

封面图



GenBook

使用Gitbook生成电子书

O'RELY?

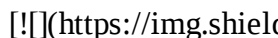

gluang

目录

简介	1.1
1-项目结构	1.2
2-查看帮助	1.3
3-Docker的部署和删除	1.4
4-生成PDF	1.5
5-构建本地web服务	1.6
6-Github Page	1.7
7-生成可执行二进制文件	1.8
封面图	1.9

GenBook

使用 **Docker** 搭建 **Gitbook**，用于生成电子书，静态网站，二进制可执行文件等。

 (LICENSE) 
(<https://img.shields.io/badge/docker%20build-passing-green>)

在线预览：gluang.github.io/GenBook

1. Option

- 生成PDF
- 构建本地 web 服务
- 生成 Github Page 静态资源文件
- 生成可执行二进制文件

2. 需要工具

- Docker
- make

3. 项目结构

```
.
├── .git
├── assets          // 存放生成的 pdf 和可执行二进制文件
├── content         // 存放 markdown 文件
├── docs            // 存放 web 静态资源文件
├── font            // 打印 pdf 所需字体
├── images          // 存放图片
├── .bookignore
├── .gitignore
├── Dockerfile      // Docker 构建文件
├── LICENSE
└── Makefile        // 指令
```

```
├─ README.md    // 项目简介
├─ SUMMARY.md   // 项目目录
├─ book.json     // pdf 配置文件
├─ cover.jpg     // pdf 封面图
├─ docs.go
├─ go.mod
├─ go.sum
└─ main.go
```

6 directories, 13 files

4. 查看帮助

```
$ make help
Usage:
  image           构建镜像
  container       构建容器
  html            生成静态文件
  pdf             生成 PDF
  serve           启动本地 web 服务，监听 4000 端口
  exec            生成可执行文件（需要使用 Golang 编译）
  pre-go          构建 Golang 编译容器
  go              使用容器编译生成二进制文件
  end-go          删除编译容器
  clean           删除命令 html pdf serve 生成的中间物
  rm-container    删除容器
  rm-image        删除镜像
  rm              删除容器和镜像
  help            打印命令帮助信息
```

5. Docker的部署和删除

- 构建容器：

```
# 创建镜像
$ make image
# 创建容器
$ make container
```

- 删除容器：

```
# 删除镜像
$ make rm-image
# 删除容器
$ make rm-container
# 删除镜像和容器
$ make rm
```

