# 知乎项目文档

[欢迎来到烧瓶 — 烧瓶文档 （2.3.x） (palletsprojects.com)](https://flask.palletsprojects.com/en/2.3.x/)

[Flask博客开发实战 (lotdoc.cn)](http://lotdoc.cn/blog/detail/170/?menu=6)

# Flask学习

## 实例化app

from flask import Flask

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

app.config['SECRET\_KEY']='ASDFG'   #解决报错后端跨站访问

app.config['JSON\_AS\_ASCII']=False

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

def hello():

return '<h1>hello<h1>'

if \_\_name\_\_=='\_\_main\_\_':

    app.run()

## 路由设置

#<url提取参数>

#'/hi/<int:id>'

#'/hi/<string:id>'接受不包含/的文本

#'/hi/<path:id>'接受包含/的文本

#'/hi/<float:id>'接受正浮点数

#'/hi/1\d{10}'以1开头10位

@app.route('/url/<id>',methods=['GET','POST'])

def url():

    if id=='1':

        return 'url'

    return '<h1>url<h1>'

#自定义转换器

# from werkzeug.routing import BaseConverter

# class RegexConverter(BaseConverter):

#     def \_\_init\_\_(self, url\_map,regex):

#         #调用父类方法

#         super(RegexConverter.self).\_\_init\_\_(url\_map)

#         self.regex=regex

#     def to\_python(self, value: str):

#         return value

#将自定义转换器添加到flask应用中

#app.url\_map\_converters['re']=RegexConverter

#@app.route('/hi/<re("")>',methods=['GET','POST'])

## 模板

https://www.cnblogs.com/xiaxiaoxu/p/10428508.html

#模板

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

def temp():

    data={

        'username':"张三",

        'password':'123',

        'admin':[0,1,2],

    }

    #前端中使用

    # {{data}}

    # {{data.mylist[0]}}

    # {{data.mylist[0]+data.mylist[0]}}

    return render\_template('login.html',data=data)

#过滤器

    #{{"hello" | upper}}

    #{{"hello" | replace('hello','hi')}}

#自定义过滤器

def list\_step(li):

    return li[::2]

#注册过滤器(自定义函数名字，你要用时的名字)

app.add\_template\_filter(list\_step,'li2')

#前端中{{"hello" | li2}}

前端中使用python代码

            {%for items in data\_orders%}

            <tr>

                {% for item in items%}

                <td> {{item}}</td>

                {%endfor%}

                <td><a class="btn btn-danger" data-type="orders" data-id="{{items[0]}}">Delete</a></td>

            </tr>

            {%endfor%}

## 数据库SQLAlchemy

参考链接

[(112条消息) flask\_sqlalchemy常用查询语句总结\_flask\_sqlalchemy查询语句\_Steven灬的博客-CSDN博客](https://blog.csdn.net/weixin_40547993/article/details/105857937)

https://blog.csdn.net/qq\_42265220/article/details/120670267

https://blog.csdn.net/qq\_41603102/article/details/88792295?ops\_request\_misc=&request\_id=&biz\_id=102&utm\_term=flask\_sqlalchemy%E6%95%99%E7%A8%8B&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~sobaiduweb~default-4-88792295.142^v86^wechat,239^v2^insert\_chatgpt&spm=1018.2226.3001.4187

from flask\_sqlalchemy import SQLAlchemy

#flask数据库

class Config:

    SQLALCHEMY\_DATABASE\_URI='mysql://root:123456@127.0.0.1:3306/zhihu'

    SQLALCHEMY\_TRACK\_MODIFICATIONS=True

app.config.from\_object(Config)

#SQLAlchemy和app绑定

db=SQLAlchemy(app)

#创建数据库模型类（表）

class Users(db.Model):

    # Users.\_Users\_\_tablename\_\_-'users'

    用户ID = db.Column(db.Integer, primary\_key=True, nullable=False)

    用户名= db.Column(db.String(45),  nullable=False)

    用户介绍= db.Column(db.String(100),nullable=False)

    def \_\_init\_\_(self,id, username, usersay):

        self.用户ID = id

        self.用户名 = username

        self.用户介绍=usersay

    # 返回函数

    # def \_\_repr\_\_(self):

    #     return '%s' % self.username

#创建所有表db.creat\_all()清除所有表db.drop\_all()

