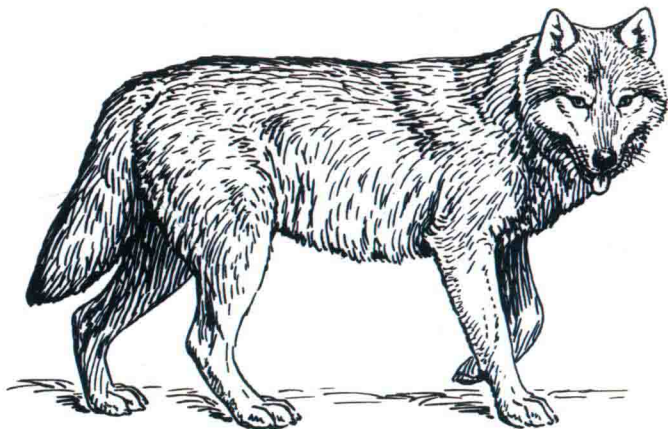


Flask开发团队成员撰写，Flask开发团队核心维护者高度评价并推荐

基于全新Flask技术版本，从基础知识到进阶实战，再到源码分析，提供完善的Flask学习路径



Flask Web开发实战

入门、进阶与原理解析

PYTHON WEB DEVELOPMENT WITH FLASK

李辉 著



机械工业出版社
China Machine Press



Flask Web开发实战

入门、进阶与原理解析

PYTHON WEB DEVELOPMENT WITH FLASK

李辉 著



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

Flask Web 开发实战：入门、进阶与原理解析 / 李辉著. —北京：机械工业出版社，2018.8
(Web 开发技术丛书)

ISBN 978-7-111-60659-8

I. F… II. 李… III. 软件工具 - 程序设计 IV. TP311.561

中国版本图书馆 CIP 数据核字 (2018) 第 183791 号

Flask Web 开发实战：入门、进阶与原理解析

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：李 艺 责任校对：李秋荣
印 刷：北京市荣盛彩色印刷有限公司 版 次：2018 年 9 月第 1 版第 1 次印刷
开 本：186mm×240mm 1/16 印 张：44
书 号：ISBN 978-7-111-60659-8 定 价：129.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：(010) 88379426 88361066
购书热线：(010) 68326294 88379649 68995259

投稿热线：(010) 88379604
读者信箱：hzit@hzbook.com

版权所有·侵权必究


封底无防伪标均为盗版

本书法律顾问：北京大成律师事务所 韩光 / 邹晓东

Flask 是目前最流行的 Python Web 框架之一。自 2010 年开源以来，Flask 受到了越来越多的 Python 开发者的喜欢，其受欢迎程度不输于 Django。截至 2018 年 6 月，它在 GitHub 上已有近 36 000 个 Star，2000 多位 Watcher，是目前 GitHub 中 Star 数最多的 Python Web 框架。