6. 可选服务

6.1. PDF

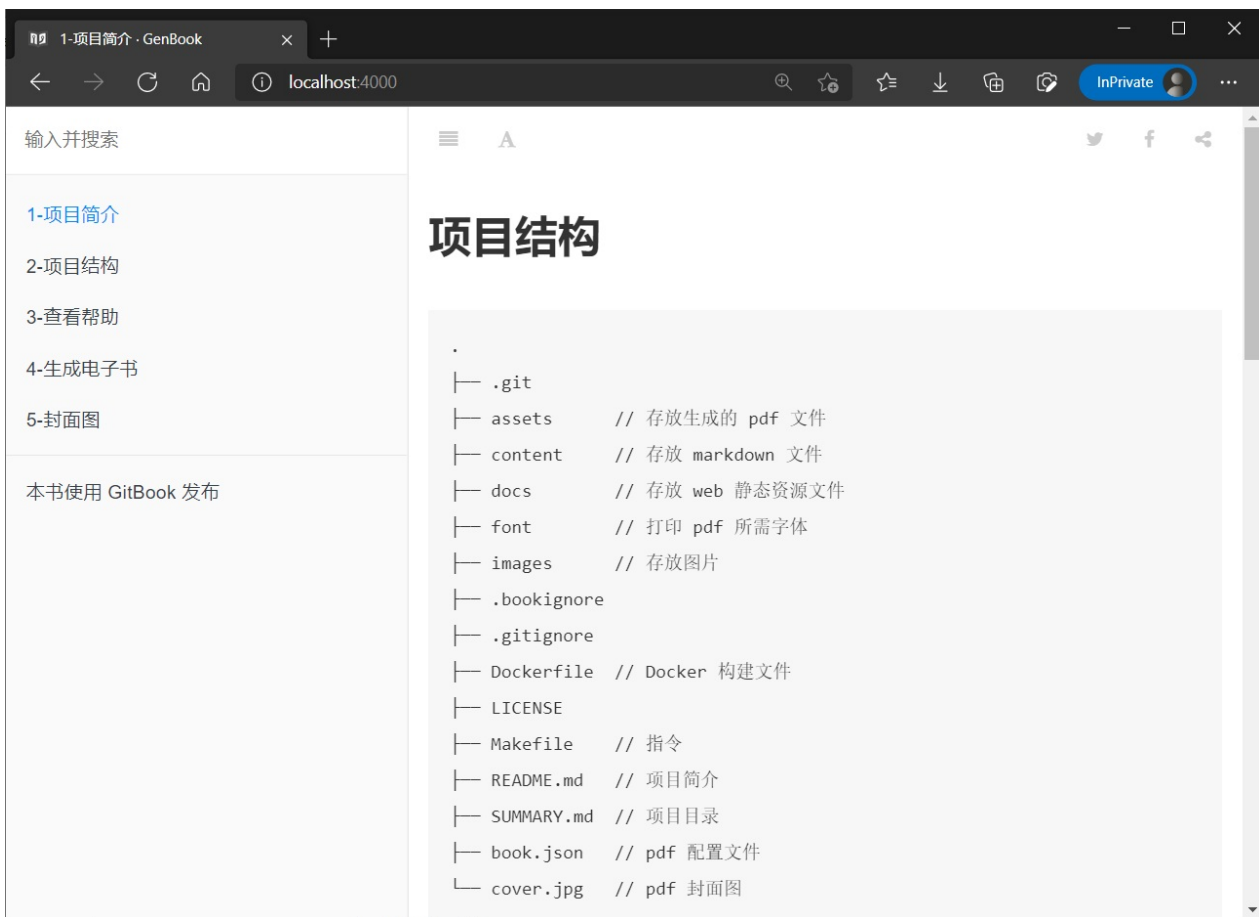
生成的 PDF 电子书位于 `assets` 目录下。

```
$ make pdf
# 如果有封面图，可使用 img=xxx 指定
$ make pdf img=images/cover.jpg
# 可以使用 name=xxx 指定生成的 pdf 文件名，默认为 book.pdf
$ make pdf name=mybook
```

6.2. 本地 Web 服务

```
$ make serve
```

浏览器访问 `http://localhost:4000`



6.3. Github Page

生成 web 静态资源文件：

```
$ make html
```

在 Github Page 设置时指定 `main` 分支下的 `docs` 路径即可。每次 push 后会自动更新。

6.4. 二进制可执行文件

本地编译

前提：需要 Golang 环境提供编译，且版本要求：`>= 1.16`。

默认编译为 linux 平台下的二进制文件，如需 windows 平台请使用 `os=windows` 进行指定。

```
# linux 平台
$ make exec
# windows 平台
$ make exec os=windows
```

Docker编译

- 构建编译容器：

```
$ make pre-go
```

- 编译生成二进制文件：

```
# linux 平台
$ make go
# windows 平台
$ make go os=windows
```

- 删除容器：

```
$ make end-go
```

程序使用

例如：在 linux 平台下

- 查看版本等信息

```
$ ./assets/exec-v0.0.1-linux-x86_64 -v
```

- 启动本地 web 服务，程序默认监听 12300 端口：

```
$ ./assets/exec-v0.0.1-linux-x86_64
```

- 也可手动指定监听端口：

```
$ ./assets/exec-v0.0.1-linux-x86_64 -p 12300
```

浏览器访问: `http://localhost:12300`

7. TODO

- [x] 将静态资源文件打包，构建可执行文件
- [x] Dockerfile 时区
- [x] 部署 Github Page
- [] \LaTeX 的支持
- [x] 添加 go lang 编译环境镜像