#添加用户

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

def insert():

    #render\_template('hello.html')

    if request.method=='GET':

        return render\_template('insert.html')

    if request.method=='POST':

        #print("POST")

        id=int(request.form.get('用户ID'))

        username=request.form.get('用户名')

        usersay=request.form.get('用户介绍')

        user = Users(id,username,usersay)

        # user.用户ID=int(request.form.get('用户ID'))

        # user.用户名=request.form.get('用户名')

        # user.用户介绍=request.form.get('用户介绍')

        #print(username)

        # 将新创建的用户添加到数据库会话中

        db.session.add(user)

        # 将数据库会话中的变动提交到数据库中, 如果不commit, 数据库中是没有变化的.

        db.session.commit()

        return render\_template('insert.html')

#删除用户

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

def drop():

    if request.method=='POST':

        #print("POST")

        userid=int(request.form.get('删除ID'))

        # 获取用户对象

        user = Users.query.filter\_by(用户ID=userid).first() #查询出id=1的用户

        # 删除用户

        db.session.delete(user)

        #提交数据库会话

        db.session.commit()

    return render\_template('drop.html')

#修改用户信息

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

def upgate():

    if request.method=='POST':

        #print("POST")

        userid=int(request.form.get('userID'))

        username=request.form.get('username')

        usersay=request.form.get('usersay')

        # 获取用户对象

        user = Users.query.filter\_by(用户ID=userid).first()

        # 直接赋值更新数据

        user.用户名 = username

        user.用户介绍=usersay

        #提交数据库会话

        db.session.commit()

    return render\_template('upgate.html')

#查询用户信息

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

def select():

    #render\_template('hello.html')

    if request.method=='GET':

        return render\_template('select.html')

    if request.method=='POST':

        #print("POST")

        id=request.form.get('userID')

        username=request.form.get('username')

        usersay=request.form.get('usersay')

        print(usersay)

        print(username)

        print(id)

        if not id:

            if not username:

                if not usersay:

                    users\_list=Users.query.all()

                    print(users\_list)

                else:

                    users\_list = Users.query.filter(Users.用户介绍==usersay).all()

            else:

                if not usersay:

                    users\_list = Users.query.filter(Users.用户名==username ).all()

                else:

                    users\_list = Users.query.filter(Users.用户名==username ,Users.用户介绍==usersay).all()

        else:

            id=int(id)

            if not username:

                if not usersay:

                    users\_list = Users.query.filter(Users.用户ID==id).all()

                else:

                    users\_list = Users.query.filter(Users.用户ID==id,Users.用户介绍==usersay).all()

            else:

                if not usersay:

                    users\_list = Users.query.filter(Users.用户ID==id,Users.用户名==username ).all()

                else:

                    users\_list = Users.query.filter(Users.用户ID==id,Users.用户名==username ,Users.用户介绍==usersay).all()

        return render\_template('show.html',users\_list=users\_list)

Flask多表查询



[(112条消息) Flask-sqlalchemy高级机制之多表查询\_flask sqlalchemy多表查询\_来一串八二年的葡萄干的博客-CSDN博客](https://blog.csdn.net/li944254211/article/details/109330531)

[(112条消息) 【Flask】sqlalchemy高级用法(注意join的用法)\_flask sqlalchemy join\_敲代码敲到头发茂密的博客-CSDN博客](https://blog.csdn.net/YZL40514131/article/details/123414370)

## 渲染模板

from flask import render\_template

## 重定向

from flask import redirect,url\_for

#redirect重定向,+urt\_for重定向函数，

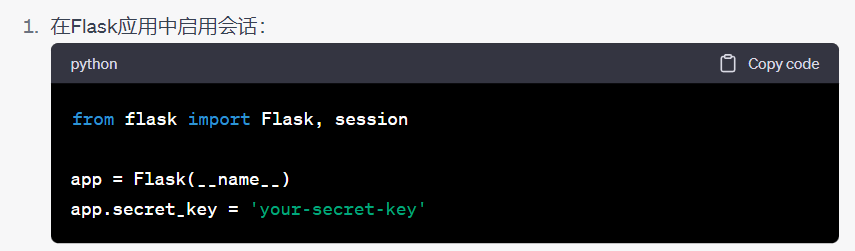
@app.route('/baidu')

def baidu():

