

---

# 目录

关于 Honkit	1.1
项目示例	1.2
Command Parser	1.2.1
Genotype Blocking Compressor (GBC)	1.2.2
配置环境	1.3
下载及使用 Typora	1.3.1
安装并启动 Docker	1.3.2
Windows	1.3.2.1
Macos	1.3.2.2
Ubuntu	1.3.2.3
通过 Dockerfile 构建 Honkit 服务镜像	1.3.3
使用 Honkit	1.3.4
Windows	1.3.4.1
初始化环境	1.3.4.1.1
搭建服务进行本地测试	1.3.4.1.2
构建网页文件	1.3.4.1.3
创建 PDF 文件	1.3.4.1.4
Macos 或 Linux	1.3.4.2
初始化环境	1.3.4.2.1
搭建服务进行本地测试	1.3.4.2.2
构建网页文件	1.3.4.2.3
创建 PDF 文件	1.3.4.2.4
编写文档	1.4
配置资源	1.4.1
修改网页基本信息	1.4.1.1
修改插件信息	1.4.1.2
多语言使用不同的插件配置	1.4.1.3
设置页面密码 (password-pro 插件)	1.4.1.4
扩展高亮语法使用说明	1.4.1.5
编写多语言/多版本列表	1.4.2
制作导航文件	1.4.3
制作子页面文件	1.4.4
制作 PDF 文件	1.4.5
搭建 Web 服务器	1.5

---

在 Mac 中使用 apache 服务器	1.5.1
使用 Docker 搭建 apache 服务器	1.5.2

---

# 关于 Honkit

honkit 是一个基于 Node.js、使用 Markdown 构建静态页面的命令行工具，通常被用于制作静态博客、软件/程序说明文档等。Honkit 是 Gitbook 的分支之一，兼容 Gitbook 的插件。我们推荐使用 Docker 搭建 Honkit 环境，使用 Typora 进行文档编写。

Honkit/Gitbook 指南: <https://snowdreams1006.github.io/markdown/>

Docker 指南: <https://tsejx.github.io/devops-guidebook/deploy/docker/overview>

## 项目示例

### Command Parser

文档预览: <http://pmglab.top/commandParser/>

代码仓库: <https://github.com/Zhangliubin/commandParser-1.1>

### Genotype Blocking Compressor (GBC)

文档预览: <http://pmglab.top/gbc/>

代码仓库: <https://github.com/Zhangliubin/GBC>

## 配置环境

## 下载及使用 Typora

类型	地址
MacOs (免费版)	<a href="https://download.typora.io/mac/Typora-0.11.18.dmg">https://download.typora.io/mac/Typora-0.11.18.dmg</a>
Windows x64 (免费版)	<a href="https://download.typora.io/windows/typora-update-x64-1117.exe">https://download.typora.io/windows/typora-update-x64-1117.exe</a>
Windows x86 (免费版)	<a href="https://download.typora.io/windows/typora-update-ia32-1117.exe">https://download.typora.io/windows/typora-update-ia32-1117.exe</a>
Linux (免费版)	<a href="https://download.typora.io/linux/typora_0.11.18_amd64.deb">https://download.typora.io/linux/typora_0.11.18_amd64.deb</a>
最新版	<a href="https://typoraio.cn">https://typoraio.cn</a>

破解教程: <https://github.com/taozhiyu/TyProAction/blob/main/README.zh.md>

Typora 语法教程: <https://support.typoraio.cn/zh/Markdown-Reference/>

[!WARNING|label:Typora 支持广泛的 html 语法和 markdown 语法]

Honkit 不支持 Markdown 中的 `==内容==` 高亮语法。

# 安装并启动 Docker

## Windows

**Step1:** 在 控制面板 -> 所有控制面板项 -> 程序和功能 -> 启用或关闭 Windows 功能 中开启 Hyper-V;

**Step2:** 前往 <https://www.docker.com/get-started> 下载、安装桌面版 Docker Desktop;

**Step3:** 前往 <https://docs.microsoft.com/zh-cn/windows/wsl/install-manual#step-4---download-the-linux-kernel-update-package> 下载内核更新包。

