《Flask Web开发：基于Py》

图书基本信息

书名：《Flask Web开发：基于Python的Web应用开发实战》

13位ISBN编号：978711537399X

出版时间：2014-12

作者：[美] Miguel Grinberg

页数：224

译者：安道

《Flask Web开发：基于Py》

内容概要

《Flask Web开发：基于Py》

作者简介

Miguel Grinberg
拥有25年开发经验的高级软件工程师，目前为广播公司开发视频软件。他常在个人博客（blog.miguelgrinberg.com）上撰写各类博文，内容主要涉及Web开发、机器人技术、摄影，偶尔也会有一些影评。他和妻子、四个孩子、两只狗和一只猫共同生活在俄勒冈州波特兰市。Twitter：@miguelgrinberg。

《Flask Web开发：基于Py》

书籍目录

精彩短评

《Flask Web开发：基于Py》

精彩书评

1、这本书非常适合Flask 入门，虽然说是入门书，但是我个人认为可以作为初中级的参考书籍，因为作者对 Flask 的讲解把握的非常到位，所以可以将很多概念和扩展的使用讲解得通俗易懂，但是，又不乏给有心人留下很多空间去进行自我寻找资料扩展学习。这本书我读了3遍了现在，也不敢说书中提及的所有内容我都了然于胸，当然，也可能是本人愚钝，没有得该书的精要。此外，还需要说明的是，本书作者也是 Pycon 的常客，如果大家有意思的话还可以去找找往年作者在 PyCon 上的分享，相信对自己加深 Flask 的理解大有帮助和另有一番感悟。

《Flask Web开发：基于Py》

章节试读

1、《Flask Web开发：基于Python的Web应用开发实战》的笔记-第6页

```
    # 虚拟环境
#### 使用虚拟环境的好处
- 避免包的混乱和版本的冲突
- 不需要管理员权限

### 安装
虚拟环境曾使用第三方实用工具 [virtualenv](https://virtualenv.pypa.io/en/stable/#) 创建，但 Python 3.3 开始通过 [venv模块](https://docs.python.org/3/library/venv.html) 原生支持虚拟环境，可以完全代替 virtualenv 。

### 创建和激活
参见Python 官方文档中 [Virtual Environments and Packages](https://docs.python.org/3/tutorial/venv.html#virtual-environments-and-packages) 和 [venv模块](https://docs.python.org/3/library/venv.html) 的说明。
#### 在当前目录创建
`$ pyvenv venv`
`$ python -m venv venv` （Windows下有效）
#### 激活
`$ source venv/bin/activate`
`$ venv/Scripts/activate`
#### 使用 pip
- 安装：
`$ pip install &lt;package_name&gt;`
- 特定版本的安装
`$ pip install &lt;package_name&gt;==&lt;package_version&gt;`
- 升级
`$ pip install --upgrade &lt;package_name&gt;`
- 卸载
`$ pipuninstall &lt;package_name&gt;`
- 显示包信息
 `$ pip show &lt;package_name&gt;`
- 显示所有安装的包
`$ pip list`
- 导出所有安装包的信息
`$ pip freeze &gt; requirements.txt`
- 安装需求文件中的所有包
`$ pip install -r requirements.txt`

#### 关于 [Pycharm](https://www.jetbrains.com/pycharm/)
若使用Pycharm管理项目，也可以方便地[创建虚拟环境](https://www.jetbrains.com/help/pycharm/2016.1/creating-virtual-environment.html) 。
若要在 Pycharm 的命令行中直接激活虚拟环境，可参考 stackoverflow 上的[这个问题的答案](http://stackoverflow.com/questions/22288569/how-do-i-activate-a-virtualenv-inside-pycharms-terminal)：PyCharm 2016.1 or 2016.2:Settings,Tools,Terminal, and add""/K &lt;path-to-your-activate.bat&gt;""toShell path and add (mind the quotes). Also add quotes around cmd.exe, resulting in:
```

`"cmd.exe" /k ""C:\mypath\my-venv\Scripts\activate.bat""`

2、《Flask Web开发：基于Python的Web应用开发实战》的笔记-第17页

请问网络中的其他电脑怎么访问啊？

3、《Flask Web开发：基于Python的Web应用开发实战》的笔记-第42页

这一章看着真吃力啊

4、《Flask Web开发：基于Python的Web应用开发实战》的笔记-第17页

```
@app.route(404)
def page_not_found(e):
  return render_template('404.html'),404

@app.route(500)
def internal_server_error(e):
  return render_template('500.html'),500
```