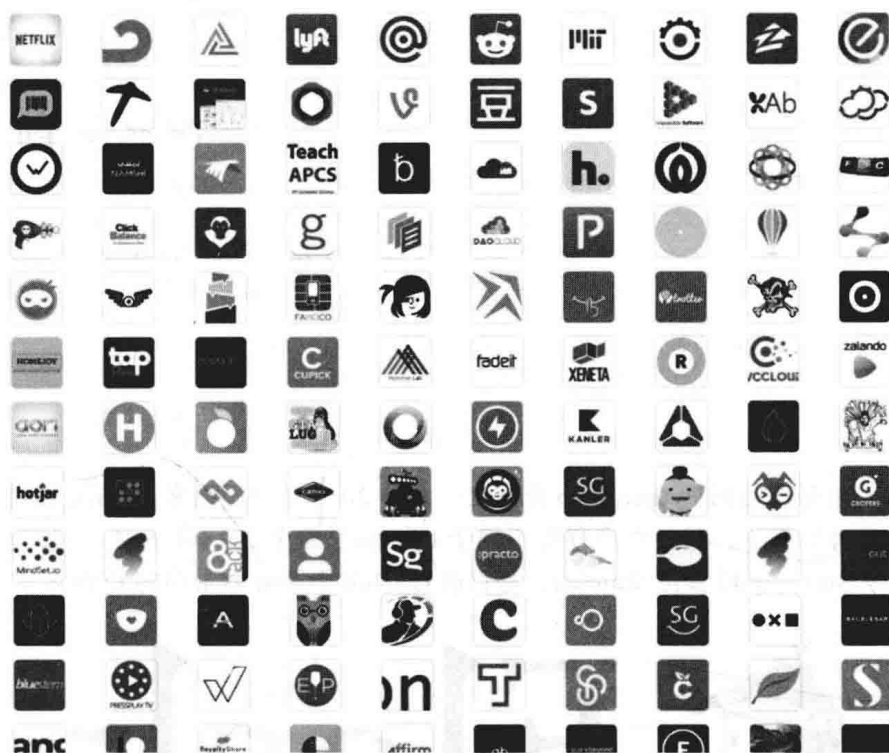
Flask 的 logo

 **附注** Flask 的图标虽然看起来很像辣椒，但其实它是角状的容器（powder horn）。

Flask 仅保留了 Web 框架的核心，其他的功能都交给扩展实现。如果没有合适的扩展，你甚至可以自己编写。Flask 不会替你做决定，也不会限制你的选择。它足够轻量，你可以只用 5 行就编写出一个最简单的 Web 程序，但并不简陋，它能够适应各类项目的开发。

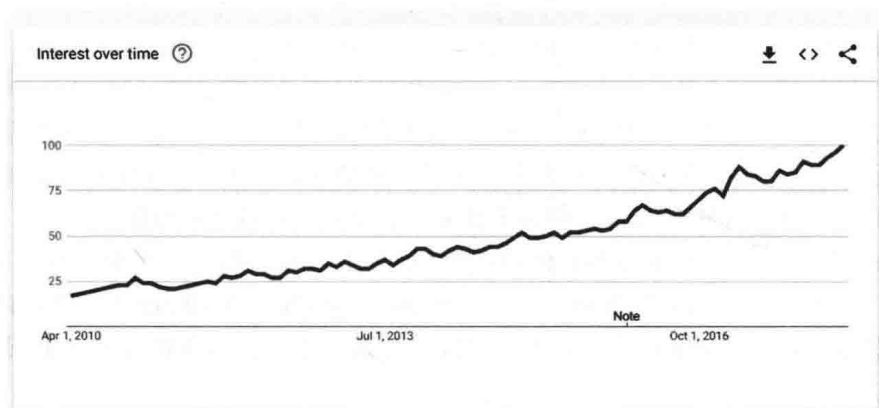
因为 Flask 的灵活性，越来越多的公司选择 Flask 作为 Web 框架，甚至开始从 Django 迁移到 Flask。使用 Flask 的公司在国外有 Netflix、Reddit、Twilio、Mailgun 等，在国内则有豆瓣、果壳、下厨房等，这说明 Flask 能经受大型项目的挑战，能够适应各种需求。下图列出了部分使用 Flask 的公司。

 **附注** 你可以在 StackShare 上查看完整的使用 Flask 的公司列表 (<https://stackshare.io/flask>)。



使用 Flask 的公司

在国内，越来越多的 Python 程序员开始关注和学习 Flask。对于国内的程序员来说，相关书籍仅有一两本，内容上也过于陈旧和单薄，希望本书可以填补这一空白。本书提供了学习 Flask 的完整路径，从基础内容到进阶实践，再到源码分析。同时也安排了丰富的示例程序，让读者可以通过亲自实践来更快地掌握 Flask 开发。

Flask 自 2010 年开源以来在 Google 上的搜索趋势[⊖]

[⊖] 参考来源：Google Trends (<https://trends.google.com/trends/explore?date=2010-04-01%202018-04-01&q=%2Fm%2F0dgs72v>)。

目标读者

在技术层面，本书适合所有 Python 程序员（了解 Python 即可）阅读，包括已经学习过其他 Python Web 框架（比如 Django）的读者和没有接触过 Web 框架的读者。

在难度水平层面，本书适合新手以及中级读者阅读。新手会在这里学到 Flask 的基础内容，并且通过丰富、完善的实例学习 Flask 开发的方方面面；中级读者则可以通过阅读和实践进阶内容来进一步提高 Flask 开发能力。

综上所述，本书主要适合以下几类读者：

- ❑ 了解 Python 基本语法，想要自己动手做网站的编程爱好者。
- ❑ 熟悉 Python，想要从事 Python Web 开发的后端工程师、运维工程师和爬虫工程师。
- ❑ 想要从 Django 等其他 Python Web 框架转向 Flask 的 Python 工程师。

本书主要特点

本书主要有三个显著的特点：

（1）内容全面

本书内容覆盖了 Flask Web 开发的完整路径：从基础知识的学习，到不同类型和复杂程度的程序的编写，再到代码的测试优化以及 Flask 源码分析；从基础的内容管理，到用户认证和权限管理，再到 Flask 与 JavaScript 的数据交互、Web API 的编写以及 WebSocket 的应用等。

（2）实践丰富

本书包含大量代码片段，并附带多个完整可运行的示例程序。在本书第一部分的第 2~6 章均分别提供一个示例程序；第二部分则会通过介绍 5 个比较完善的 Flask 项目来讲解各个方面的进阶知识；在第三部分还会通过一个真实的扩展来讲解 Flask 扩展开发。通过将各类知识融入实际的项目开发实践中，可以让你更直观地了解具体的代码实现，并且快速应用到实际开发中。