[!NOTE|label:安装完成后需要重启设备]

## MacOS

Macos 系统前往 <https://www.docker.com/get-started/> 下载安装，即可直接使用。

还可以使用 homebrew 安装:

```
# 安装 homebrew
/bin/zsh -c "$(curl -fsSL https://gitee.com/cunkai/HomebrewCN/raw/master/Homebrew.sh)"

# 使用 homebrew 安装 docker
brew install --cask docker
```

## Ubuntu

使用国内镜像进行安装:

```
# install curl
apt-get update
apt-get install curl -y

# use aliyun image to auto-install docker
curl -fsSL https://get.docker.com | bash -s docker --mirror Aliyun
```

启动:

```
sudo systemctl enable docker
sudo systemctl start docker
```

默认情况下, docker 命令会使用 Unix socket 与 Docker 引擎通讯。而只有 root 用户和 docker 组的用户才可以访问 Docker 引擎的 Unix socket。出于安全考虑, 一般 Linux 系统上不会直接使用 root 用户。因此, 更好地做法是将需要使用 docker 的用户加入 docker 用户组。建立 docker 组:

```
sudo groupadd docker
```

将当前用户加入 docker 组:

```
sudo usermod -aG docker $USER
```

退出当前终端并重新登录。

## 通过 Dockerfile 构建 Honkit 服务镜像

从 github 上下载资源：

```
# 下载 honkit-docker
git clone https://github.com/Zhangliubin/honkit-docker.git honkit-docker

# 进入下载的资源文件夹
cd honkit-docker
```

[!NOTE|label:下载 git]

下载 Git 工具，请参考：[安装 Git](#)

honkit-docker 的目录结构如下：

+ - LICENSE	开源许可证
+ - README.md	本文档 web 的主页内容
+ - SUMMARY.md	本文档 web 的导航信息
+ - book.json	本文档 web 的配置信息
+ - book.pdf	本文档的 pdf 版，使用 honkit pdf 导出
+ - docker	docker 镜像
+ - multi-lang	多语言环境模版
+ - single-lang	单语言环境模版

接着，我们通过 dockerfile 安装 honkit 服务：

```
# 构建名为 honkit 的镜像（需要挂上梯子进行下载）
docker build -t honkit -f docker/Dockerfile docker/
```

[!NOTE|label:如果无法构建该镜像(通常是网络连接问题)，也可以选择下载打包好的镜像]

- 实验室内网: <http://192.168.30.2/download/honkit.tar.gz>
- 中山大学校内网: <http://hpc.snplife.com/share/honkit.tar.gz>
- 校外使用百度云: [https://pan.baidu.com/s/1MTPJEa67g1YkZ\\_6S88gf6w](https://pan.baidu.com/s/1MTPJEa67g1YkZ_6S88gf6w) 提取码: 5b83

下载完成后，建议将该文件放在 honkit-docker/docker 目录下，使用以下语句加载镜像：

```
# 加载打包好的镜像，自动命名为 honkit:latest
docker load -i docker/honkit.tar.gz
```

## 使用 Honkit

以下操作请在项目文档文件夹 (例如: /commandParser/docs) 中进行，路径中的 `.` 代表当前文件夹路径。

[!TIP|label:honkit-docker 实际上包含 3 个项目的资源]

- 一个完整项目的示例: honkit-docker
- 单语言环境模版: honkit-docker/single-lang
- 多语言环境模版: honkit-docker/multi-lang

因此, 这三个文件夹都支持下面的操作 2~4 (搭建服务、构建网页文件夹、创建 PDF)

## Windows

### 1. 初始化环境

在终端 (cmd) 中输入以下指令:

```
# 语法: honkit init [workspace]
# 默认: honkit init .
docker run -v %cd%:/honkit/ -w /honkit/ --rm -it honkit init .
```

随后 `[workspace]` 文件夹中出现 `SUMMARY.md` 和 `README.md` 文件。这两个文件是启动 **Honkit** 必备的文件。

[!TIP|label:从模版中构建初始环境]

