一、 Flask 概述

1. 什么是Flask

1) Flask 介绍

Flask是一个基于Python并且依赖于Jinja2模板引擎和Werkzeug WSGI 服务的一个微型框架
WSGI ：Web Server Gateway Interface(WEB服务网关接口)，定义了使用python编写的web app与web server之间接口格式

2) Flask 的框架模式 - MTV

1. 经典三层结构 ：MVC模式
    - M ：Models ，模型层，负责数据库建模
    - V ：Views，视图层，用于处理用户显示的内容，如 :html
    - C ：Controller，控制器，处理与用户交互的部分内容。处理用户的请求并给出响应
2. python常用：MTV模式
    - M ：Models ，模型层，负责数据库建模
    - T ：Templates ，模板层，用于处理用户显示的内容，如：html
    - V ：Views ，视图层，处理与用户交互的部分内容。处理用户的请求并给出响应

2. 准备工作

1. 安装 flask

```
sudo pip3 install flask
```

2. 查看flask版本
    - 进入python3交互模式 ：

```
>>>import flask
>>>flask.__version__
```

- 注意 ：不同版本之间会有细微差异，尽量以教学环境为主

3. 初始化flask应用

```
from flask import Flask
# 将当前运行得到主程序构建成Flask的应用，以便接收用户的请求(request),并给出响应(response)
app = Flask(__name__)

@app.route('/')
def index():
    return "<h1>this is my first flask app</h1>"

# 运行Flask应用
if __name__ == '__main__':
    app.run(debug=True)
```

- @app.route(): Flask中的路由定义，定义用户的访问路径, / 表示的是整个网站的根路径
- def index(): 表示匹配上@app.route()路径后的处理程序-视图函数，该类函数必须要有return，return后要给一个字符串 或 响应对象
- 运行应用后会启动flask自带的小型服务器，默认在本机开启的端口号是5000
- debug=True,是将当前的启动模式改为调试模式(开发环境中推荐使用调试模式,生产环境中不允许使用)

二、Flask 使用

1. Flask-路由(route)

1) 什么是路由

- 客户端将请求发送给web服务器，web服务器再将请求发送给flask程序实例
- 程序实例需要知道每个url请求要运行哪些代码，所以需要建立一个 url 到 python 函数的映射

路由就是处理url和python函数之间的关系的程序
在Flask中，路由是通过 @app.route 装饰器来表示的

2) 路由的使用

1. 基本使用方式

```
@app.route('/')
def index():
    return 'This is index page.'

@app.route('/login')
def login():
    return 'This is login page.'
```

2. 带参数的路由

- 基本带参路由

```
@app.route('/show/<name>')
def show1(name):
    # 在函数中 name 表示的就是地址栏上传递过来的数据
    return 'xxx'
```

- 带多个参数的路由

```
@app.route('/show2/<name>/<age>')
def show1(name,age):
    # 在函数中 name 表示的就是地址栏上传递过来的数据
    return 'xxx'
```

- 指定参数类型的路由

```
@app.route('/show3/<name>/<int:age>')
def show1(name,age):
    # 在函数中 name 表示的就是地址栏上传递过来的数据
    return 'xxx'
```

3. 多 URL 的路由匹配
   - 允许在一个视图处理函数中设置多个url路由规则

```
@app.route('/')
@app.route('/index')
def index():
    return "xxx"
```

4. 路由中设置 HTTP 请求方法
   - Flask路由规则也允许设置对应的请求方法，只有将匹配上请求方法的路径交给视图处理函数去执行
   - 如果没有指定请求方法，默认允许GET请求

```
#只有post请求方式允许访问 localhost:5000/post
@app.route('/post',methods=['POST'])
def post():
    return 'xxxx'
```

3) 反向解析

- 正向解析：程序自动解析，根据@app.route()中的访问路径来匹配处理函数
- 反向解析：通过视图处理函数的名称自动生成视图处理函数的访问路径
  - Flask 中提供了 url_for() 函数，用于反向解析url

```
@app.route('/')
def index():
      return "Index"

@app.route('/show/<name>')
def show(name):
    return "name:%s" % name
```

1. url_for('index') : 结果为 ：/
2. url_for('show',name='qtx') : 结果为 ：/show/qtx

2、Flask模板（templates）

1) 什么是模板

1. 模板是一个包含响应文本的文件(通常是html文件)
2. 模板中允许包含"占位变量"来表示动态的内容，"占位变量"最终会被真实的值所替换
3. 模板最终也会被解析成相应的字符串，这一过程称为"渲染"