（3）内容最新

本书的另一个特点就是内容保证最新。书中的代码和示例程序都基于 Flask 最新发布的稳定版 1.0。书中涉及的其他 Python 包和前端框架（Bootstrap、Materialize 等）全部使用最新版本，并且对未来可能会有的变化会加以说明。这些特点可以保证书中的内容在一定时间内不会过时。对于其他书籍或教程中存在的关于 Flask 的误区，本书也会逐一纠正说明。

除了使用的工具保持最新，本书还引入了 Python 和 Flask 开发中的新变化，比如 Flask 的命令行系统、新的 Python 包管理工具（Pipenv）、新的包上传工具（twine）、新的 PyPI 站点（<https://pypi.org>）、在 PyPI 上使用 Markdown 格式的 README……

本书核心内容

本书由三部分组成，分别为基础篇、实战篇、进阶篇，共 16 章。本书章节经过精心设计，力求让读者可以循序渐进地掌握 Flask 开发的基础知识和技巧。

第一部分：基础篇。介绍 Flask 开发相关的基础知识。

- 第 1 章：搭建开发环境，编写一个最小的 Flask 程序并运行它，了解 Flask 基本知识。
- 第 2 章：介绍 Flask 与 HTTP 的交互方式以及相关的 Flask 功能。
- 第 3 章：介绍 Jinja2 模板的使用。
- 第 4 章：介绍 Web 表单的创建和表单数据的验证。
- 第 5 章：介绍在 Flask 程序中使用数据库进行 CRUD 操作。
- 第 6 章：介绍在 Flask 程序中发送电子邮件的几种方式。

第二部分：实战篇。通过几个示例程序来介绍 Flask 开发中各类功能的实现方法和技巧。

- 第 7 章：通过一个简单的留言板程序 SayHello 介绍 Web 开发的基本流程和基本的项目管理方式，对第一部分的基础知识进行简单回顾。
- 第 8 章：通过个人博客程序 Bluelog 介绍 CRUD 操作、用户认证、文章评论、管理后台等功能。
- 第 9 章：通过图片社交程序 Albumy 介绍用户注册和认证、用户权限管理、图片上传与处理、用户头像、复杂的数据库关系、复杂的数据库查询、全文搜索等内容。
- 第 10 章：通过待办事项程序 Todoism 介绍单页应用、国际化与本地化、Web API、OAuth 服务器端实现等内容。
- 第 11 章：通过聊天室程序 CatChat 介绍 Websocket 应用、OAuth 客户端实现（第三方登录）、Markdown 支持、代码语法高亮等内容。

第三部分：进阶篇。介绍 Flask 程序的部署流程，如测试、性能优化、部署上线；介绍 Flask 开发的进阶话题，如 Flask 扩展开发、Flask 源码与机制分析。

- 第 12 章：介绍 Flask 程序的自动化测试，包括单元测试和 UI 测试的编写、计算测试覆盖率和代码质量检查。
- 第 13 章：介绍对 Flask 程序进行性能优化的主要措施，包括函数与数据库查询的性能分析、缓存的使用、静态文件优化。
- 第 14 章：介绍部署 Flask 程序前的准备，以及部署到 Linux 服务器和云平台 Heroku、PythonAnywhere 的完整流程。
- 第 15 章：通过扩展 Flask-Share 来介绍编写 Flask 扩展的完整流程，从创建项目到上传到 PyPI。
- 第 16 章：介绍 Flask 的一些设计理念，包括底层 WSGI 的相关实现，并对各个主要功能点进行源码分析。

此外，书的最后还提供了附录 A，补充介绍一些 Flask 学习相关的资源。

阅读前的准备

在开始我们的 Flask 之旅前，还有一些准备工作要做。首先，你要有一台安装了 Python (<https://www.python.org/>) 的电脑，并且，你要了解 Python 的基础知识。



本书中所有示例程序的代码均通过了 Python 2.7 和 Python 3.6 的测试，建议你选用这两个版本。因为大多数 Python 包（包括 Flask）已经不再支持 Python 2.6 及以下版本，以及 Python 3.3 及以下版本，确保不要使用这些版本。另外，Python 官方社区将于 2020 年 1 月 1 日停止对 Python 2.x 的维护，这或许可以作为你选择 Python 版本时的考量之一。

其次，本书有大量操作需要在命令行（CLI，Command Line Interface）下进行，所以你要熟悉你所在操作系统下的命令行。书中会在涉及操作系统特定的命令时给出提示，Windows 系统给出的命令对应的是 CMD.exe，Linux 和 macOS 系统则对应的是 Bash。

