Flask 使用的是jinja2模板

1. 模板的使用
2. 创建模板，在templates文件夹下面创建一个index.html
3. 渲染模板，在py文件中定义一些变量，在利用render\_template(‘index.html’,变量=变量)

如

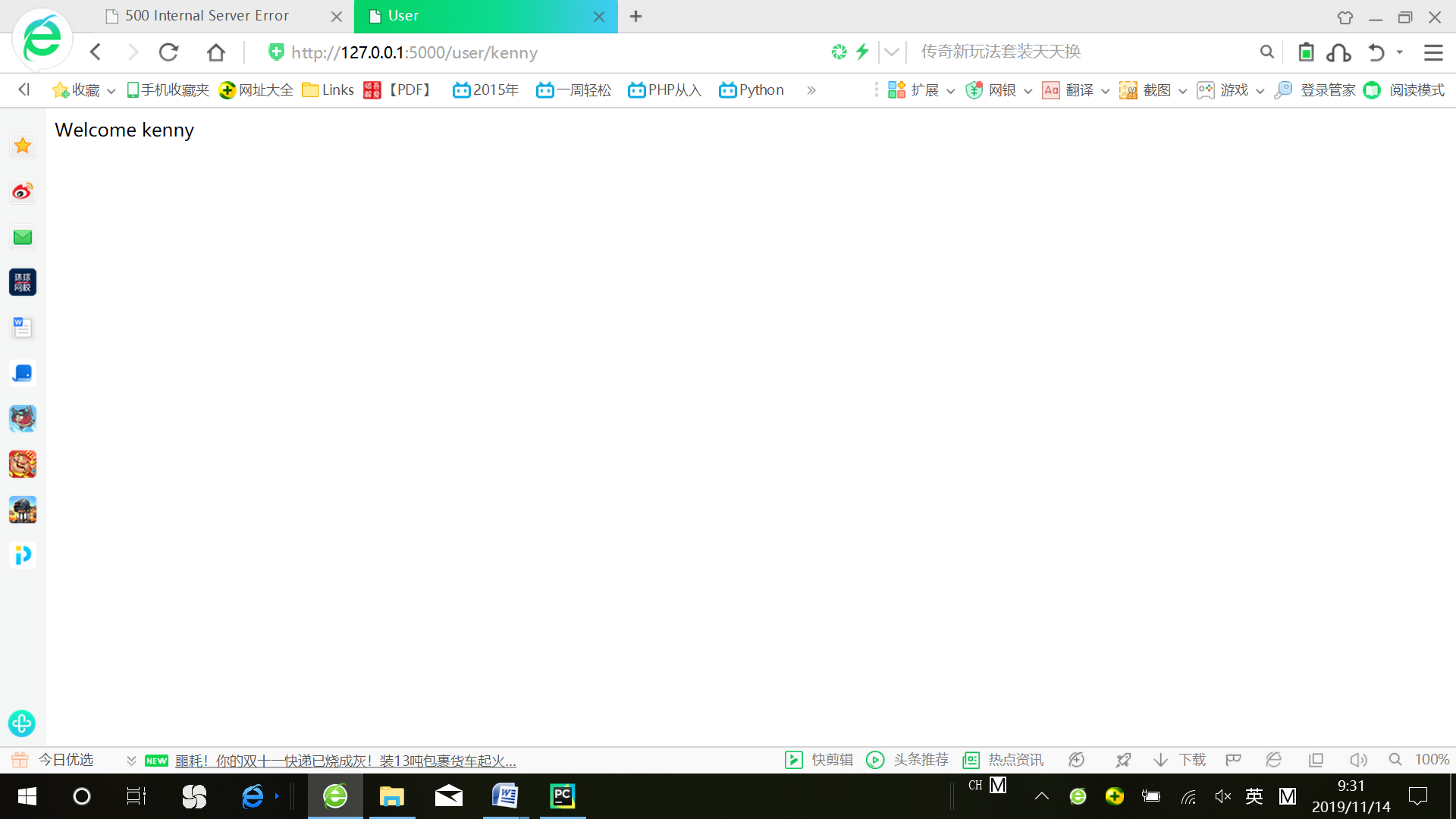
Manage.py

**from** flask **import** Flask,render\_template  
**from** flask\_script **import** Manager  
  
app=Flask(\_\_name\_\_)  
manager=Manager(app)  
  