Honkit 依赖大量插件进行配置、美化。我们建议从已经配置好的模版中进行修改:

- 单语言环境: honkit-docker/single-lang
- 多语言环境: honkit-docker/multi-lang

将对应语言环境内的文件复制到当前 docs 目录下。

### 2. 搭建服务进行本地测试

在终端中输入以下指令:

```
# 语法: honkit serve [workspace] [output_dir]
# 默认: honkit serve . ./_book
docker run -it --init -p 4000:4000 -v %cd%:/honkit/ -w /honkit/ --rm honkit serve . ./_book
```

等待服务构建完成后 (即出现 `Serving book on http://localhost:4000` ), 在浏览器中输入 `http://localhost:4000` 或 `http://127.0.0.1:4000/` 访问网站。

[!DANGER|label:请勿将 `[workspace]` 和 `[output_dir]` 设置为同一个值]

[!TIP|label:这个网页能否被其他人浏览?]

搭建好测试服务后, 本机通过 `http://127.0.0.1:4000/` 可以访问该页面, 同个局域网下的用户也可以通过 `http://局域网IP地址:4000` 访问该页面。对于外网环境, 可以使用 [花生壳域名代理服务](#) 映射到公网上供其他用户访问。

当然, 我们更建议使用专门的 Web 服务器进行部署访问, 见 [搭建 Web 服务器](#) 一节。

### 3. 构建网页文件

在终端中输入以下指令：

```
# 语法: honkit build [workspace] [output_dir]
# 默认: honkit build . ./_book
docker run -v %cd%:/honkit/ -w /honkit/ --rm -it honkit build . ./_book
```

该指令会在 `[workspace]` 目录下生成 `[output_dir]` 文件夹，该文件夹内的文件即为网页资源。

[!DANGER]label:请勿将 `[workspace]` 和 `[output_dir]` 设置为同一个值

## 4. 创建 PDF 文件

在终端中输入以下指令：

```
# 语法: honkit pdf [workspace] [output]
# 默认: honkit pdf . ./book
docker run -v %cd%:/honkit/ -w /honkit/ --rm -it honkit pdf . ./assets/book.pdf
```

在单语言环境下，该指令会导出文件 `[output]`；

在多语言环境下，该指令会导出文件 `[output]_[lang].pdf`（如果 `[output]` 包含了 `.pdf` 扩展名，则会去掉扩展名后重新生成）。

[!TIP]label:导出 PDF 时隐藏网页页脚信息

在导出 PDF 时建议将 `book.json` 文件中的 `pluginsConfig - pagefooter-freedom - hide` 设置为 `true`。

## Macos 或 Linux

### 1. 初始化环境

在终端中输入以下指令：

```
# 语法: honkit init [workspace]
# 默认: honkit init .
docker run -v `pwd`:`pwd` -w `pwd` --rm -it honkit init .
```

随后 `[workspace]` 文件夹中出现 `SUMMARY.md` 和 `README.md` 文件。这两个文件是启动 **Honkit** 必备的文件。

[!TIP]label:从模版中构建初始环境

Honkit 依赖大量插件进行配置、美化。我们建议从已经配置好的模版中进行修改：

- 单语言环境：honkit-docker/single-lang
- 多语言环境：honkit-docker/multi-lang

将对应语言环境内的文件复制到当前 docs 目录下。

### 2. 搭建服务进行本地测试

在终端中输入以下指令：

```
# 语法: honkit serve [workspace] [output_dir]
# 默认: honkit serve . ./_book
docker run -it --init -p 4000:4000 -v `pwd`:`pwd` -w `pwd` --rm honkit serve . ./_book
```

等待服务构建完成后 (即出现 `Serving book on http://localhost:4000` ), 在浏览器中输入

`http://localhost:4000` 或 `http://127.0.0.1:4000/` 访问网站。

[!DANGER|label:请勿将 `[workspace]` 和 `[output_dir]` 设置为同一个值]

[!NOTE|label:端口占用时如何处理?]