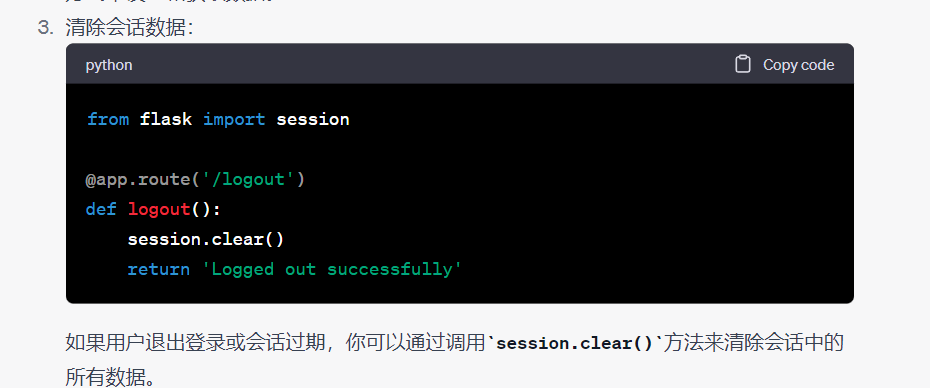
    #return redirect('https://www.baidu.com')

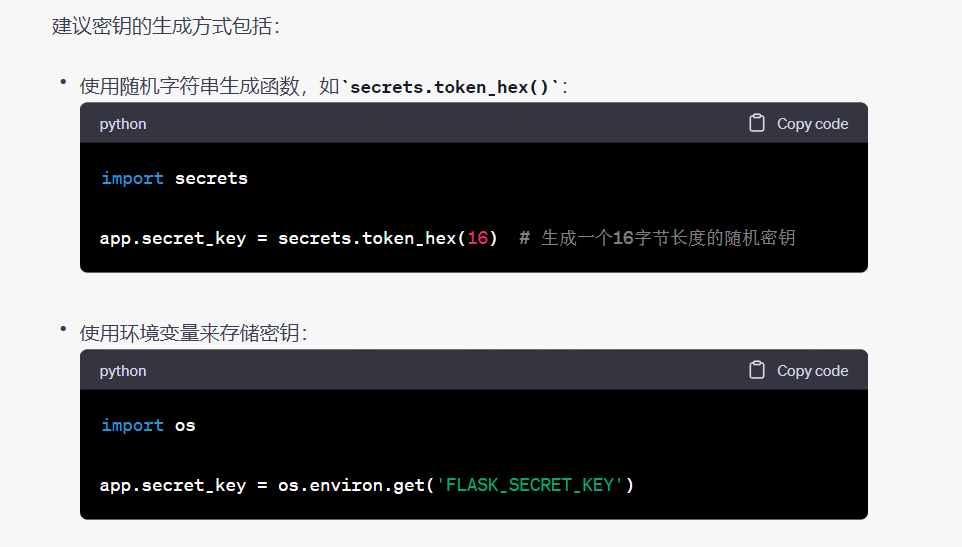
    return redirect(url\_for('login'))

## session









## 异常

#abort抛出异常

from flask import abort

## 获取请求数据

from flask import request

#前端使用form表单发送

#后端使用request获取数据

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

def loginer():

    if request.method=='GET':

        return render\_template('login.html')

    if request.method=='POST':

        username=request.form.get('username')

        password=request.form.get('password')

        print(username)

        if username=='zhangsan' and password=='123':

            return 'login sucess'

        else :

            abort(404)

            return 'this is post'