最后，HTML、CSS、JavaScript 分别作为一个 Web 页面的结构层、表现层和行为层，是 Web 开发的基础，你需要对它们有基本的了解。任何一个 Web 程序都是由单个或多个 Web 页面，页面上包含的内容，以及按钮、表单等交互组件构成的。在本书中，我们会使用 Flask 操作 HTML 页面；为了让 HTML 页面更加美观，我们会使用 CSS 定义样式，为了简化编写样式的操作，我们会使用 CSS 框架，比如 Bootstrap（<http://getbootstrap.com/>）；为了让某些操作更加合理和方便，或为了给程序增加动画效果，我们会使用 JavaScript 来操作页面元素，为了简化编写 JavaScript 的工作，我们会使用 JavaScript 库 jQuery（<https://jquery.com/>）。



在 Web 开发中，大部分程序离不开 JavaScript，JavaScript 可以很方便、简洁地实现很多页面逻辑和功能。为了更多地介绍 Flask，本书将尽量避免使用过多的 JavaScript 代码。

如果你还不熟悉这些内容，那么可以通过下面的网站来快速入门：

- W3Schools (<https://www.w3schools.com>)。
- MDN Web 文档 (<https://developer.mozilla.org/docs/Web>)。
- Codecademy (<https://www.codecademy.com>)。

使用示例程序

示例程序均使用 Git 来管理程序版本，为了便于大家获取示例程序，代码均托管在 GitHub（<https://github.com/>）上。Git（<https://git-scm.com/>）是最流行的开源 VCS（Version Control System，版本控制系统），大多数项目都使用它来追踪文本文件（代码）的变化。Git 非常易于上手，如果你还不熟悉它，可以访问 Try Git（<https://try.github.io/>）来快速了解 Git。


你可以访问 Git 官网的下载页面（<https://git-scm.com/downloads>）了解不同操作系统下 Git 的安装方法，安装成功后即可使用它来获取示例程序。下面介绍了两种使用示例程序的方式。

1. 阅读示例程序

因为示例程序都托管在 GitHub 上，所以阅读示例程序最简单的方式是在浏览器中阅读。在对应的章节，我们会给出示例程序在 GitHub 上的仓库链接。

如果要在本地阅读，那么首先使用 `git clone` 命令把 GitHub 上的示例程序克隆（即复制）到本地，以本书的项目仓库为例：


```
$ git clone https://github.com/greyli/helloflask.git
```

 **提示** clone 命令后面的参数是远程 Git 仓库的 URL，最后的 “.git” 后缀可以省略。这里的 URL 中的传输协议使用了 http(s):// 协议，你也可以使用 git:// 协议，即 git://github.com/greyli/helloflask.git。

使用 ls (即 List) 命令 (Windows 下使用 dir 命令) 列出当前目录下的文件信息，你会看到当前目录中多了一个 helloflask 文件夹，这就是我们刚刚复制下来的项目仓库。下面使用 cd (即 change directory) 命令切换进这个文件夹：

```
$ cd helloflask
```

现在你可以使用你喜欢的文本编辑器打开项目文件夹并准备阅读了。建议使用轻量的文本编辑器来阅读示例代码，比如 Atom (<https://atom.io/>)、Sublime Text (<https://www.sublimetext.com/>) 或 Notepad++ (<https://notepad-plus-plus.org/>)。

在对应章节的开始处都会包含从 GitHub 复制程序、创建虚拟环境并运行程序的基本步骤，你可以一边阅读源码，一边实际尝试使用对应的程序功能。

示例程序根据章节内容设置了对应的标签，每个标签都对应了一个程序版本。你可以使用 git tag -n 命令查看当前项目仓库中包含的所有标签：


```
$ git tag -n
```

使用 git checkout 命令即可签出对应标签版本的代码，添加标签名作为参数，比如：

```
$ git checkout foo
```

在后面，书中会在每一次包含更改文件的章节提示应该签出的标签名。如果在执行新的签出命令之前，你对文件做了修改，那么需要使用 git reset 命令来撤销改动：

```
$ git reset --hard
```

 **注意** git reset 命令会删除本地修改，如果你希望修改示例程序源码并保存修改，可以参考后面的“改造示例程序部分”。

如果你想比较两个版本之间的变化，可以使用 git diff 命令，添加比较的两个标签作为参数，比如：

```
$ git diff foo bar
```

如果你想更直观地查看版本变化，可以使用下面的命令打开内置的 Git 浏览客户端：

```
$ gitk
```

除了内置的 Git 客户端，还有大量的第三方客户端可以使用，详情可以访问 <https://git-scm.com/downloads/guis> 查看。另外，你也可以访问 GitHub 的 Web 页面查看不同版本 (标签) 的变化，即查看某项目两个版本之间的变化可以访问 <https://github.com/用户名/仓库名/compare/标签 A... 标签 B>，比如对 foo 和 bar 标签进行比较可以访问 <https://github.com/greyli/helloflask/compare/foo...bar>。