项目结构

```
.
├── .git
├── assets          // 存放生成的 pdf 文件
├── content         // 存放 markdown 文件
├── docs           // 存放 web 静态资源文件
├── font           // 打印 pdf 所需字体
├── images          // 存放图片
├── .bookignore
├── .gitignore
├── Dockerfile     // Docker 构建文件
├── LICENSE
├── Makefile       // 指令
├── README.md      // 项目简介
├── SUMMARY.md     // 项目目录
├── book.json      // pdf 配置文件
├── cover.jpg      // pdf 封面图
├── docs.go
├── go.mod
├── go.sum
└── main.go
```

6 directories, 13 files

查看帮助

```
$ make help
Usage:
  image          构建镜像
  container      构建容器
  html           生成静态文件
  pdf            生成 PDF
  serve          启动本地 web 服务，监听 4000 端口
  exec           生成可执行文件（需要使用 Golang 编译）
  pre-go         构建 Golang 编译容器
  go             使用容器编译生成二进制文件
  end-go         删除编译容器
  clean          删除命令 html pdf serve 生成的中间物
  rm-container   删除容器
  rm-image       删除镜像
  rm             删除容器和镜像
  help          打印命令帮助信息
```

Docker的部署和删除

- 构建容器：

```
# 创建镜像
$ make image
# 创建容器
$ make container
```

- 删除容器：

```
# 删除镜像
$ make rm-image
# 删除容器
$ make rm-container
# 删除镜像和容器
$ make rm
```

PDF

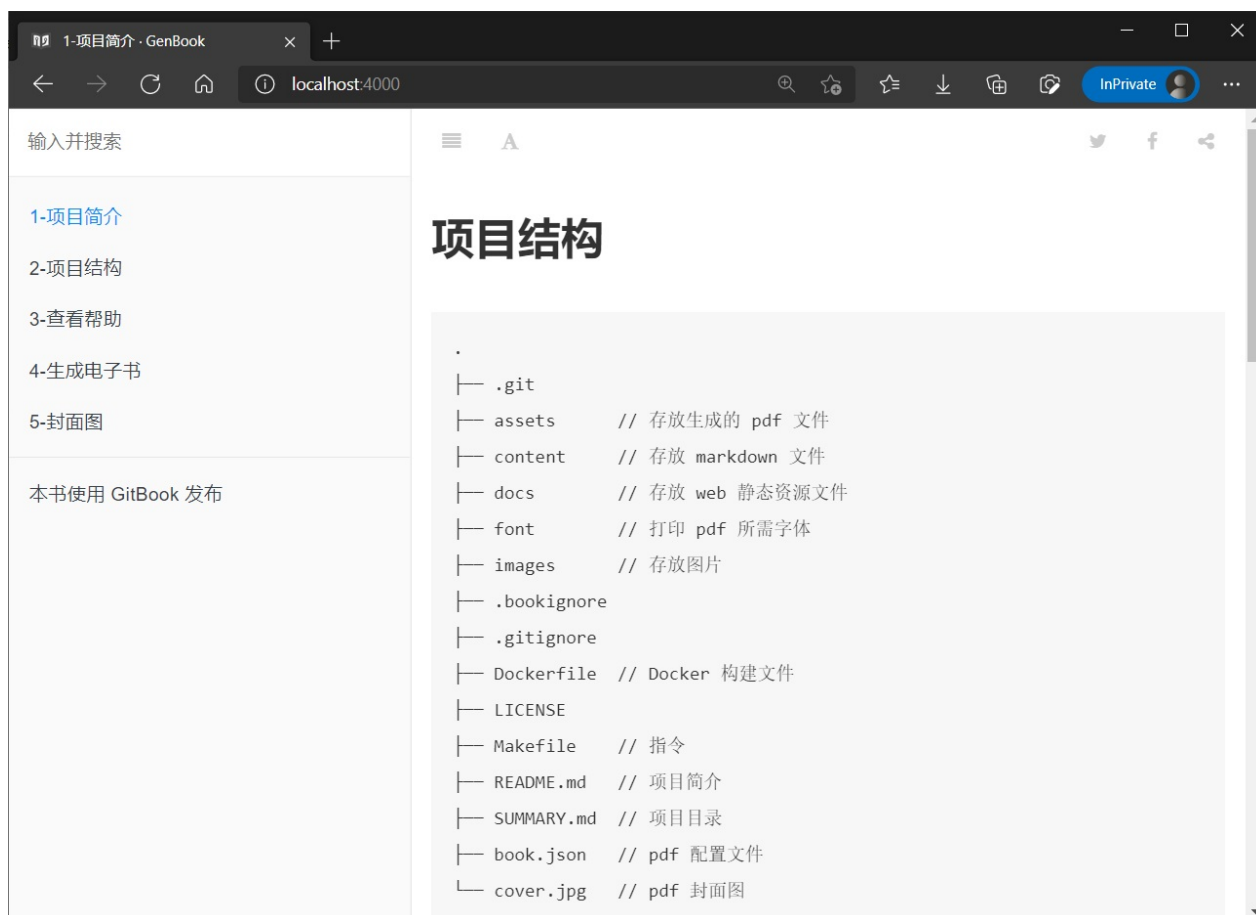
生成的 PDF 电子书位于 `assets` 目录下。

```
$ make pdf
# 如果有封面图，可使用 img=xxx 指定
$ make pdf img=images/cover.jpg
# 可以使用 name=xxx 指定生成的 pdf 文件名，默认为 book.pdf
$ make pdf name=mybook
```