本机端口被占用时, 请尝试将 `4000:4000` 更换为 `5000:4000` 等 ( 本机端口:服务端口 )。

```
docker: Error response from daemon: Ports are not available: exposing port TCP 0.0.0.0:4000 -> 0.0.0.0:0: listen tcp 0.0.0.0:4000: bind: address already in use.
ERRO[0000] error waiting for container: context canceled
```

[!TIP|label:这个网页能否被其他人浏览?]

搭建好测试服务后, 本机通过 `http://127.0.0.1:4000/` 可以访问该页面, 同个局域网下的用户也可以通过 `http://局域网IP地址:4000` 访问该页面。对于外网环境, 可以使用 [花生壳域名代理服务](#) 映射到公网上供其他用户访问。

### 3. 构建网页文件

在终端中输入以下指令：

```
# 语法: honkit build [workspace] [output_dir]
# 默认: honkit build . ./_book
docker run -v `pwd`:`pwd` -w `pwd` --rm -it honkit build . ./_book
```

该指令会在 `[workspace]` 目录下生成 `[output_dir]` 文件夹, 该文件夹内的文件即为网页资源。

[!DANGER|label:请勿将 `[workspace]` 和 `[output_dir]` 设置为同一个值]

### 4. 创建 PDF 文件

在终端中输入以下指令：

```
# 语法: honkit pdf [workspace] [output]
# 默认: honkit pdf . ./book
docker run -v `pwd`:`pwd` -w `pwd` --rm -it honkit pdf . ./assets/book.pdf
```

在单语言环境下, 该指令会导出文件 `[output]` ；

在多语言环境下, 该指令会导出文件 `[output]_[lang].pdf` (如果 `[output]` 包含了 `.pdf` 扩展名, 则会去掉扩展名后重新生成)。

[!TIP|label:导出 PDF 时隐藏网页页脚信息]



在导出 PDF 时建议将 `book.json` 文件中的 `pluginsConfig - pagefooter-freedom - hide` 设置为 `true`。

# 编写文档

## 1. 配置资源

### 1.1 修改网页基本信息

文档配置文件为文档根目录 `book.json`，将字段修改为本站点信息 (修改 `title`，`description`，`author`，`language`)。

其中，若网页主体语言为英语，可将 `language` 修改为 `en`。

### 1.2 修改插件信息

在配置文件 `plugins` 处填写导入的插件信息，在 `pluginsConfig` 处填写插件的配置信息。默认导入的插件列表如下：

插件名	描述
<code>favicon-absolute</code>	自定义网站 logo
<code>edit-link</code>	主页面左上角“编辑本页”按钮，可以跳转到 Github 仓库链接
<code>search-plus</code>	加强版搜索引擎，支持多语言搜索
<code>expandable-chapters-smal</code>	导航栏可折叠
<code>back-to-top-button</code>	页面内“回到顶部”按钮
<code>page-toc-button</code>	页面内导航条
<code>katex</code>	数学公式支持
<code>hide-element</code>	隐藏页面内的特定元素
<code>prism</code>	代码块配色
<code>prism-themes</code>	代码块配色主题
<code>github-buttons</code>	主页面右上角添加“Github”链接及图标
<code>summary-logo</code>	导航栏上面添加 Logo 及可跳转链接
<code>code</code>	代码可复制、显示行号
<code>todo</code>	使得“任务列表”样式不可编辑
<code>flexible-alerts</code>	扩展高亮语法
<code>theme-comscore</code>	页面内标题彩色
<code>language-picker</code>	主页面左上角多语言切换，“grid-columns”用于控制每行显示的语言个数
<code>pagefooter-freedom</code>	页脚自定义
<code>get-pdf</code>	主页面左上角显示下载 PDF

password-pro

为页面添加密码

根据自己的需求增删插件。插件配置信息需要修改：

- 代码仓库地址： `edit-link` 的 `base` , `github-buttons` 的 `buttons-user` 和 `buttons-repo`
- 版权信息： `pagefooter-freedom` 的 `copyright`
  - 在多语言环境下，还需要修改子语言文件夹中的 `book.json` 文件
- 下载 PDF： `get-pdf` 的 `base`
- 网页导航 logo： `summary-logo` 的 `url` 和 `link`
- 网页密码： `password-pro`