最后，你可以定期使用 `git fetch` 命令来更新本地仓库：

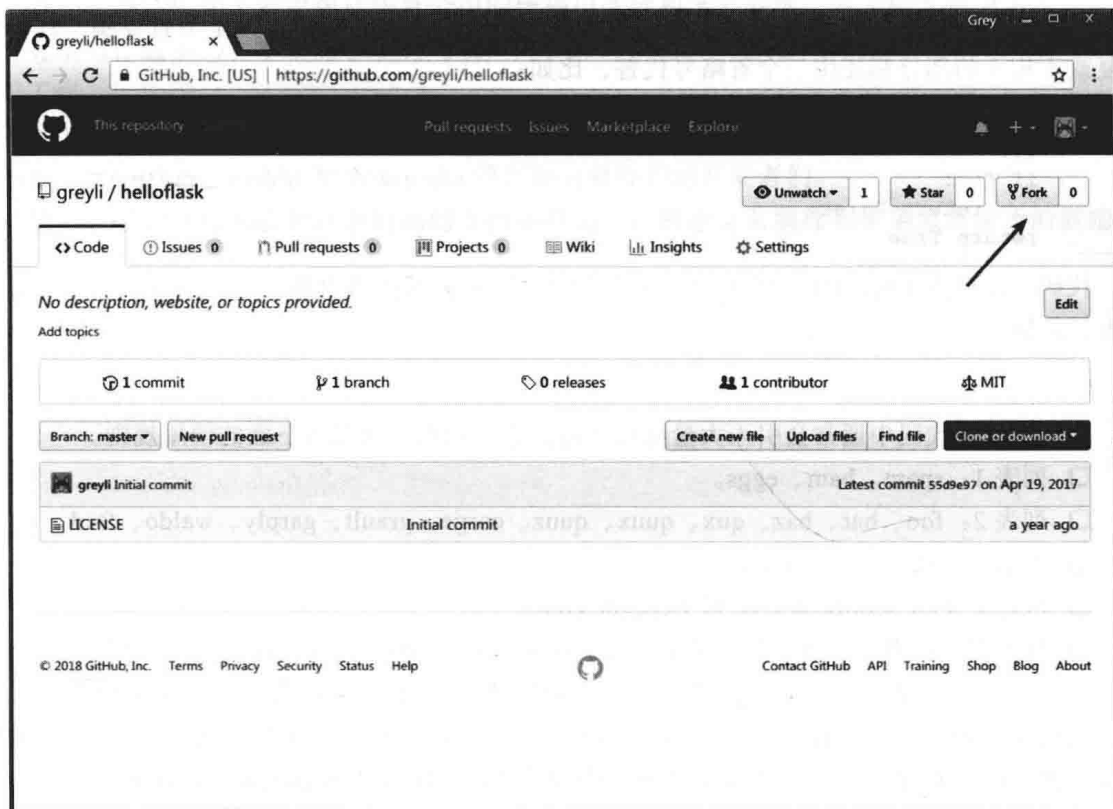
```
$ git fetch --all
$ git fetch --tags
$ git reset --hard origin/master
```

2. 改造示例程序

只看菜谱是不能学会烹饪的，自己动手编写代码才是学习 Flask 最有效的途径。你可以在阅读示例程序的同时编写自己的 Flask 程序，将书中介绍的内容和实际的示例程序代码作为参照。另外，你也可以创建一份示例程序的拷贝（派生，fork），这样你就可以自由地修改示例程序的源码，改造成你自己的示例程序。创建派生仓库的主要步骤如下：

1) 注册一个 GitHub 账号 (<https://github.com>)。

2) 访问示例程序的 GitHub 仓库页面（比如 <https://github.com/greyl/helloflask>），单击右上角的 Fork 按钮创建一个派生仓库，如下图所示。



创建派生仓库

3) 在本地使用 `git clone` 命令复制新创建的派生仓库，使用你的用户名构建 URL：

```
$ git clone https://github.com/你的用户名/helloflask.git
```

现在你可以在本地自由修改实例程序，并提交到你的 GitHub 账户的远程仓库中了。

排版约定

Windows 中的命令提示符为 “>”，而 Linux 或 macOS 中的命令提示符为 “\$”，本书中将统一使用美元符号（即 “\$”）作为命令提示符，比如：

```
$ cd hello
```

命令提示符为三个大于号（即 “>>>”）的表示 Python Shell 中输入的代码，比如：

```
>>> import os
```

“\$” 或 “>>>” 标记的文本下方没有命令提示符的文字表示输出的字符，不需要打出，比如：

```
$ cat hello.txt  
Hello, Flask!
```

