## 安装flask

pip install flask

## flask第一个应用

from flask import Flask  
  
app = Flask(\_\_name\_\_)  
  
@app.route(**"/"**)  
def hello\_world():  
 return **"helloworld"**if \_\_name\_\_ == **'\_\_main\_\_'**:  
 app.run()

## MTV模型

from flask import Flask, render\_template  
  
app = Flask(\_\_name\_\_)  
  
@app.route(**"/"**)  
@app.route(**"/index"**)  
def hello\_world():  
 user = {  
 **'name'**: **"张三"** }  
 return render\_template(**"hello.html"**, user=user)  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 app.run()

## 命令行启动flask

1. 设置环境变量

windows: set FLASK\_APP=app.py

linux: export FLASK\_APP=app.py

1. flask run启动内置web服务器

flask run --host=0.0.0.0 --port=8001

或者

flask run -h 0.0.0.0 -p 8001

## 启动Debug模式

不设置是生产模式

windows: set FLASK\_ENV=development

Linux: export FLASK\_ENV=development

if \_\_name\_\_ == **'\_\_main\_\_'**:  
 app.run(debug=True)

## flask扩展

FlaskSQLAlchemy 数据库操作

FlaskWTF 表单相关扩展

FlaskRESTful 接口

## URL配置及路由

### 配置路由

#### 使用装饰器

@app.route(url\_name,methods)

methods:GET、POST

@app.route("/login",methods=['GET','POST'])

#### 2、使用API配置

app.add\_url\_rule(url,url\_name,view\_name)

url\_name：给url命名

view\_name:视图函数

app.add\_url\_rule('/home','home',hello\_world)

def hello\_world():

return "hello\_world"

### 路由传参

@app.route("/user/<username>")

指定参数类型

@app.route("/post/<int:post\_id>")

常见参数类型

string 默认类型

int

float

path 类似string，可包含斜杠

uuid

### url规则

查看url规则列表

print(app.url\_map)

[<Rule '/home',(HEAD, GET, OPTIONS)-> hello\_world>]

#### url逆向解析

Map([<Rule '/static/<filename>' (HEAD, OPTIONS, GET) -> static>])

根据名称解析成url字符串

url\_for(url\_name,\*\*kwargs)

# 静态文件引用

url\_for('static',filename='style.css')

## 视图函数取值

@app.route("/login")

@app.route("/login/<username>")

# 可不传值

def login(username=None):

pass

post或者get方式传值

get方式取值 args

request.args

# default是没取到，设置的默认值

request.args.get("name",default)

获取远程ip地址

request.remote\_addr

post方式取值from

## 上下文对象

应用上下文 current\_app g

@app.route(**'/h1'**)  
def h1():  
 print(app)  
 print(current\_app)  
 return **"h1"**

请求上下文

request 请求对象

session 请求会话，请求之间数据共享

cookie 是存在浏览器上面的

在分派请求之前激活上面两种上下文

请求处理完成之后被删除

### 请求钩子

装饰器,其他函数不需要调用，自然会默认调用

每个请求验证用户信息

限制恶意ip请求

# 服务器初始化后第一个请求到达之前

before\_first\_request

@app.before\_first\_request  
def before\_first\_request():  
 print(**"request hook")**

# 每个请求到达前

before\_request

@app.before \_request  
def before\_request():  
 print(**"request hook")**

# 每次请求后

after\_request

# 无论如何都执行,异常也执行

teardown\_request