请问被装饰的函数为什么参数是e呢？

5、《Flask Web开发：基于Python的Web应用开发实战》的笔记-2.2 路由和视图函数

#路由和视图函数
##概念
- 客户端发送URL请求给Web服务器，服务器再通过WSGI协议将请求发送给Flask程序实例。
- Flask程序实例使用请求的URL所映射的**路由**来处理相应的请求。
- 一旦路由收到请求，就会执行其中的**视图函数**，而且路由中的变量部分可以以关键字参数传递给视图函数。
- 视图函数执行后会将返回值反馈给客户端，这个返回值就是**响应**。

##URL路由的注册
在路由系统中定义规则可以的方法可以概括为三种:
1. 使用 flask.Flask.route() 装饰器
2. 使用 flask.Flask.add_url_rule() 函数
3. 直接访问暴露为 flask.Flask.url_map 的底层的 Werkzeug 路由系统

##路由的变量
前面提到变量部分可以以关键字参数传递给视图函数。
路由中的变量部分用尖括号指定（`/user/&lt;username&gt;`），默认使用不带斜线的字符串，不过也可使用类型定义。
Flask支持在路由中使用的变量类型有：

类型	说明
string	接受任何不带斜线的字符串
int	接受整数
float	同 int，但是接受浮点数
path	和默认的相似，但也接受斜线

###*参考资料：*
[Flask API URL路由注册](http://docs.jinkan.org/docs/flask/api.html#url)

6、《Flask Web开发：基于Python的Web应用开发实战》的笔记-第159页

##为什么
- Web服务需要用户的认证状态来保护信息的安全
- REST Web 服务是无状态的，客户端发出的请求必须包含认证信息

7、《Flask Web开发：基于Python的Web应用开发实战》的笔记-第7页

#WSGI
[WSGI](https://wsgi.readthedocs.io/en/latest/)(Web Server Gateway Interface, Web服务器网关接口）Web服务器网关接口（Python Web Server Gateway Interface，缩写为WSGI）是为Python语言定义的Web服务器和Web应用程序或框架之间的一种简单而通用的接口。自从WSGI被开发出来以后，许多其它语言中也出现了类似接口。WSGI是作为Web服务器与Web应用程序或应用框架之间的一种低级别的接口，以提升可移植Web应用开发的共同点。WSGI是基于现存的CGI标准而设计的。
WSGI区分为两个部份：一为“ 服务器 ”或“ 网关 ”，另一为“ 应用程序 ”或“ 应用框架 ”。在处理一个WSGI请求时，服务器会为应用程序提供环境资讯及一个回呼函数（Callback Function）。当应用程序完成处理请求后，透过前述的回呼函数，将结果回传给服务器。所谓的 WSGI 中间件同时实现了API的两方，因此可以在WSGI服务和WSGI应用之间起调解作用：从WSGI服务器的角度来说，中间件扮演应用程序，而从应用程序的角度来说，中间件扮演服务器。“ 中间件 ”组件可以执行以下功能：
重写环境变量后，根据目标URL，将请求消息路由到不同的应用对象。
允许在一个进程中同时运行多个应用程序或应用框架。
负载均衡和远程处理，通过在网络上转发请求和响应消息。
进行内容后处理，例如应用XSLT样式表。*参考资料*：
1. [网关协议学习：CGI、FastCGI、WSGI](http://www.biaodianfu.com/cgi-fastcgi-wsgi.html)
2. [Wsgi研究](http://blog.kenshinx.me/blog/wsgi-research/)
3. [化整為零的次世代網頁開發標準: WSGI](http://blog.ez2learn.com/2010/01/27/introduction-to-wsgi/)

#Flask类
class flask.Flask(import_name, static_path=None, static_url_path=None, static_folder='static', template_folder='templates', instance_path=None, instance_relative_config=False)
一般FLASK类创建程序实例的方法如下：from flask import Flask
app = Flask(__name__)
##import_name
这是Flask类唯一一个必须指定的参数，Flask用该参数决定程序的根目录，以便稍后能够找到相对于程序根目录的资源文件位置。
该参数一般用“ __name__ ”,但更推荐使用程序主模块或包的名字，因为这样更便于调试debug。Why is that? The application will work even with __name__, thanks to how resources are looked up. However it will make debugging more painful. Certain extensions can make assumptions based on the import name of your application. For example the Flask-SQLAlchemy extension will look for the code in your application that triggered an SQL query in debug mode. If the import name is not properly set up, that debugging information is lost. (For