@app.route(**'/user/<username>'**)  
**def** user(username):  
 **return** render\_template(**'user.html'**,username=username)  
**if** \_\_name\_\_==**'\_\_main\_\_'**:  
 manager.run()

user.html

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>User</**title**>  
</**head**>  
<**body**>  
 Welcome {{username}}  
</**body**>  
</**html**>

结果：



1. 可以使用全局g变量

Manage.py

**from** flask **import** Flask,render\_template,g  
**from** flask\_script **import** Manager  
  
app=Flask(\_\_name\_\_)  
manager=Manager(app)  
  
@app.route(**'/user/<username>'**)  
**def** user(username):  
 *# return render\_template('user.html',username=username)  
#使用全局g变量* g.name=username  
 **return** render\_template(**'user.html'**)  
**if** \_\_name\_\_==**'\_\_main\_\_'**:  
 manager.run()

user.html

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>User</**title**>  
</**head**>  
<**body**>  
 Welcome {{g.name}}  
</**body**>  
</**html**>

1. 使用函数(过滤器)

<!DOCTYPE **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>User</**title**>  
</**head**>  
<**body**>  
 Welcome {{g.name|capitalize}} # capitalize函数的作用是将首字母大写  
</**body**>  
</**html**>

常用过滤器

**Flask框架(flask模板（jinja2）与字符串过滤器、列表过滤器以及自定义过滤器)**

**模板变量 : {{name}}**

**1. 模板的定义： 模板变量 : {{name}}**

**2. 模板的渲染: render\_template("模板名字",键=值,键=值)**

**例如： render\_template("index.html",name="python",age="18")**

**如果传值的话，想要直接在渲染函数里面直接传字典则：**

**可以事先定义好一个字典data,然后，render\_template("index.html",\*\*data)**

**首先是模板，里面的变量的写法和python的是一样的， 如下面的 index.html (放在自建的templates目录下面)**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<title>Title</title>**

