发布

可以使用Github Pages服务将我们写的gitbook发布到互联网上，前提是你已经了解了Git、Github及Github Pages的使用。

发布到Github Pages

将静态网站直接发布到Github Pages

可以将编写好的.md文件通过Gitbook处理成静态网站，然后发布到Github Pages上。

我的Github Pages是使用Octopress来进行构建的，于是就写了一个简单的脚本，将生成好的Gitbook直接同步到我的octopress项目中。

```
#!/bin/sh

Usage(){
    echo "welcome use front-end release script
    ----------------------------------------
    use it require input your deploy target
           gook luck!
    author:
        wolf.deng@gmail.com
    ----------------------------------------
    Usage:

    # 发布github.io
    ./deploy.sh github.io
    "
}

die( ){
    echo
    echo "$*"
    Usage
    echo
    exit 1
}

git pull

# get real path
cd `echo ${0%/*}`
abspath=`pwd`

#清除之前生成的文件
rm -rf $abspath/build

TARGET=$1
PROJECT='gitbook-howtouse'


sync( ){
    case $* in
        "github.io" )
            echo "sync $PROJECT gitbook website to $TARGET"
```

```
            GITHUB_PROJECT="~/workspace/github/tonydeng.github.i
            Sync="rsync -avu --delete --exclude '*.sh' --exclude
            echo $Sync
            eval $Sync

            cd $GITHUB_PROJECT
            rake generate
            rake deploy
        ;;

    esac
}

build(){
    echo "build $PROJECT document"

    OUTPUT="./build/$PROJECT"

    gitbook init

    rm  -rf $OUTPUT
    gitbook build . --output=$OUTPUT
}

blog(){
    echo "build & sync $PROJECT to github.io"
    build
    sync $TARGET
}

# 判断执行参数，调用指定方法
case $TARGET in
    github.io )
       blog
        ;;
    * )
        die "parameters is no reght!"
        ;;
esac
```

这样就可以直接调用脚本来将写好的Gitbook发布到我的Github Pages服务了。

这样其他人就可以通过下面的链接来查看这本Gitbook使用入门了。

> http://tonydeng.github.io/gitbook-zh/gitbook-howtouse/

使用项目的Pages服务

除了上面的直接发布静态文件到Github Pages的方法以外，还可以使用一个单独的项目的Github Pages功能。

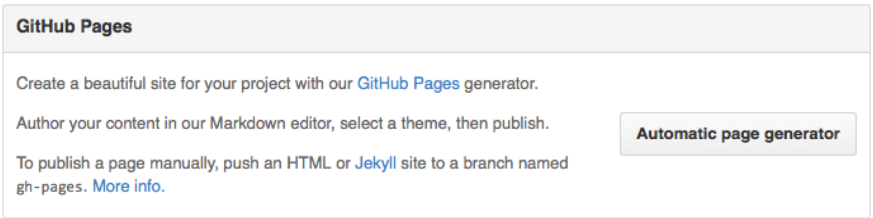
创建仓库与分支

1. 登陆到Github，创建一个新的仓库，名称我们就命名为book，这样我就得到一个book仓库。
2. 克隆仓库到本地： git clone git@github.com:/USER_NAME/book.git
3. 创建一个新分支： git checkout -b gh-pages，注意，分支名必须为gh-pages。
4. 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，我们已经创建了gh-pages分支了，有了这个分支，Github会自动为你分配一个网址。

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

你可以在项目页面右下角setting中看到：



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：

1. 切换出master分支目录。我们需要将gh-pages分支内容存放在另外一个目录中
2. 克隆gh-pages分支： git clone -b gh-pages git@github.com:USER_NAME/book.git book-end。这步我们只是克隆了gh-pages分支，并存放在一个新的目录book-end。
3. Copy静态网站到book-end目录中
4. Push到仓库

然后，等十来分钟后，你就可以访问到你的在线图书了。以后，只要你每次修改之后，将生成静态网站Copy到book-end目录，然后Push一下就OK了。

发布到公司文档服务器

我写了一个简单地脚本，基本原理如下：

```
#!/bin/sh

git pull

# get real path
cd `echo ${0%/*}`
abspath=`pwd`

#清除之前生成的文件
rm -rf $abspath/build


PROJECT='gitbook-howtouse'

sync(){
    echo "sync $PROJECT document to gitbook server"
    SyncServer="rsync://doc.dq.in/rsync"
    Sync="rsync -avu --port=30873 --delete  --exclude '*.sh' --p
    echo $Sync
    eval $Sync
}

build(){
    echo "build $PROJECT document"

    OUTPUT="./build/$PROJECT"

    gitbook init

    rm  -rf $OUTPUT
    gitbook build . --output=$OUTPUT
    sync
}

build
```

这样就可以直接访问下面的地址来在公司的服务器上查看这本Gitbook使用入门了

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，以前使用过Sphinx，可以使用在线工具直接生成一个静态网站。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或者编辑。

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，如有不对的地方，还请多多指教。