# Flask

flask\_app = Flask(\_\_name\_\_)

# 原理

current\_app 使用了设计模式中的代理模式

request

session

g 本地代理

应用上下文-current\_app, g, session

请求上下文-request

离线应用-单元测试

app=Flask(\_\_name\_\_)

ctx=app.app\_context()//请求上下文

ctx.push()

a=current\_app.config[‘debug’]

ctx.pop()

## 使用with语句改写

with需要

#

#上下文管理器

# 实现\_\_enter\_\_(); \_\_exit\_\_()两个方法

# 上下文表达式必要返回一个上下文管理器

# with A() as AA: AA是class A()中\_\_enter\_\_的返回值

with app.app\_context():

current\_app[‘debug’]

## 线程隔离

## flask多线程

app = Flask(\_\_name\_\_)

app.run(threaded=true,)

# 生命周期

**@flask\_app.before\_reqeust**

def before\_request():

print（所有请求之前调用）

# 配置文件

setting.py

DEBUG = True //配置文件写法

app Flask核心对象

app.config.from\_object(“app.setting”) //读取/app/setting.py

读取配置文件

app.config[‘DEBUG’]

from flask import current\_app //当前flask核心对象

current\_app.config[‘DEBUG’]

# 蓝图

本质封装了werkzeug库

## 1、基础使用方法

1. **创建蓝图**

api = Blueprint(“api”, \_\_name\_\_)

1. **注册蓝图**

# 需要导入api模块

flask\_app.register\_blueprint(api)

1. **view使用蓝图**

# 需要导入api

@api.route(“/”)

def index():

print(“index.py”)

## 配置统一蓝图

flask\_app.register\_blueprint(api, **url\_prefix=’/admin’**)

访问的路径变成 [https://127.0.0.1](https://127.0.0.1/admin/index)**[/admin](https://127.0.0.1/admin/index)**[/index](https://127.0.0.1/admin/index)

/admin 是路由添加

## 路由参数

@api.route(“/api/<page>/<a>”, methods=[“GET”])

def api(page, a):pass //将值传递给api方法

## 获取GET参数

from flask import request #这个参数包含所有的请求参数，任何地方都可以获取到，通过视图函数创建

resqust.args.get(“A”, “”) //获取URL请求的参数

request.args.to\_dict() //转变成可变字典

make\_reponse()

# Form验证

## 创建wtform方法

pip install wtforms

from wtforms import Form, StringField

class XXX(Form):

# validators 是wtforms内置的验证方法

q = StringField(validators=[**PAMS, PAMS**])

#自定义验证

def

## 使用wtform

from mywforms import XXX

@api.router(“/index”, methods=[“GET”])

def index():

form = XXX(reqeust.args) //传入所有参数

if form.validate(): //方法验证

# 验证通过

form.q.data //获取key为q的数据

form.q.data.strip() //去掉空格

form.errors //错误信息

## Field类型

StringField, IntergerField

## PAMS-参数设置1

default=1 验证失败填写默认值1

Length(min=1, max=30) 验证长度1-30

NumberRange(min=1, max=30)

message=”11” 错误提示信息

DataRequired() 数据不能为空

## 自定义验证

# 模板

## 1、基础使用方法

**如果不配置templates\_dir参数，则默认访问启动文件下的templates目录**

**（1） 配置templates\_dir参数**

Flask\_app= Flask(\_\_name\_\_, templates\_dir=None)

**（2）视图调用：**

def hello(name=None):

return render\_template('hello.html', name=name)

return redirect(“/index”) # 路由跳转

## 2、模板配置路由

<link rel="stylesheet" href='{{ url\_for("static",filename = "css/index.css" ) }}'>

<script src="{{ url\_for("static",filename = "js/js.js") }}">

## 模板函数

# 将函数加入模板

flask\_app.add\_template\_global(“自定义函数”, “buildstatic”)

# 模板中使用

<img src=”{{buildstatic(1,1)}}” />

## 4、模板语法

{% if name==1%} {%else%} {% endif %}

{% for item in name %} {% else %} 为空执行 {% endfor %}

开发期间响应静态文件

# 静态文件

**如果不配置static\_dir参数，则默认访问启动文件下的static目录**

**（1） 配置static\_dir参数**

Flask\_app= Flask(\_\_name\_\_, static\_dir=None)

# 数据库

pip install flask-sqlalchemy

## 连接方法

pip install pymysql

配置文件：

mysql 数据类型

pymysql 数据库驱动

root 数据库用户名

123 数据库密码

@localhost 数据库IP

shoplow 库名

**SQLALCHEMY\_DATABASE\_URL** = "mysql+pymysql://root:123@localhost/shoplow"

## 创建model类

from sqlalchemy import Column

db = SQLAlchemy()

class Book(db.Model):

id = db.Column(**PAMS**)

def PAFunc1():pass

## PAMS-参数说明

Integer 整形

String(50) 字符串50

nullable=False 不为空

default=”” 默认值

primary\_key=True 主键

autoincrement=True 自增

## 将数据sqlalchemy连接到flask

from app.models import db //引入db=SQLAlchemy()

def create\_app():

app = Flask(\_\_name\_\_)

db.init\_app(app) //注册到flask

## 生成数据库表

1. 引入所有model
2. db.create\_all(app=app)

# Xadmin操作方法

# 功能

## 上传文件

1. **前端**
2. **后端**

接收

image = []

for file in files:

if file and allowed\_file(file.name):

fileTime = str(int(time.time() \* 1000))

filename - fileTime + os.path.splitext(file.filename)[1]

file.save(os.path.join(UPLOAD\_Folder, filename))

imagesName.append(filename)

## 错误信息

路由错误：最后写着got an unexpected keyword argument ‘method’ 路由配置允许访问的方法参数错误