为了节省篇幅，本书中的代码片段没有严格遵照 PEP8 的约定，比如类和函数之间的空行被缩减为 1 行。另外，出现过的导入语句和无关的代码块会被省略掉。为了节省篇幅，代码中重复或不相干的部分都使用三个省略号代替，比如：

```
def do_something():  
    ...  
    if foo:  
        return False  
    return True
```

代码、命令或 URL 中有时会使用 “<” 和 “>” 来标识演示内容，在实际输入中并不需要写出，比如：

```
https://github.com/<你的用户名>
```

因为在示例代码中通常会引入大量随机字符，这些随机字符包含下面的使用规则。

- ❑ 列表 1: spam、ham、eggs。
- ❑ 列表 2: foo、bar、baz、qux、quux、quuz、corge、grault、garpily、waldo、fred。
- ❑ 人名会使用 Grey Li 或 grey。
- ❑ 网站会使用 helloflask.com 或 example.com。
- ❑ 其他需要读者自己修改的占位字符会使用类似 your_password、you_email 的文本。

最后，为了尽量让正文保持简洁，每一章新涉及的 Python 库都会在第一小节前汇总列出对应的版本和相关链接（比如主页、源码和文档）。因为大部分项目在 PyPI 上提供的介绍都不够完善，除非程序有独立的主页，否则会优先使用 GitHub 或 BitBucket 上的项目页面作为主页。

读者反馈与疑问

由于笔者水平有限，编写时间也比较仓促，书中难免有错误或者不全面的地方，在此恳请读者朋友批评指正。

关于本书的疑问和反馈可以到本书在 GitHub 上的项目仓库 HelloFlask (<https://github.com/greyli/helloflask>) 中创建 Issue 并提交; 书中的错误或笔误可以修改仓库中的勘误文件 (Errata.md) 并提交 Pull Request。

对于示例程序的疑问、反馈和改进建议可以到示例程序在 GitHub 上的项目仓库提交 Issue 或 Pull Request, 具体的网址可以在对应的章节看到。

当然, 你也可以直接发邮件与笔者联系, 笔者的邮箱是 withlihui@gmail.com。

本书的配套资源索引可以在本书的主页 <http://helloflask.com/book> 上看到。另外, 你可以访问笔者的博客 (<http://greyli.com>) 或是知乎专栏 “Hello, Flask!” (<https://zhuoanlan.zhihu.com/flask>) 阅读更多与 Flask 相关的文章。

致谢

首先, 感谢机械工业出版社华章公司的杨福川老师和李艺老师。因为杨老师的信任, 才让笔者有幸写作这本书。本书能够顺利完成, 离不开两位老师的悉心指导, 更离不开其他编辑的辛苦工作。

其次, 感谢 Flask 社区和其他开源项目的贡献者们创造了这一切, 也感谢在 Stack Overflow、GitHub、Reddit 和 Wikipedia 等网站贡献知识的开发者们。

最后, 感谢父母和奶奶这段时间的支持和帮助, 也感谢女友魏瑶和弟弟家辉给予的鼓励和陪伴。

目 录 Contents

前言

第一部分 基础篇

第 1 章 初识 Flask	2
1.1 搭建开发环境	3
1.1.1 Pipenv 工作流	3
1.1.2 安装 Flask	7
1.1.3 集成开发环境	8
1.2 Hello, Flask!	11
1.2.1 创建程序实例	11
1.2.2 注册路由	12
1.3 启动开发服务器	14
1.3.1 Run, Flask, Run!	14
1.3.2 更多的启动选项	18
1.3.3 设置运行环境	18
1.4 Python Shell	20
1.5 Flask 扩展	21
1.6 项目配置	22
1.7 URL 与端点	23
1.8 Flask 命令	23
1.9 模板与静态文件	24
1.10 Flask 与 MVC 架构	25

1.11 本章小结	26
-----------------	----

第 2 章 Flask 与 HTTP

2.1 请求响应循环	27
2.2 HTTP 请求	29
2.2.1 请求报文	29
2.2.2 Request 对象	31
2.2.3 在 Flask 中处理请求	34
2.2.4 请求钩子	37
2.3 HTTP 响应	38
2.3.1 响应报文	39
2.3.1 在 Flask 中生成响应	40
2.3.2 响应格式	43
2.3.3 来一块 Cookie	46
2.3.4 session: 安全的 Cookie	49
2.4 Flask 上下文	54
2.4.1 上下文全局变量	54
2.4.2 激活上下文	55
2.4.3 上下文钩子	56
2.5 HTTP 进阶实践	57
2.5.1 重定向回上一个页面	57
2.5.2 使用 AJAX 技术发送异步请求	60
2.5.3 HTTP 服务器端推送	64

