


# flask-api 使用手册



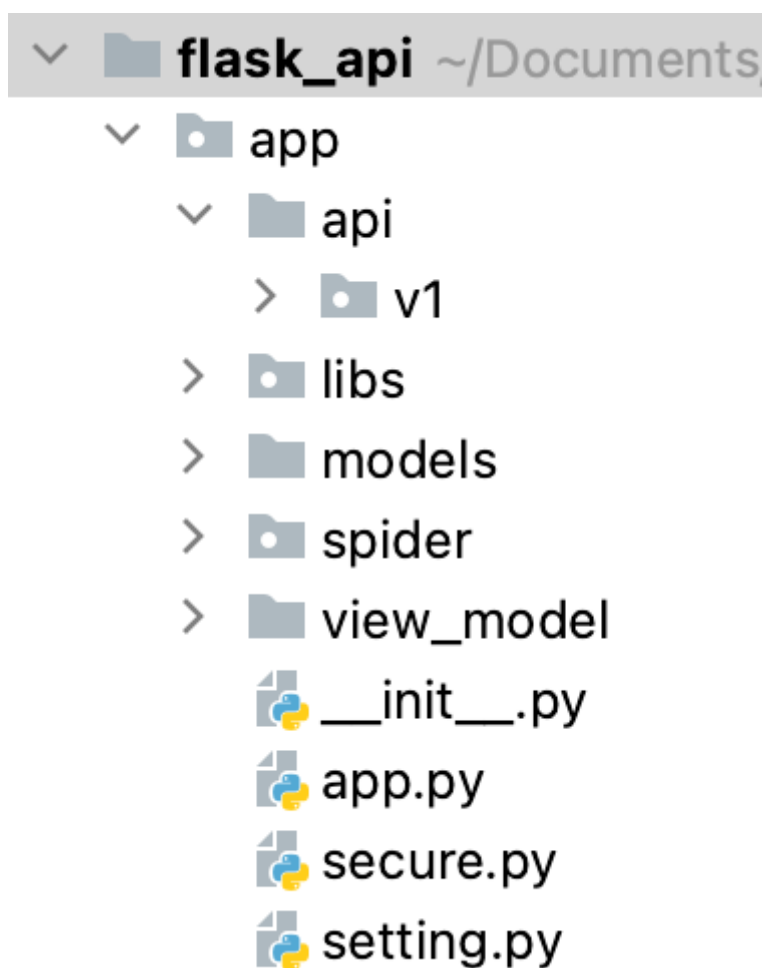
東南大學  
SOUTHEAST UNIVERSITY

Author: sony的小鼠 

此项目是笔者从之前项目中精简而来，旨在方便开发者简单开启后端接口服务

代码设计思想与文件管理方式源于MVC+RESTful

以下为项目代码组织结构，之后会——讲解

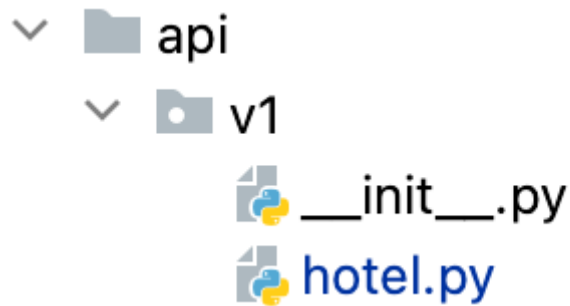


## 1、视图函数的注册

由于采用RESTful接口规范，于是将服务器中数据当成资源进行接口设计

故推荐将每个数据类型的视图函数单独写入一个py文件

例：



将hotel的所有数据视作服务器上一类资源，并需要编写相关视图函数以提供接口服务

api为视图函数组织文件夹，v1代表版本号（供后续迭代升级），各类资源的视图函数集中到单独的py文件中，如hotel

代码组织较为繁琐，不推荐一上手便去深入细节，重点关注使用方式

## 视图函数注册方式

1. 请在各类资源的py文件当中导入Redprint，并实例化该对象

以hotel.py举例

即在hotel.py当中添加以下代码

```
from app.libs.redprint import Redprint
```

```
api = Redprint('hotel')
```

2. 请将实例化后的Reprint对象注册到上层Blueprint对象中

以hotel.py为例

即在v1/ \_\_init\_\_文件当中添加以下代码

```
from flask import Blueprint
```

```
from app.api.v1 import hotel
```

```
2 usages  👤 moleofsony +1 *
```

```
def create_blueprint_v1():
```

```
    bp_v1 = Blueprint('v1', __name__)
```

```
#---->please add code in this section <----#
```

```
    hotel.api.register(bp_v1)
```

```
#---->please add code in this section <----#
```

```
    return bp_v1
```

若添加新模块，需要在该文件当中导入相应的包

之后，在标注区域添加XX.api.register(bp\_v1)

### 3. 请根据对资源的请求动作设计url

如获取hotel信息，那他的动作是get,故url为"v1\hotel\get"(蓝图层的'v1'(构建Blueprint时传入),红图层的'hotel'(构建Redprint时传入),请求动作'get')

视图函数和url的绑定方式如下

```
@api.route('/get', methods=['GET'])
def get_hotel():
    return 'hotel!!!'
```

路由绑定涉及到python装饰器语法，较为复杂，不做讲解，会用即可

## 2、数据库的映射与查询

---

## 3、模型序列化的使用方式

---

## 4、视图层使用推荐

---

## 5、工具函数定义

---

## 6、RESTful抛异常的方式

---

## 7、爬虫函数编写

---