2) 模板的设置

1. 默认模板目录
   默认情况下，Flask会在程序文件夹中的 templates 子文件夹中寻找模板
   注意：需要手动创建 templates 文件夹
2. 自定义模板文件的目录
   可以修改配置，为template_folder属性指定一个文件名字符串

```
app = Flask(__name__,template_folder='templates') # 配置模板文件的文件夹
```

locals()可以在函数中把变量赋值转化成字典。

2) 模板的语法

占位变量 {{变量名}}

注释　　{#要注释的内容#}

标签　　{%标签名%}

　　{%结束标签(end+标签名)%}

　　{%for 变量名 in 可迭代对象%}

　　{%endfor %}

```
@app.route('/01-exer')
def exer01_view():
    title = '《钢铁是咋炼成的》'
    author = '奥斯特洛夫斯基'
    price = 56
    pub = '北京大学出版社'
    #locals将局部变量快速的变成字典
    #locals()将局部变量变成字典  **data将字典打散成关键字参数
    data = locals()
    return render_template('exercise01.html',**data)
    # return render_template('exercise01.html',
    #                        title='《钢铁是咋炼成的》',
    #                        author='奥斯特洛夫斯基',
    #                        price=56,
    #                        pub='北京大学出版社')
```

```
from flask import Flask,render_template,request
```

```
app=Flask(__name__,template_folder="t")

@app.route('/user/login')
def index_view():
    uname=request.args.get("uname")
    pwd=request.args.get("pwd")
    locals()
    return render_template("login.html",**locals())


if __name__=="__main__":
    app.run()
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>login.html</title>
</head>
<body>
    {% if uname:%}
    <h1>欢迎您：{{uname}}</h1>
    {%else:%}
    <form action="/user/login">
        <div>
            姓名 <input type="text" name="uname">
        </div>
        <div>
            密码 <input type="text" name="pwd">
        </div>
        <input type="submit">
    </form>
    {% endif %}
</body>
</html>
```

```
#run01.py

from flask import Flask,url_for

#将当前的模块构建成flask应用
#当flask应用构建完成后就可以接收请求并给出响应
app = Flask(__name__)

@app.route('/')#匹配当前服务器的根路径
@app.route('/index')
@app.route('/1')
@app.route('/<int:page>')
def index_view(page=None):
    if page == None:
        return '这是第一个flask应用的首页'
    else:
        return '这是第一个flask应用的第%s页' % page
```

```
#如果访问127.0.0.1:5000/   或  /index 或 /1
#显示 这是第一个flask应用的首页
#如果访问127.0.0.1:5000/任意数字
#显示 这是第一个flask应用的第x页




# http://127.0.0.1:5000/mil
@app.route('/mil')
def mil_view():
    return '这是军事相关的页面'


#127.0.0.1:5000/show/shibw/25
@app.route('/show/<name>/<int:age>')
def show_view(name,age):
    # return '欢迎%s'  % name
    print(type(name))#str
    print(type(age))#int
    #url_for(视图函数名,关键字形式的参数)
    myurl = url_for('show_view',name='QTX',age=25)
    return '当前函数的访问路径是%s' % myurl
    # return '姓名:%s,年龄%s' % (name,age)

#访问地址 127.0.0.1:5000/birthday/2020/1/1
#在页面显示 您的生日为 2020年1月1日
@app.route('/birthday/<int:year>/<int:month>/<int:day>')
def birth_view(year,month,day):
    return '您的生日为%s年%s月%s日' % (year,month,day)

if __name__  == '__main__':
    # http://127.0.0.1:5000/
    #                host          port
    # run(host=None,port=None,debug=None)
    app.run(debug=True)
```

```
#run02.py

from flask import Flask,url_for

#将当前的模块构建成flask应用
#当flask应用构建完成后就可以接收请求并给出响应
app = Flask(__name__)

@app.route('/')#匹配当前服务器的根路径
@app.route('/index')
@app.route('/1')
# ValueError: urls must start with a leading slash
#路由地址没有以 / 开头
@app.route('/<int:page>')
def index_view(page=None):
    if page == None:
        return '这是第一个flask应用的首页'
```

```
    else:
        return '这是第一个flask应用的第%s页' % page
#如果访问127.0.0.1:5000/   或  /index 或 /1
#显示 这是第一个flask应用的首页
#如果访问127.0.0.1:5000/任意数字
#显示 这是第一个flask应用的第x页

# AssertionError: View function mapping
# is overwriting an existing endpoint function: index_view
#index_view视图函数重名了   相同名字的视图函数只能出现一次