2.5.4 Web 安全防范	65	4.3 处理表单数据	112
2.6 本章小结	74	4.3.1 提交表单	112
第 3 章 模板	75	4.3.2 验证表单数据	113
3.1 模板基本用法	75	4.3.3 在模板中渲染错误消息	117
3.1.1 创建模板	76	4.4 表单进阶实践	118
3.1.2 模板语法	77	4.4.1 设置错误消息语言	118
3.1.3 渲染模板	78	4.4.2 使用宏渲染表单	120
3.2 模板辅助工具	80	4.4.3 自定义验证器	121
3.2.1 上下文	80	4.4.4 文件上传	122
3.2.2 全局对象	82	4.4.5 使用 Flask-CKEditor 集成富 文本编辑器	129
3.2.3 过滤器	83	4.4.6 单个表单多个提交按钮	132
3.2.4 测试器	85	4.4.7 单个页面多个表单	133
3.2.5 模板环境对象	87	4.5 本章小结	137
3.3 模板结构组织	88	第 5 章 数据库	138
3.3.1 局部模板	88	5.1 数据库的分类	139
3.3.2 宏	88	5.1.1 SQL	139
3.3.3 模板继承	90	5.1.2 NoSQL	139
3.4 模板进阶实践	93	5.1.3 如何选择?	140
3.4.1 空白控制	93	5.2 ORM 魔法	140
3.4.2 加载静态文件	94	5.3 使用 Flask-SQLAlchemy 管理 数据库	142
3.4.3 消息闪现	98	5.3.1 连接数据库服务器	142
3.4.4 自定义错误页面	100	5.3.2 定义数据库模型	144
3.4.5 JavaScript 和 CSS 中的 Jinja2	101	5.3.3 创建数据库和表	145
3.5 本章小结	103	5.4 数据库操作	146
第 4 章 表单	104	5.4.1 CRUD	147
4.1 HTML 表单	104	5.4.2 在视图函数里操作数据库	151
4.2 使用 Flask-WTF 处理表单	106	5.5 定义关系	156
4.2.1 定义 WTForms 表单类	106	5.5.1 配置 Python Shell 上下文	157
4.2.2 输出 HTML 代码	109	5.5.2 一对多	158
4.2.3 在模板中渲染表单	110		

5.5.3	多对一	164	7.1.1	配置文件	196
5.5.4	一对一	165	7.1.2	创建程序实例	197
5.5.5	多对多	166	7.2	Web 程序开发流程	198
5.6	更新数据库表	168	7.2.1	程序功能设计	199
5.6.1	重新生成表	168	7.2.1	前端页面开发	200
5.6.2	使用 Flask-Migrate 迁移数据库	169	7.2.3	后端程序开发	202
5.6.3	开发时是否需要迁移?	171	7.3	使用 Bootstrap-Flask 简化页面编写	206
5.7	数据库进阶实践	172	7.3.1	加载资源文件	207
5.7.1	级联操作	172	7.3.2	快捷渲染表单	207
5.7.2	事件监听	175	7.4	使用 Flask-Moment 本地化日期和时间	209
5.8	本章小结	177	7.4.1	本地化前的准备	209
第 6 章	电子邮件	178	7.4.2	使用 Flask-Moment 集成 Moment.js	209
6.1	使用 Flask-Mail 发送电子邮件	179	7.4.3	渲染时间日期	210
6.1.1	配置 Flask-Mail	179	7.5	使用 Faker 生成虚拟数据	213
6.1.2	构建邮件数据	182	7.6	使用 Flask-DebugToolbar 调试程序	215
6.1.3	发送邮件	182	7.7	Flask 配置的两种组织形式	216
6.2	使用事务邮件服务 SendGrid	183	7.7.1	环境变量优先	217
6.2.1	注册 SendGrid	183	7.7.2	实例文件夹覆盖	217
6.2.2	SendGrid SMTP 转发	185	7.8	本章小结	218
6.2.3	SendGrid Web API 转发	185	第 8 章	个人博客	219
6.3	电子邮件进阶实践	188	8.1	大型项目结构	220
6.3.1	提供 HTML 正文	188	8.1.1	使用蓝本模块化程序	221
6.3.2	使用 Jinja2 模板组织邮件正文	189	8.1.2	使用类组织配置	227
6.3.3	异步发送邮件	191	8.1.3	使用工厂函数创建程序实例	228
6.4	本章小结	192	8.2	编写程序骨架	232
第二部分 实战篇			8.2.1	数据库	233
第 7 章	留言板	194	8.2.2	模板	240
7.1	使用包组织代码	195			