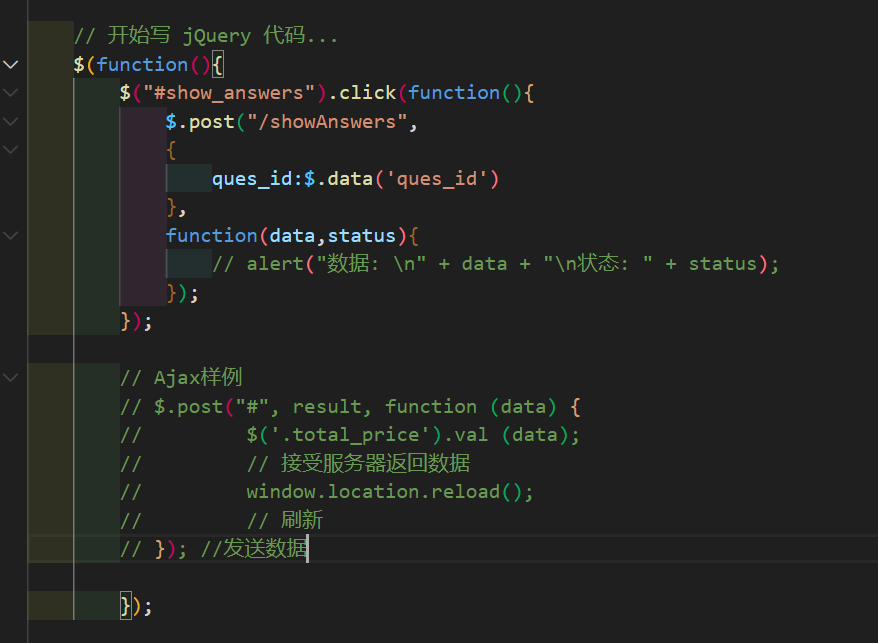
#自定义错误

@app.errorhandler(404)

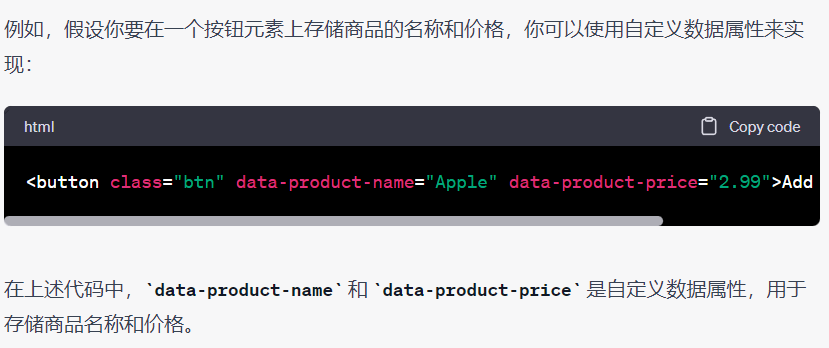
def handle\_404\_error(err):

    return '出现了404错误，错误信息是s%'%err

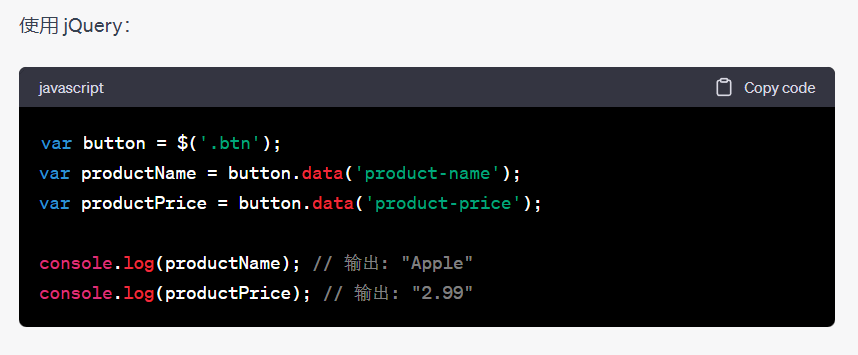
## Ajax



自定义数据属性（结合模板）







[Jquery使用ajax与Flask后端进行数据交互 - HALUK - 博客园 (cnblogs.com)](https://www.cnblogs.com/haluk/p/12014402.html)



## 传递数据给前端

#make\_response把数据传给前端,json,jsonify将字典转为json

from flask import make\_response,json,jsonify

#json数据

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

def json():

    data={

        'name':"张三"

    }

    #转为json(1)

    response= make\_response(json.dumps(data,ensure\_ascii=False))#中文

    response.mimetype='application/json'

    #转为json(2)

    #前面加上一行app.config['JSON\_AS\_ASCII']=False

    response=jsonify(data)

    return response

## form表单

from wtforms import StringField,PasswordField,SubmitField  #类型

from flask\_wtf import FlaskForm

from wtforms.validators import DataRequired,EqualTo #验证数据不能为空，验证数据是否相同

#等价于前端的form表单

#定义表单模型类

class Register(FlaskForm):

    username=StringField(label='用户名',validators=[DataRequired('用户名不能为空')])

    password=StringField(label='密码',validators=[DataRequired('密码不能为空')])

    password2=StringField(label='再次输入密码',validators=[DataRequired('密码不能为空'),EqualTo('password')])

    sunmit=SubmitField(label='提交')