example it would only pick up SQL queries in yourapplication.app and not yourapplication.views.frontend)因此如果你的程序是在 *yourapplication/app.py* 中定义的，那么你最好从以下方法中选择一种来创建你的程序实例：app = Flask('yourapplication')
app = Flask(__name__.split('.')[0])##其他参数
其他参数用于指定模板和资源的路径，默认为程序根目录下的 *template* 和 *static* 文件夹。因此为了避免不必要的麻烦，这两个文件夹的名称注意不要取错。
###*参考资料：*
[Flask API 应用对象](http://docs.jinkan.org/docs/flask/api.html#id2)

8、《Flask Web开发：基于Python的Web应用开发实战》的笔记-第69页

##什么是蓝图？##
一个蓝图定义了可用于单个应用的视图，模板，静态文件等等的集合。

##蓝图作用##
根据应用的功能将代码分割开来，组织成不同的组件，这些组件之间共享应用配置

参考资料：
http://docs.jinkan.org/docs/flask/blueprints.html
[蓝图 | Flask之旅】(https://spacewander.github.io/explore-flask-zh/7-blueprints.html)

9、《Flask Web开发：基于Python的Web应用开发实战》的笔记-第15页

1. 请求对象##
###需求###
虽然通过路由可以建立起URL请求和视图函数的映射关系，在客户端发起请求时调用对应的视图函数（其中路由中的动态变量还可以传递给视图函数），但这还不够。为了让视图函数能够处理客户端发来的请求，它还必须要能够访问客户端发来的数据信息。
###作用###
Flask使用**上下文**中临时的** request**[(API)][1] 对象向视图函数提供必要的信息，它的作用有：
- 封装客户端发来的 HTTP 请求
- 作为全局可访问的对象，向视图函数统一地提供请求数据
- 得益于上下文，在多线程环境中，Flask可以保证[总会在当前线程上获取正确的数据][4]

##2. 上下文##
###概念###
[@vczh](https://www.zhihu.com/question/26387327/answer/32611575)在知乎回答“什么是上下文”提到：每一段程序都有很多外部变量。只有像Add这种简单的函数才是没有外部变量的。一旦你的一段程序有了外部变量，这段程序就不完整，不能独立运行。你为了使他们运行，就要给所有的外部变量一个一个写一些值进去。这些值的集合就叫上下文。从这个角度理解，上下文并非天然存在的，而是类似于某种特定环境。**当你使用了“外部变量”与一个应用或一段程序发生交互，也就产生了相应的上下文；或者说，只有在特定的上下文中，这些“外部变量”才有意义。**

###上下文变量

Flask中存在两种上下文：[应用上下文][2]和[请求上下文][3]，这可以通过相应上下文提供的变量类型来区分判断。
需要注意的是，如上文“ 请求对象 ”中所讲，这里的变量都是被封装成对象来使用的。
应用上下文的变量有：
- **current_app** 当前活动应用的实例对象
- **g** 应用在处理请求时用来储存临时信息的对象

请求上下文的变量有：
- **request** 封装了客户端发来的 HTTP 请求的请求对象
- **session** 一个形式为字典的用户会话对象，储存不同请求间需要“ 记住 ”的值

###应用状态和上下文的激活###
可以认为，Flask对象在实例化后在模块层次上应用有[两个状态][2]。
第一个状态为“ 休息 ”状态，该状态的情况是：
- 目前没处理任何请求
- 可以安全地修改应用对象
- 必须得有一个指向对象的引用来修改它，因为不会有某个神奇的代理变量（
current_app）指向你刚创建的或者正在修改的应用对象

当激活一个请求前，应用开始进入“ 待命 ”状态，激活上下文：
- 激活应用上下文：
current_app 指向实例化的应用对象，线程中开始可以
- 激活请求上下文：
request 和 *session* 指向当前的请求

此时，视图函数就可以一边从请求上下文中获取请求，一边从应用上下文中与实例对象进行交互，从而给出正确的响应。

###命令行应用###
一般情况下，Flask会在分发请求之前激活（或推送）上下文，请求处理完成后再将其删除。但在Python shell会话中，需要手动激活上下文：
```>> from hello import app
>> from flask import current_app
>> app_ctx = app.app_context()
>> app_ctx.push()
>> current_app.name
'hello'
>> app_ctx.pop()
```###[代理对象][5]###
需要注意的是，Flask实际提供的是其他对象的代理，如果你需要访问潜在的被代理的对象，你可以使用 _get_current_object() 方法。

