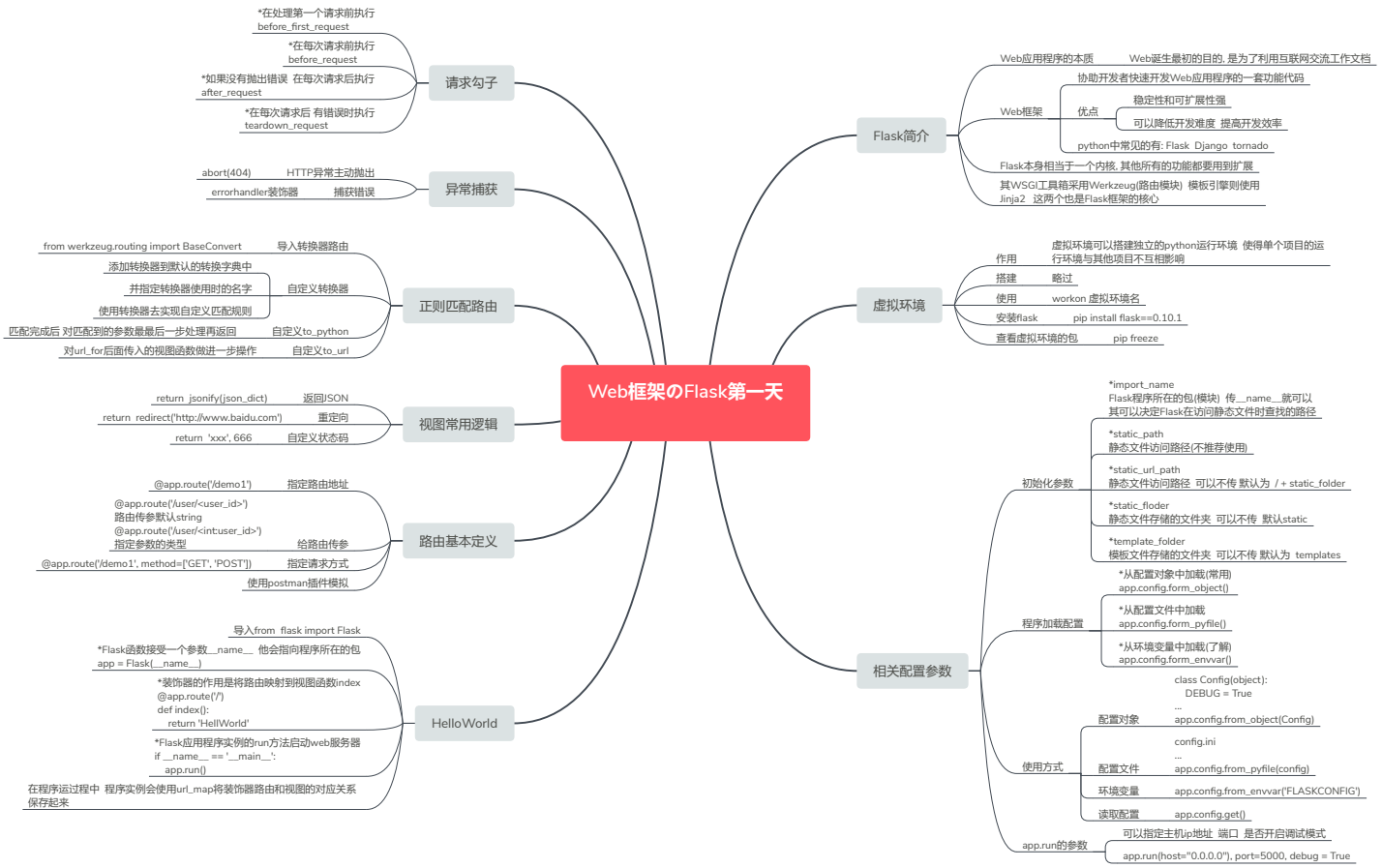
Web框架のFlask第一天
Flask简介
Web应用程序的本质 Web诞生最初的目的, 是为了利用互联网交流工作文档
Web框架
协助开发者快速开发Web应用程序的一套功能代码
优点
稳定性和可扩展性强
可以降低开发难度 提高开发效率
python中常见的有: Flask Django tornado
Flask本身相当于一个内核, 其他所有的功能都要用到扩展
其WSGI工具箱采用Werkzeug(路由模块) 模板引擎则使用Jinja2 这两个也是Flask框架的核心
虚拟环境
作用
虚拟环境可以搭建独立的python运行环境 使得单个项目的运行环境与其他项目不互相影响
搭建 略过
使用 workon 虚拟环境名
安装flask pip install flask==0.10.1
查看虚拟环境的包 pip freeze
相关配置参数
初始化参数
*import_name
Flask程序所在的包(模块) 传__name__就可以
其可以决定Flask在访问静态文件时查找的路径
*static_path
静态文件访问路径(不推荐使用)
*static_url_path
静态文件访问路径 可以不传 默认为 / + static_folder
*static_floder
静态文件存储的文件夹 可以不传 默认static
*template_folder
模板文件存储的文件夹 可以不传 默认为 templates
程序加载配置
*从配置对象中加载(常用)
app.config.form_object()
*从配置文件中加载
app.config.form_pyfile()
*从环境变量中加载(了解)
app.config.form_envvar()
使用方式
配置对象
class Config(object):
DEBUG = True
...
app.config.from_object(Config)
配置文件
config.ini
...
app.config.from_pyfile(config)
环境变量 app.config.from_envvar('FLASKCONFIG')
读取配置 app.config.get()
app.run的参数
可以指定主机ip地址 端口 是否开启调试模式
app.run(host="0.0.0.0"), port=5000, debug = True
HelloWorld
导入from flask import Flask
*Flask函数接受一个参数__name__ 他会指向程序所在的包
app = Flask(__name__)
*装饰器的作用是将路由映射到视图函数index
@app.route('/')
def index():
return 'HellWorld'
*Flask应用程序实例的run方法启动web服务器
if __name__ == '__main__':
app.run()
在程序运过程中 程序实例会使用url_map将装饰器路由和视图的对应关系保存起来
路由基本定义
@app.route('/demo1') 指定路由地址
@app.route('/user/<user_id>')
路由传参默认string
@app.route('/user/<int:user_id>')
指定参数的类型 给路由传参
@app.route('/demo1', method=['GET', 'POST']) 指定请求方式
使用postman插件模拟
视图常用逻辑
return jsonify(json_dict) 返回JSON
return redirect('http://www.baidu.com') 重定向
return 'xxx', 666 自定义状态码
正则匹配路由
from werkzeug.routing import BaseConvert 导入转换器路由
自定义转换器
添加转换器到默认的转换字典中
并指定转换器使用时的名字
使用转换器去实现自定义匹配规则
自定义to_python
匹配完成后 对匹配到的参数最最后一步处理再返回
自定义to_url
对url_for后面传入的视图函数做进一步操作
异常捕获
abort(404) HTTP异常主动抛出
errorhandler装饰器 捕获错误
请求勾子
*在处理第一个请求前执行
before_first_request
*在每次请求前执行
before_request
*如果没有抛出错误 在每次请求后执行
after_request
*在每次请求后 有错误时执行
teardown_request