请注意，涉及路径信息的参数尽量保持格式一致 (相对路径、绝对路径、网页路径)。为了节省带宽资源，本页面的所有下载链接 (PDF、honkit-docker 等) 都托管在 Github 或公共平台。对于较小的服务资源，也可以直接存放在 Web 服务器中。

### 1.3 多语言使用不同的插件配置

`multi-lang` 文件夹下，除了根路径的 `book.json` 外，在每个语言的文件夹内也可以配置 `book.json`，用于规定在该版本下的插件配置。

[!TIP]label:什么情况下会用到多语言-多插件配置?]

- 多语言环境下在每个语言环境中使用统一的版权声明 (例如模版文件的多语言环境)；
- 不同的版本中为子页面添加密码、独立版权信息等。

### 1.4 设置页面密码 (password-pro 插件)

使用 `password-pro` 插件可以设置简易的网页密码。本插件有 4 个全局参数：

```
"password-pro": {
  "password": "123456",
  "tip": "请输入该页面的访问密码:",
  "errorTip": "密码错误, 请重新输入:",
  "reject": "当前页面拒绝访问: 身份验证失败."
},
```

"password" 表示为所有页面都添加密码。当传入的字符串为空字符串或不传入时表示不加密。

为指定页面添加密码：

```
"password-pro": {
  "README.md": "",
  "password": "20220611",
  "download.md": {
    "password": "12306",
    "tip": "下载资源需要验证密码:"
  },
  "command-line-interface.md": "complex12306",
  "tip": "请输入当前页面的访问密码:",
  "errorTip": "密码错误, 请重新输入:"
}
```

该语句表示 `README.md` 不加密，全局密码为 `20220611`，`download.md` 密码为 `12306`，`command-line-interface.md` 密码为 `complex12306`。

即格式有两种:

```
"<page>": "<password>"
```

和

```
"<page>": {
  "password": "<password>",
  "tip": "<tip>",
  "errorTip": "<errorTip>",
  "reject": "<reject>"
},
```

在第二种格式中，缺少的字段将使用全局参数替代。在同一次访问中，只要单个页面正确输入了密码。则下次访问该页面也不需要密码。

## 1.5 扩展高亮语法使用说明

导入该插件时，支持使用彩色框提示。目前有 6 种主要样式:

- NOTE 样式

[!NOTE] 包括 NOTE、COMMENT、TIP、WARNING、DANGER

- COMMENT 样式

[!COMMENT]

COMMENT 基于 NOTE 设置，但图标不一致

- TIP 样式

[!TIP|label:请联系我] 修改标签内容，使用 [!默认标签名|label:新标签名]

例: [!TIP|label:请联系我]

- UPDATE 样式

[!UPDATE|label:2022/06/10] UPDATE 基于 TIP 设置，但图标不一致

- WARNING 样式

[!WARNING|style:callout] 修改提示框为精简提示框，使用 [!默认标签名|style:callout]

例: [!WARNING|style:callout]

- DANGER 样式

[!DANGER|label:修改标签内容|style:callout]

例: [!DANGER|label:修改标签内容|style:callout]

# 上述样式对应的 markdown 代码

```

- NOTE 样式

> [!NOTE]
> 包括 NOTE、COMMENT、TIP、WARNING、DANGER

- COMMENT 样式

> [!COMMENT]
>
> COMMENT 基于 NOTE 设置，但图标不一致

- TIP 样式

> [!TIP|label:请联系我]
> 修改标签内容，使用 `[!默认标签名|label:新标签名]`
>
> 例：`[!TIP|label:请联系我]`

- UPDATE 样式

> [!UPDATE|label:2022/06/10]
> UPDATE 基于 TIP 设置，但图标不一致

- WARNING 样式

> [!WARNING|style:callout]
> 修改提示框为精简提示框，使用 `[!默认标签名|style:callout]`
>
> 例：`[!WARNING|style:callout]`

- DANGER 样式

> [!DANGER|label:修改标签内容|style:callout]
>
> 例：`[!DANGER|label:修改标签内容|style:callout]`

