# 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,)

from werkzeug.local import localStack //栈元素

from werkzeug.local import local

oo = local() //线程隔离

oo.a = 1

oo.b = b //再不同的线程中修改值

# 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，PasswordField

## 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

from wtform import ValidationError

def validate\_email(self, field): //对email字段做校验

User.query.filter\_by(email = field.data) //field由系统传入

raise ValidationError(“错误信息”)

# 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级模板**

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

蓝图级应用

Blueprint(“web”, templates\_dir, templates\_url\_path)

1. **视图调用：**

**from flask import render\_template, redirect**

def hello(name=None):

return render\_template('hello.html', name=name) #调用模板

return redirect(“/index”) # 路由跳转

## 方向构建url

访问/static/css/index.css文件

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

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

访问web.book视图

<a href=”{{ url\_for(‘web.book’, isbn=book.isbn)}}”></a>

## 自定义模板函数

# 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)}}” />

## 模板语法

{# 注释 #}

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

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

{# 遍历字典 #}

{% for key, item in name.items() %} {% else %} 为空执行 {% endfor %}

## 模板继承

父类模板

{% block head %} <div>this head</div> {% endblock %}

子类模板 -- view视图调用的模板

{% extends ‘layout.html’ %}

{% block head %}

{% super() %} {# 显示父类模板信息 #}

<div>填充到head中</div>

{% endblock %}

## 模板过滤器

{{data.name | default(“默认”) }} 不存在的数显示默认值

{{data.name | default(“”, true)}} 支持检查不存在的数据

{{ data | length() }}

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**

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

static\_dir默认位置是以Flask核心对象位置区别的，配置static\_dir后url路径也会变成文件名例如

static\_dir=”view/aaa” 这里访问路径将变成http://localhost/aaa/image.jpg

static\_url\_path 设置url访问路径,当设置这个，static\_dir将失去效果

static\_url\_path=”ss/cc” <http://localhost/ss/cc,> 访问的物理位置为flask核心对象目录下的SS/cc

**配置蓝图级别静态文件**

Blueprint(‘web’, \_\_name\_\_, static\_folder=””, static\_url\_path=””)

# 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

## 数据库操作函数

def PAFunc1():

## 预处理数据/设置只读

from werkzeug.security import generate\_password\_hash

class User(db.Model):

\_password = Column(“passwd”,String(64), nullable=true)

# 读取

@property

def password(self):

return self.\_password

# 写入--password加密

# 可以限制读取

@property.setter

def password(self, raw):

self.\_password = generate\_password\_hash(raw)

# 验证密码

def check\_passwd(self, raw):

return check\_password\_hash(self.\_password, raw) # 读取并验证密码

## 自动添加时间

class Base(db.Model)

\_\_abstract\_\_ = True

time = Column(“create\_time”, Integer) //数据库中的字段名为create\_time

def \_\_init\_\_(self):

self.time = int(datetime.now().timestamp()) //当前时间戳

## 时间转换

class Base(db.Model):

@property

def create\_time(self):

if self.create\_time:

return datetime.fromtimestamp(self.time)

else:

return None

## flask\_login中的登录验证方法

因为其flask\_login中配置了多种方法，所以要继承

from flask\_login import UserMixin

class User(UserMixin, Base):

... //正常的UserModel

# 如果不是使用id表示用户身份，还要重写get\_id方法

def get\_id(self): //固定的方法名

return self.id

## 生成数据库表

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

# Xadmin操作方法

# 消息闪现

基于session实现，flask的session是保存在客户端，所以需要加密，使用secretyKey

1. 配置secretyKey

配置文件：

SECRET\_KEY = “”

1. view视图

from flask import flash

flash(“这里添加消息”) //产生消息

flash(“xxx”, category=”erry”) //分类消息

1. 模板

{% set meg = get\_flashed\_messages() %}//获取消息

{{meg}} //打印消息

{% set msg = get\_flashed\_messages(category\_filter[‘erry’])%} //获取分类消息

{% with msg = get\_flashed\_message() %}

限制作用域

{% endwith %}

# token登录

## 生成令牌

def

# flask\_login登录登出

pip install flask-login

from flask\_login import LoginManager

login\_mananger = LoginManager()

def create\_app():

login\_mananger.init\_app(app) # 将flask核心对象传入

login\_manager.login\_view = “web.login” //指定用户登录页

login\_manager.login\_message = “请登录” //显示权限登录信息

from flask\_login import login\_user()

@web.route(“/login”, methods=[“GET”,”POST”]

def login():

if request.method == GET:

#第一次登录写入游览器Cookid

return

else:

# 验证用户的账号密码

if ....

# 登录成功的流程

# userModel中需要有一个固定的方法 def get\_id(),这个函数返回用户的id

login\_user(user) //将用户Model传入

# REMEMBER\_COOKIE\_DURATION=10设置cookie有效期

login\_user(user, remeber=True) //记住我

next = request.args.get(“next”)

if not next or not next.startswith(“/”):

next = url\_for(“web.index”)

return redirect(next)

else:

#登录失败--flask-login会自动闪现一条消息

from flask\_login import login\_required

@web.route(“index”, methods=[“GET”])

@login\_required # 添加验证方法

def index():pass

# login\_manager是在create\_app上创建

@login\_manager.user\_loader

def get\_user(uid):

user = User.query.get(int(uid))

return user

## 从cookie中获取用户id

from flask\_login import current\_user

current\_user 是获取的@login\_manager.user\_loader中的方法

gif.uid = current\_user.id

## 验证是否是登录状态

current\_user.is\_authenticated

true: 登录中

## 登出

from flask\_login import logout\_user

@web.route(“logout”, method=[“POST”])

def logout():

logout\_user() //登出

# 功能

## 跳转

url\_for(“web.login”)

redirect()

## 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

## 限制用户访问静态文件

@api.route()

def download(“/download”):

判断用户信息

send\_static\_file() // FLask处理静态文件的方法