构建本地 web 服务

```
$ make serve
```

浏览器访问 `http://localhost:4000`



Github Page

生成 web 静态资源文件：

```
$ make html
```

在 Github Page 设置时指定 `main` 分支下的 `docs` 路径即可。每次 push 后会自动更新。

本地编译

前提：需要 Golang 环境提供编译，且版本要求：`>= 1.16`。

默认编译为 linux 平台下的二进制文件，如需 windows 平台请使用 `os=windows` 进行指定。

```
# linux 平台
$ make exec
# windows 平台
$ make exec os=windows
```

Docker编译

- 构建编译容器：

```
$ make pre-go
```

- 编译生成二进制文件：

```
# linux 平台
$ make go
# windows 平台
$ make go os=windows
```

- 删除容器：

```
$ make end-go
```

程序使用

例如：在 linux 平台下

- 查看版本等信息

```
$ ./assets/exec-v0.0.1-linux-x86_64 -v
```

- 启动本地 web 服务，程序默认监听 12300 端口：

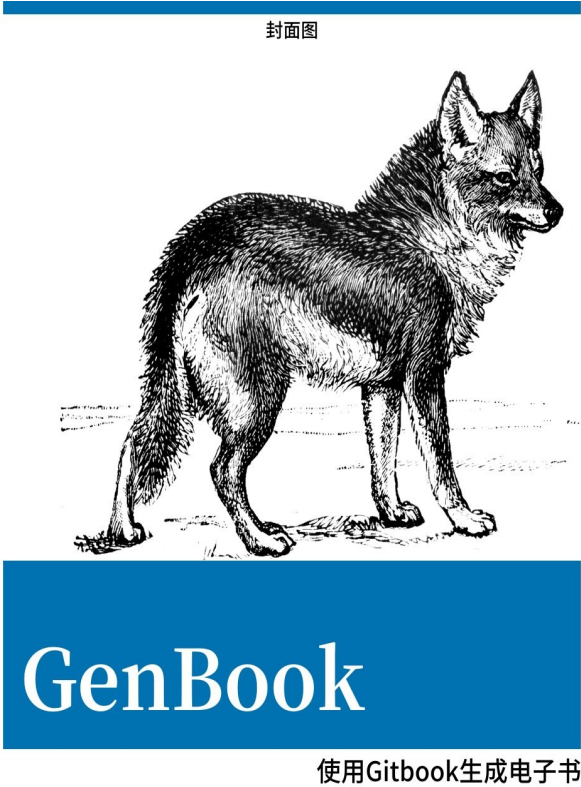
```
$ ./assets/exec-v0.0.1-linux-x86_64
```

- 也可手动指定监听端口：

```
$ ./assets/exec-v0.0.1-linux-x86_64 -p 12300
```

浏览器访问： `http://localhost:12300`

封面图



O'RLY?

gluang

来源：[O'RLY 动物书封面](#)