```

## 2. 编写语言列表 (多语言)

在多语言环境下，文档根目录 `LANGS.md` 声明了包含的语言：

```

# Languages

- [🇨🇳 简体中文](zh/)
- [🇬🇧 English](en/)

```

对于更多语言的支持, 按照如上格式进行续写。 `_layouts/languages.html` 定义了在本文件中的第一个语言将作为站点的默认页面。

多语言实际上也可以用于制作面向不同用户群体的文档界面（如面向用户的文档和面向开发者的文档；面向普通用户的文档和面向管理员的文档）；也可以用于制作文档历史页面。例如：

```

# Languages

- [2022 年度工作日志](2022/)
- [2021 年度工作日志](2021/)
- [2020 年度工作日志](2020/)

```

[!TIP|label:修改文档历史页面的样式及页面自动跳转]

`_layouts/languages.html` 定义了该页面的样式。我们在第 23~24 行处添加了以下信息，以支持自动跳转：

```
<!-- 设置自动跳转到第一个 url 处 -->
<script type="text/javascript">if ( >= 1) {window.location.href="";}</script>
```

## 3. 制作导航文件 (SUMMARY.md)

在单语言环境下，`SUMMARY.md` 文件位于文档根目录下；在多语言环境下，`SUMMARY.md` 文件位于各个语言子文件夹的根目录下。

```
# Summary

### Part I

- [Section I](part1/README.md)
- [Sub Section I](part1/section1.md)
- [Sub Section II](part1/section2.md)

### Part II

- [Section I](part2/README.md)
- [Sub Section I](part2/section1.md)
- [Sub Section II](part2/section2.md)

### Part III

- [Section I](part3/README.md)
- [Sub Section I](part3/section1.md)
- [Sub Section II](part3/section2.md)
```

此文件用于制作网页左侧导航，一级标题不设地址；根据语法建议，二级标题应该以单独文件夹的 `README.md` 文件作为入口（此建议不是强制性的，实际上任意的 markdown 文件都可以作为入口文件）。

由于页面内部也会自动按照文档级别生成导航条，因此建议左侧导航条至多到 三级/四级。

标题可以通过以下方式设定链接：

- 跳转到某个 md 页面：[标题名](文件相对路径地址)；
- 跳转到某个 md 页面的指定锚点：[标题名](文件相对路径地址#锚点)，锚点需要在页面内使用 {#锚点} 进行标记。请注意，锚点字符中不能用空格、点。

## 4. 制作子页面文件

子页面文件采用常规的 Markdown 格式书写。通过导航目录或者页面内跳转的方式进行连接。

请注意，在单语言环境下，文档根目录的 `README.md` 文件必不可少，它是网页的入口文件；在多语言环境下，每个语言子文件夹的 `README.md` 文件必不可少（例如：`zh/README.md`）。

[!NOTE|label:为什么 README.md 必不可少?]

文档根目录或语言子文件夹根目录中的 README.md 是该网页的入口 (例如, 当网页资源挂载在 pmglab.top/honkit-docker 目录下时, 输入: pmglab.top/honkit-docker 或 pmglab.top/honkit-docker.html 将跳转到该 README.md 文件对应的网页资源文件)。

## 5. 制作 PDF 文件

要提供页面左上角“下载 PDF 文件”功能, 需要在本地预先导出。当使用:

```
# Windows
docker run -v %cd%:/honkit/ -w /honkit/ --rm -it honkit pdf ./assets/book.pdf

# MacOS 或 Linux
docker run -v `pwd`:`pwd` -w `pwd` --rm -it honkit pdf ./assets/book.pdf
```

导出 PDF 文件时, 单语言环境会生成 `./assets/book.pdf` 文件; 多语言环境则会生成 `./assets/book_en.pdf` 和 `./assets/book_zh.pdf` 等。