##3. 请求钩子##
上下文的一个典型应用场景就是用来缓存一些我们需要在发生请求之前要使用的资源。但光有上下文变量还不够，我们需要注册通用函数在请求开始前对这些变量进行操作，如创建数据库连接或认证发起请求的用户，这些函数叫*请求钩子函数*。

[1]: http://docs.jinkan.org/docs/flask/api.html#id4
[2]: http://docs.jinkan.org/docs/flask/appcontext.html#app-context
[3]: http://docs.jinkan.org/docs/flask/reqcontext.html?highlight=%E8%AF%B7%E6%B1%82%E4%B8%8A%E4%B8%8B%E6%96%87#request-context
[4]: http://docs.jinkan.org/docs/flask/api.html#flask.request
[5]: http://docs.jinkan.org/docs/flask/reqcontext.html#notes-on-proxies

10、《Flask Web开发：基于Python的Web应用开发实战》的笔记-第19页

##模板作用##
视图函数的作用是：处理请求，生成响应。
但视图函数处理请求不单单只是为了生成响应，还需要通过上下文与数据库（以及其他资源）进行交互。可以说，数据库等Web资源才是根本，请求和响应都是基于这些资源而产生的，没有资源就无从谈起。
从这个角度讲，视图函数的处理可以明显地分为两个过程，与内部资源进行交互的**业务逻辑**和生成响应的**表现逻辑**。
将变现逻辑从视图函数中抽离出来，转移到**模板**中，实现两个过程的分化，可以方便代码的理解和维护。在这里，模板承担了视图函数原有的一部分功能，它也有动态变量，可以与上下文进行交互（它的这一处理过程叫**渲染**），但模板更重要的作用是通过丰富的HTML定义和CSS样式定义响应的表现形式。

#3.1 模板变量#
下面的全局变量默认在 Jinja2 模板中可用:
- config
当前的配置对象 (flask.config)
- request
当前的请求对象 (flask.request)。当模版不是在活动的请求上下文中渲染时这个变量不可用。
- session
当前的会话对象 (flask.session)。当模版不是在活动的请求上下文中渲染时这个变量不可用。
- g
请求相关的全局变量 (flask.g)。当模版不是在活动的请求上下文中渲染时这个变量不可用。
- url_for()
flask.url_for() 函数
- get_flashed_messages()
flask.get_flashed_messages() 函数

11、《Flask Web开发：基于Python的Web应用开发实战》的笔记-第189页

1. 注册Heroku账户
2. 安装 Heroku Toolbelt
`$ heroku login`
3. 创建程序
`$ heroku create <appname>

4. 配置数据库
`$ heroku addons:create heroku-postgresql:hobby-dev`
提升数据库为主数据库
`$ heroku pg:promote HEROKU_POSTGRESQL_BROWN_URL

5. 执行部署
`$ git push heroku master`
`$ heroku run python manage.py deploy`
`$ heroku restart

6. 查看日志
`$ heroku logs`
`$ heroku logs -t`

7. 部署一次升级
<代码开始>
$ heroku maintenance:on
$ git push heroku master
$ heroku runpython manage.py depoly
$ heroku restart
$ heroku maintenance:off
```

参考资料：
1. [【flasky/heroku/部署】（欢迎大家挑刺和补充）可能是目前为止最详细的了](http://cocode.cc/t/flasky-heroku/6589)
2. [分享一下flask程序部署到heroku平台的过程](http://cocode.cc/t/flask-heroku/4253)
3. [(first update)终于成功部署在heroku上了，欢迎交流](http://cocode.cc/t/first-update-heroku/3711)

# 注意事项
- 使用python3
heroku默认使用的是python 2.7，如果需要使用 python3，则需要在应用根目录下添加 *runtime.txt* 文件，内容为：python-3.5.2具体为所使用的python版本号，参见[资料](http://stackoverflow.com/questions/32252124/deploying-with-python-3-on-heroku)
- html5lib.sanitizer not found 错误
为避免出现 sanitizer 错误，请将 html5lib 的版本号设置为 0.9999999。
若已经遭遇，使用以下命令行将 html5lib 降级：
`$ pip install --upgrade bleach`
参见[资料](https://github.com/taigaio/taiga-back/issues/794)
```

《Flask Web开发：基于Py》

版权说明