8.2.3	表单	246	9.1.3	混合式架构	303
8.2.4	视图函数	249	9.1.4	如何选择	303
8.2.5	电子邮件支持	249	9.2	编写程序骨架	303
8.3	编写博客前台	251	9.2.1	数据库模型与虚拟数据	305
8.3.1	分页显示文章列表	251	9.2.2	模板与静态文件	307
8.3.2	显示文章正文	258	9.3	高级用户认证	308
8.3.3	文章固定链接	259	9.3.1	用户注册	309
8.3.4	显示分类文章列表	262	9.3.2	验证邮箱地址	311
8.3.5	显示评论列表	263	9.3.3	使用装饰器过滤未确认用户	315
8.3.6	发表评论与回复	266	9.3.4	密码重置	316
8.3.7	支持回复评论	267	9.4	基于用户角色的权限管理	319
8.3.8	网站主题切换	269	9.4.1	角色与权限模型	319
8.4	初始化博客	271	9.4.2	设置角色与权限	320
8.4.1	安全存储密码	271	9.4.3	写入角色与权限	321
8.4.2	创建管理员用户	273	9.4.4	验证用户权限	323
8.5	使用 Flask-Login 管理用户认证	275	9.5	使用 Flask-Dropzone 优化文件上传	325
8.5.1	获取当前用户	276	9.5.1	配置 Flask-Dropzone	326
8.5.2	登入用户	277	9.5.2	渲染上传区域	328
8.5.3	登出用户	278	9.5.3	处理并保存上传图片	329
8.5.4	视图保护	279	9.6	使用 Flask-Avatars 处理用户头像	334
8.6	使用 CSRFProtect 实现 CSRF 保护	281	9.6.1	默认头像	335
8.7	编写博客后台	283	9.6.2	生成随机头像	335
8.7.1	文章管理	284	9.7	图片展示与管理	337
8.7.2	评论管理	291	9.7.1	在用户主页显示图片列表	338
8.7.3	分类管理	297	9.7.2	图片详情页	341
8.8	本章小结	298	9.7.3	上一张下一张跳转	342
			9.7.4	删除确认模态框	344
			9.7.5	举报图片	346
第 9 章	图片社交网站	299	9.7.6	图片描述	347
9.1	项目组织架构	300	9.7.7	图片标签	349
9.1.1	功能式架构	300	9.7.8	用户资料弹窗	353
9.1.2	分区式架构	302	9.8	收藏图片	358

9.8.1	使用关联模型表示多对多关系	358	9.14	编写网站后台	407
9.8.2	添加和取消收藏	360	9.14.1	用户管理	408
9.8.3	收藏者和收藏页面	362	9.14.2	资源管理	411
9.9	用户关注	365	9.14.3	面向管理员的用户资料编辑	412
9.9.1	自引用多对多关系	365	9.15	本章小结	413
9.9.2	关注与取消关注	366			
9.9.3	显示关注用户列表	369	第 10 章 待办事项程序		415
9.9.4	使用 AJAX 在弹窗中执行关注操作	371	10.1	使用 JavaScript 和 AJAX 编写单页程序	417
9.10	消息提醒	378	10.1.1	单页程序的模板组织	418
9.10.1	提醒消息在数据库中的表示	379	10.1.2	在根页面内切换子页面	421
9.10.2	创建提醒	379	10.1.3	生成测试账户	423
9.10.3	显示和管理提醒	380	10.1.4	添加新待办条目	424
9.10.4	通过轮询实时更新未读计数	382	10.2	国际化与本地化	426
9.11	用户资料与账户设置	384	10.2.1	使用 Flask-Babel 集成 Babel	427
9.11.1	编辑个人资料	385	10.2.2	区域和语言	427
9.11.2	自定义头像	386	10.2.3	文本的国际化	432
9.11.3	更改密码	392	10.2.4	文本的本地化	433
9.11.4	提醒消息开关	394	10.2.5	时间与日期的本地化	438
9.11.5	将收藏设为仅自己可见	395	10.3	设计并编写 Web API	440
9.11.6	注销账户	396	10.3.1	认识 Web API	441
9.12	首页与探索	397	10.3.2	设计优美实用的 Web API	443
9.12.1	获取正在关注用户的图片	399	10.3.3	使用 Flask 编写 Web API	446
9.12.2	使用联结和分组查询获取热门标签	401	10.3.4	使用 OAuth 认证	453
9.12.3	使用数据库通用函数获取随机图片	402	10.3.5	资源的序列化	461
9.13	使用 Flask-Whooshee 实现全文搜索	403	10.3.6	资源的反序列化	465
9.13.1	创建索引	404	10.3.7	Web API 的测试与发布	470
9.13.2	搜索表单	405	10.4	本章小结	473
9.13.3	显示搜索结果	406			
			第 11 章 在线聊天室		474
			11.1	编写程序骨架	476
			11.2	Gravatar 头像	477