# TypeError: The view function did not return a valid response.
# The function
# either returned None or ended without a return statement.
#视图函数没有返回有效的响应
#如果不写return   返回值默认为None  None不是一个有效的响应
#1.1.1
# @app.route('/qtx')
# def noReturn_view():
#     print('这是第二个视图函数')
    # return '这是第二个视图函数'


# http://127.0.0.1:5000/mil
@app.route('/mil')
def mil_view():
    return '这是军事相关的页面'


#127.0.0.1:5000/show/shibw/25
@app.route('/show/<name>/<int:age>')
def show_view(name,age):
    # return '欢迎%s'  % name
    print(type(name))#str
    print(type(age))#int
    #url_for(视图函数名,关键字形式的参数)
    myurl = url_for('show_view',name='QTX',age=25)
    return '当前函数的访问路径是%s' % myurl
    # return '姓名:%s,年龄%s' % (name,age)

#访问地址 127.0.0.1:5000/birthday/2020/1/1
#在页面显示 您的生日为 2020年1月1日
@app.route('/birthday/<int:year>/<int:month>/<int:day>')
def birth_view(year,month,day):
    return '您的生日为%s年%s月%s日' % (year,month,day)




if __name__  == '__main__':
    # http://127.0.0.1:5000/
    #                host          port
    # run(host=None,port=None,debug=None)
    app.run(debug=True)
```

```python
# run03.py
from flask import Flask,render_template,request

#修改默认的模板目录位置
#templates   -->  t
app = Flask(__name__,template_folder='t')

# jinja2.exceptions.TemplateNotFound: index.html
@app.route('/')
def index_view():
    #通过render_template函数 渲染模板
    name = 'QTX'
    age = 25
    email = 'qtx@tedu.cn'
    address = 'xxxx'
    phone = '12345678911'

    mylist = [
        {'id':1,'content':'我爱学习'},
        {'id':2,'content':'学习使我快乐'},
        {'id':3,'content':'沉迷学习,日渐发胖'},
        {'id':4,'content':'以梦为马,越骑越傻'}
    ]


    #locals 收集所有的局部变量  保存在字典中
    dic = locals()
    return render_template('index.html',**dic)
    # dic = {
    #     'name':'QTX',
    #     'age':25,
    #     'email':'qtx@tedu.cn',
    #     'address':'xxxx',
    #     'phone':'12345679811'
    # }
    # return render_template('index.html',name='QTX',age=25)


#当用户访问/user/login  能看到login.html
@app.route('/user/login')
def login_view():
    #from flask import request
    #request.args保存前端get请求提交的数据
    print(request.args)
    # uname = request.args['uname']
    #get(key,default)  获取字典对应键的值
    #如果键不存在 返回default
    uname = request.args.get('uname')
    pwd = request.args.get('pwd')
    print(uname,pwd)
    return render_template('login.html',**locals())

# werkzeug.exceptions.BadRequestKeyError:
# 400 Bad Request: The browser (or proxy)
#  sent a request that this server could not
#   understand.
# KeyError: 'upwd'
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <h1>这是第一个flask应用的首页</h1>
    {#   模板中的注释
    {{context}}
    <div>姓名:{{context['name']}}</div>
    <div>年龄:{{context['age']}}</div>
    #}
    <div>姓名:{{name}}</div>
    <div>年龄:{{age}}</div>
    <hr>
    <table>
        <tr>
            <th>id</th>
            <th>content</th>
        </tr>
        {% for item in mylist %}
        <tr>
            <td>{{item.id}}</td>
            <td>{{item.content}}</td>
        </tr>
        {% endfor %}
    </table>
</body>
</html>
```

login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>登录页</title>
</head>
<body>
    <!-- if uname:
         欢迎xxx
    else:
        表单 -->
        {% if uname %}
```

```
        <h1>欢迎您: {{uname}}</h1>
        {% else %}
        <form action="/user/login">
            <div>
                姓名  <input type="text" name="uname">
            </div>
            <div>
                密码  <input type="password" name="pwd">
            </div>
            <input type="submit">
        </form>
       {% endif %}
</body>
</html>
```

```
```