**环境说明**

python -V pip -V

python3 -V pip3 -V

python pip conda

**Flask框架**

**Flask介绍**[**¶**](http://localhost:8888/notebooks/Desktop/tmp/%E6%AD%A6%E6%B1%89%E5%A4%A7%E5%AD%A6%E6%8E%88%E8%AF%BE%E6%96%87%E6%A1%A3/02/01Flask%E4%BB%8B%E7%BB%8D.ipynb#2%E3%80%81Flask%E4%BB%8B%E7%BB%8D)

Flask是一个基于Python开发并且依赖Jinja2模板（基于python的模板引擎）和WSGI服务的一个微型框架。

轻量级的框架，非常快速的就能把程序搭建起来。

**WSGI**

其用于接收http请求并对请求进行预处理，然后触发Flask框架，开发人员基于Flask框架提供的功能对请求进行相应的处理，并返回给用户。

**模版引擎**

如果要返回给用户复杂的内容时，需要借助jinja2模板来实现对模板的处理，即：将模板和数据进行渲染，将渲染后的字符串返回给用户浏览器。

**轻量级框架**

* “微”并不表示你需要把整个 Web 应用塞进单个 Python 文件，也不意味着 Flask 在功能上有所欠缺。
* 微框架中的“微”意味着 Flask 旨在保持核心简单而易于扩展。
* Flask 不会替你做出太多决策——比如使用何种数据库。
* 而那些 Flask 所选择的——比如使用何种模板引擎——则很容易替换。
* 除此之外的一切都由可由你掌握。如此，Flask 可以与您珠联璧合。
* 默认情况下，Flask 不包含数据库抽象层、表单验证，或是其它任何已有多种库可以胜任的功能。然而Flask 支持用扩展来给应用添加这些功能，如同是 Flask 本身实现的一样。众多的扩展提供了数据库集成、表单验证、上传处理、各种各样的开放认证技术等功能。Flask 也许是“微小”的，但它已准备好在需求繁杂的生产环境中投入使用。

**Django**

Django（重量级）：中间件配置、路由配置、ORM（数据库操作）、cookie、session、admin（脚手架）、from（表单）

Flask（轻量级）：中间件配置、路由配置

**Flask基本使用**

1、新建falsk项目 不建议使用系统环境

* 虚拟环境（按需加载，项目移植）
* pip install virtualenv
* 新建项目
* 新建虚拟环境 virtualenv 环境名字
  + virtualenv myenv
* 使用虚拟环境
  + 一般：myenv/bin/activate
  + widows：myenv/scripts/activate
* 在项目内安装依赖 pip install flask

**Flask request对象**

* 从flask模块导入request

**request**

* method-当前请求方法
* args -解析查询字符串（querystring）
  + get('属性名')
* form-解析表单参数
* files-与上传文件有关的数据

**Flask response对象**

**使用return响应内容**

web系统

* return '字符串'
* return render\_template 渲染模版和动态数据
* return redirect('其他路由路径')

分离

* 数据接口-------json
  + json.dumps() 将Python对象编码生成json字符串
  + json.loads() 已编码的json字符串-->Python对象

​ - json: import json

​ - from flask import Flask,redirect,render\_template,Response,request

flask：

* 能够搭建flask服务
* 能够根据业务配置路由
* flask进行 json数据响应
* flask 上传文件（图片）
* 基本的数据库的操作

# 数据库操作：ORM

M：Model数据模型

V：View视图

C：controller控制器

# Flask轻量级框架

* 安装依赖

-依赖 flask-sqlalchemy

-依赖 pymysql

pip install flask-sqlalchemy

pip install pymysql

# 安装数据库

* MySQL/MariaDB
* 数据库界面化工具：Navicat