此时, 插件 `get-pdf` 只需要将 `base` 定位到 Web 服务器中的 `assets` 路径 (例如, 下面使用的是 Github 仓库), 便能够正确连接。

```
"get-pdf": {
  "base": "https://github.com/Zhangliubin/honkit-docker/tree/main/multi-lang/assets/",
  "prefix": "book",
  "label": ""
}
```

这里的 `prefix` 就是导出指令中输出文件的文件名。当修改了输出文件名时, 请同步修改 `book.json` 中的 `get-pdf` 的 `prefix` 属性字段。

[!TIP|label:制作 PDF 封面]

在 Word 中使用 A4 纸样式设计好封面, 并导出为 `cover.jpg` 到文档根目录下。

## 搭建 Web 服务器

构建网页数据后, 需要部署 Web 服务器以响应用户请求。Web 服务器一般也称为 http 服务器, 如 Apache、IIS、Nginx 等。此外, 还有存放和运行系统程序的应用服务器, 负责处理程序中的业务逻辑, 如 Tomcat、Weblogic、Jboss (现在大多数应用服务器也包含了 web 服务器的功能)。

本文介绍使用 apache 服务器搭建 Web 服务器的方式。

## 在 Mac 中使用 apache 服务器

### 启动服务

MacOS 中自带了 apache 服务器, 只需要输入以下指令即可启动:

```
sudo apachectl start
```

此时，在浏览器中输入：`http://127.0.0.1` 或 `http://127.0.0.1:80`，若出现 `It works!`，说明服务已经启动。默认情况下，该服务器资源存放在：`/Library/WebServer/Documents` 中，将 `_book` 的内容移动到该文件夹下即可实现 Web 访问。

[!TIP|label:添加所有者权限，以解决频繁要求输入密码的问题]

`/Library/WebServer/Documents` 文件夹的所有者是 `root`，这导致我们增删文件时都需要输入密码。一种解决方案是在文件夹鼠标右键 > 显示简介 > 共享与权限 中添加当前用户权限。此外，可以为该文件夹制作快捷方式、重命名，并放置到其他路径。

## 打开文件访问视图

许多网站除了可以显示 HTTP 网页 (站点服务器路径下的 `index.html` 文件) 外，还可以使用“下载模式”在该路径下显示文件的名称。打开文件访问视图需要修改配置文件：`/etc/apache2/httpd.conf` (通过 访达菜单栏 > 前往 > 前往文件夹打开 该文件)。搜索以下行字段：

```
Options FollowSymLinks Multiviews
```

将其修改为：

```
Options Indexes FollowSymLinks Multiviews
```

重启 apache 服务：

```
sudo apachectl restart
```

此时，目录列表就可以正常访问。

## 停止服务

停止 apache 服务，请输入：

```
sudo apachectl stop
```

## 使用 Docker 搭建 apache 服务器

对于 Linux 或 Windows 设备，我们建议使用 Docker 搭建 Apache 微服务。

从 Docker Hub 上拉取镜像：

```
docker pull httpd:alpine
```

将 `honkit-docker/_book` 的路径进行挂载：

```
# Windows
docker run -p 8080:80 --rm -v %cd%/_book:/usr/local/apache2/htdocs/ -d --name apache-server httpd:alpine

# MacOS 或 Linux
docker run -p 8080:80 --rm -v `pwd`/_book:/usr/local/apache2/htdocs/ -d --name apache-server httpd:alpine
```

此时，在浏览器中输入：`http://127.0.0.1:8080` 访问相应的 Web 资源。修改 `8080` 可以更改映射的端口。

[!TIP]label:同时挂载多个路径

同时挂载多个路径可以通过添加多个挂载点 `-v` 实现，以 MacOS 为例：

```
docker run --name apache-server -p 8080:80 --rm \
-v `pwd`/_book:/usr/local/apache2/htdocs/honkit-docker \
-v `pwd`/single-lang/_book:/usr/local/apache2/htdocs/single-lang \
-v `pwd`/multi-lang/_book:/usr/local/apache2/htdocs/multi-lang \
-d httpd:alpine
```

停止 apache 服务，请输入：

```
docker stop apache-server
```