**</head>**

**<body>**

**<p>name = {{ name }}</p>**

**<p>age = {{ age }}</p>**

**<p>my\_dict: city={{ my\_dict["city"] }}</p>**

**<p>my\_dict: city={{ my\_dict.city}}</p>**

**<p>my\_list : {{ my\_list }}</p>**

**<p>my\_list[my\_int] : {{my\_list[my\_int] }}</p>**

**<p>my\_list[0] + my\_list[1] : {{ my\_list[0] + my\_list[1] }}</p>**

**</body>**

**</html>**

**然后在程序中怎么进行渲染**

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

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

**@app.route("/index")**

**def index():**

**data = {**

**"name": "python",**

**"age": 18,**

**"my\_dict": {"city": "sz"},**

**"my\_list": [1, 2, 3, 4, 5],**

**"my\_int": 0**

**}**

**# render\_template("index.html",name="python",age="18")**

**"""也可以进行直接传递一个字典进去，但是字典的前面要加上两个星号"""**

**return render\_template("index.html", \*\*data)**

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

**app.run(debug=True)**

**运行的结果，就会按照模板里面取里面的值的方式，取出相应的值， 输入127.0.0.1:5000/index**

**过滤器**

1. **字符串过滤器**
2. **safe：禁用转义；**

**<p>{{ '<em>hello</em>' | safe }}</p> 结果：**hello world

**capitalize：把变量值的首字母转成大写，其余字母转小写；**

**<p>{{ 'hello' | capitalize }}</p> 结果：**Hello

**lower：把值转成小写；**

**<p>{{ 'HELLO' | lower }}</p> 结果：**hello

**upper：把值转成大写；**

**<p>{{ 'hello' | upper }}</p> 结果：**HELLO

**title：把值中的每个单词的首字母都转成大写；**

**<p>{{ 'hello pretty girls,i love you' | title }}</p> 结果：**Hello Pretty Girls,i Love You

**trim：把值的首尾空格去掉；**

**<p>{{ ' hello world ' | trim }}</p> 结果：hello world（中间的空格不会被去掉）**

**reverse:字符串反转；**

**<p>{{ 'hello' | reverse }}</p> 结果：olleh**

**format:格式化输出；**

**<p>{{ '%s is %s，and %s is %d' | format('name','Jaquline','age',17) }}</p> 结果：**name is Jaquline，and age is 17

**striptags：渲染之前把值中所有的HTML标签都删掉；**

**<p>{{ '<em>hello</em>' | striptags }}</p> 结果：hello**

**2.支持链式使用过滤器**

**<p>{{ “ hello world “ | trim | upper }}</p> 结果：**HELLO WORLD

**3.列表过滤器**

**first：取第一个元素**

**<p>{{ [1,2,3,4,5,6] | first }}</p> 结果：1**

**last：取最后一个元素**

**<p>{{ [1,2,3,4,5,6] | last }}</p> 结果：6**

**length：获取列表长度**

**<p>{{ [1,2,3,4,5,6] | length }}</p> 结果：6**

**sum：列表求和**

**<p>{{ [1,2,3,4,5,6] | sum }}</p> 结果：21**

**sort：列表排序**

**<p>{{ [6,2,3,1,5,4] | sort }}</p> 结果：**[1, 2, 3, 4, 5, 6]

**自定义的过滤器**

**自定义的过滤器名称如果和内置的过滤器重名，会覆盖内置的过滤器。**

**方式一：**

**通过 add\_template\_filter (过滤器函数, 模板中使用的过滤器名字)**

**def list\_step\_2(li):**

**"""自定义的过滤器"""**

**return li[::2] # 让列表隔一个取一个**

**"""自定义的过滤器要进行注册"""**

**# 第一个参数是函数名，第二个是过滤器名字**

**app.add\_template\_filter(list\_step\_2,"li2")**

**方式二：**

**通过装饰器 app.template\_filter (模板中使用的装饰器名字)**

**@app.add\_template\_filter("li")**

**def list\_step\_2(li):**

**"""自定义的过滤器"""**

**return li[::2] # 让列表隔一个取一个**

**在模板中直接接收：**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<title>Title</title>**

**</head>**

**<body>**

**<p>{{my\_list | li2}}</p>**

**</body>**

**</html>**

1. 使用宏
2. 在一个文件在定义宏如myMacro.html

{%macro sayHello()%}  
 <**h1**>Hello world</**h1**>  
 {%endmacro%}

1. 在另外一个文件中使用宏如index.html
2. 导入宏

{%from 'myMacro.html' import sayHello%}

1. 使用宏

{{sayHello()}}

1. 模板继承

Mybase.html 父模板

<**html**>  
<**head**>  
 {% block head %}  
 <**meta charset="UTF-8"**>  
 <**meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0"**>  
 <**meta http-equiv="X-UA-Compatible" content="ie=edge"**>  
 <**title**>{% block title %}Document{%endblock%}</**title**>  
 {%endblock%}  
</**head**>  
<**body**>  
 {%block body %}  
 <**h1**>Base html content</**h1**>  
 {%endblock%}  
</**body**>  
</**html**>

子模板

Child.html

{%extends 'mybase.html'%}  
{%block title%}Child html{%endblock%}  
{%block body%}  
 <**table border="1" cellspacing="0"**>  
 {%for i in range(5)%}  
 <**tr**>  
 <**td**>cell value:{{i+1}}</**td**><**td**>cell value:{{i+2}}</**td**><**td**>cell value:{{i+3}}</**td**>  
 </**tr**>  
 {%endfor%}  
 </**table**>  
{%endblock%}