#form表单

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

def register():

    #创建表单对象

    form=Register()

    if request.method=='GET':

        return render\_template('register.html',form=form)

    else:

        #验证表单传过来的数据都是正确的

        if form.validate\_on\_submit():

            username=form.username.data

            password=form.password.data

        return render\_template('register.html',form=form)

# 实战

创建虚拟环境

python -m venv venv

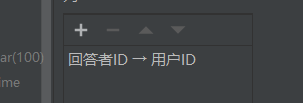
venv\Scripts\activate激活打开

pip freeze > req.txt收集依赖

[Flask博客开发实战 (lotdoc.cn)](http://lotdoc.cn/blog/detail/170/?menu=6)

# 数据库设计：

（1）设计数据库时，先设置自增再设置外键



1. 先删除数据，再设置外键

# 博客网站优化

[(112条消息) 博客系统（页面设计）\_个人博客前端页面\_'Dream是普通小孩耶的博客-CSDN博客](https://blog.csdn.net/weixin_54150521/article/details/128185533?ops_request_misc=%7B%22request%5Fid%22%3A%22168656538516800180693928%22%2C%22scm%22%3A%2220140713.130102334..%22%7D&request_id=168656538516800180693928&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-128185533-null-null.142^v88^control_2,239^v2^insert_chatgpt&utm_term=%E4%B8%AA%E4%BA%BA%E5%8D%9A%E5%AE%A2%E5%89%8D%E7%AB%AF%E8%AE%BE%E8%AE%A1&spm=1018.2226.3001.4187)

[Bulma中文文档,开始（start）,使用Bulma仅仅需要引入1 个CSS 文件 (bulmacss.cn)](https://www.bulmacss.cn/doc/read/62.html" \l ":~:text=%E5%BC%80%E5%A7%8B%EF%BC%88start%EF%BC%89 1 %E4%B8%80%E3%80%81%E7%9B%B4%E6%8E%A5%E5%BC%95%E7%94%A8%E5%B7%B2%E7%BC%96%E8%AF%91%E5%A5%BD%E7%9A%84css%E6%96%87%E4%BB%B6 %23 2 %E4%BA%8C%E3%80%81%E8%8E%B7%E5%8F%96Bulma Sass%E6%96%87%E4%BB%B6%E8%87%AA%E5%AE%9A%E4%B9%89%E7%BC%96%E8%AF%91%E5%90%8E%E4%BD%BF%E7%94%A8 %23,4 %E5%9B%9B%E3%80%81 %E5%BC%80%E5%A7%8B%E6%A8%A1%E6%9D%BF %23 5 %E4%BA%94%E3%80%81 bulma-start %23)

[Bulma CSS使用示例：仿知乎页面 - 简书 (jianshu.com)](https://www.jianshu.com/p/c8ce8e790f7c)

[查找具有完美外观和感觉的图标 |字体真棒 (fontawesome.com)](https://fontawesome.com/search?m=free&o=r)

[HTML5如何使用SVG - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/54088196)

[Font Awesome 的使用\_css\_期待mizi-DevPress官方社区 (csdn.net)](https://huaweicloud.csdn.net/638f14c5dacf622b8df8f064.html?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2~default~CTRLIST~activity-1-120147866-blog-79414167.235^v38^pc_relevant_anti_vip_base&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2~default~CTRLIST~activity-1-120147866-blog-79414167.235^v38^pc_relevant_anti_vip_base&utm_relevant_index=2)

登录

首页

文章（分页展示，详情界面，点赞），问题（分页展示，详情界面，回答）

个人信息

个人信息（修改）

我的文章（增加，删除，修改，展示），我的问题（增加，删除，修改，展示），我的回答（增加，删除，修改，展示）

管理员

用户，文章，问题，回答

感觉不应该好高骛远，应该先把简单的给处理懂了，再弄麻烦的

别着急，慢慢弄，弄不懂放弃换简单的就好了

追加文本

jQuery append() 方法

从一个页面往下一个页面传数据