小白学Flask第九天| 看看模板的那些事（一）

原创 JAP君 Python进击者
2019-09-17原文

本文内容：

1. 模板
2. 过滤器
3. 引入表单的拓展
4. 使用表单接受并检验参数

模板

在Flask当中的模板被称为Jinja2模板，那么我们怎么去使用模板呢？大家可以看到下面两块代码：

```
# -*- coding: utf-8 -*-

from flask import Flask, render_template
```

```python
app = Flask(__name__)


@app.route("/index")
def index():
    return render_template("index.html", name="kuls", age=18)


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

在templates当中创建的index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <p>name = {{name}}</p>
    <p>age = {{age}}</p>
</body>
</html>
```

我们运行代码：

name = kuls

age = 18

从上面可以知道在Flask当中模板变量为 **{{ 变量名 }}**，模板渲染使用**render_template()**函数。

在代码中还能发现我们在render_template()是通过**键值对**的形式来给模板变量赋值，那么我们还能通过其他形式来进行传参吗？当然是可以的，我们可以通过**字典的形式**来进行传参。

```
# -*- coding: utf-8 -*-
from flask import Flask, render_template


app = Flask(__name__)


@app.route("/index")
def index():
    data = {
        "name":"kuls",
        "age": 18
    }
    return render_template("index.html", **data)
```

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

那还有什么骚操作没？看下面：

```
# -*- coding: utf-8 -*-
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/index")
def index():
    data = {
        "name":"kuls",
        "age": 18,
        "dict":{"city": "cs"},
        "list":[0, 1, 2, 3],
        "int":1
    }
    return render_template("index.html", **data)

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

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <p>name = {{name}}</p>
    <p>age = {{age}}</p>
    <p>dict city= {{ dict["city"] }}</p>
    <p>dict city= {{ dict.city }}</p>
    <p>list : {{ list }}</p>
    <p>list[int]: {{ list[int] }}</p>
    <p>list[1]+list[2]: {{ list[1]+list[2] }}</p>
</body>
</html>
```

运行一下：

可以看到我们可以在模板上进行运算以及对列表或者字典取值。

过滤器

字符串过滤器：

safe：禁用转义；

```
<p>{{ '<em>hello</em>' | safe }}</p>
```

capitalize：把变量值的首字母转成大写，其余字母转小写；

```
<p>{{ 'hello' | capitalize }}</p>
```

lower：把值转成小写；

```
<p>{{ 'HELLO' | lower }}</p>
```

upper：把值转成大写；

```
<p>{{ 'hello' | upper }}</p>
```

title：把值中的每个单词的首字母都转成大写；

```
<p>{{ 'hello' | title }}</p>
```

trim：把值的首尾空格去掉；

```
<p>{{ ' hello world ' | trim }}</p>
```

reverse:字符串反转；

```
<p>{{ 'olleh' | reverse }}</p>
```

format:格式化输出；

```
<p>{{ '%s is %d' | format('name',17) }}</p>
```

striptags：渲染之前把值中所有的HTML标签都删掉；

```
<p>{{ '<em>hello</em>' | striptags }}</p>
```

支持链式使用过滤器：

```
<p>{{ “ hello world  “ | trim | upper }}</p>
```

列表过滤器：

first：取第一个元素

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

last：取最后一个元素

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

length：获取列表长度

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

sum：列表求和

```
<p>{{ [1,2,3,4,5,6] | sum }}</p>
```

sort：列表排序

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

自定义过滤器：

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 filter_double_sort(ls):
    return ls[::2]
app.add_template_filter(filter_double_sort,'double_2')
```

方式二：

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

```
@app.template_filter('db3')
```

```
def filter_double_sort(ls):
    return ls[::-3]
```

引入表单的拓展

使用Flask-WTF表单扩展，可以帮助进行CSRF验证，帮助我们快速定义表单模板，而且可以帮助我们在视图中验证表的数据。

```
 pip install Flask-WTF
```

我们来看一下，没使用表单拓展的时候是怎么去写表单的：

```
#模板文件
<form method='post'>
    <input type="text" name="username" placeholder='Username'>
    <input type="password" name="password" placeholder='password'>
    <input type="submit">
</form>
```

```
from flask import Flask,render_template,request

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

```
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        print username,password
      return “success”
  else:
    return render_template(“login.html”)
```

看着上面是不是有点苦逼，这么英俊潇洒的你怎么能这么狼狈的写代码呢？

使用表单接受并检验参数

我们接着来看一下使用了Flask-WTF拓展之后：

模板页：

```
<form method="post">
        #设置csrf_token
        {{ form.csrf_token() }}
        {{ form.us.label }}
        <p>{{ form.us }}</p>
        {{ form.ps.label }}
        <p>{{ form.ps }}</p>
```

```
        {{ form.ps2.label }}
        <p>{{ form.ps2 }}</p>
        <p>{{ form.submit() }}</p>
        {% for x in get_flashed_messages() %}
            {{ x }}
        {% endfor %}
 </form>
```

视图函数：

```
#coding=utf-8
from flask import Flask,render_template,
redirect,url_for,session,request,flash


#导入wtf扩展的表单类
from flask_wtf import FlaskForm
#导入自定义表单需要的字段
from wtforms import SubmitField,StringField,PasswordField
#导入wtf扩展提供的表单验证器
from wtforms.validators import DataRequired,EqualTo
app = Flask(__name__)
# 需要设置 SECRET_KEY 的配置参数
app.config['SECRET_KEY']='1'


#创建自定义表单类，文本字段、密码字段、提交按钮
class Login(FlaskForm):
```

```
    us =
StringField(label=u'用户：',validators=[DataRequired()])
    ps =
PasswordField(label=u'密码',validators=[DataRequired(),EqualTo('
ps2','err')])
    ps2 =
PasswordField(label=u'确认密码',validators=[DataRequired()])
    submit = SubmitField(u'提交')


#定义根路由视图函数，生成表单对象，获取表单数据，进行表单数据验证
@app.route('/',methods=['GET','POST'])
def index():
    # 创建一个Login对象
    form = Login()
    if form.validate_on_submit():
        # 调用Login对象当中的属性，并取其数值
        name = form.us.data
        pswd = form.ps.data
        pswd2 = form.ps2.data
        print(name,pswd,pswd2)
        # 重定向至login的装饰器
        return redirect(url_for('login'))
    else:
        if request.method=='POST':
flash(u'信息有误，请重新输入！')
```

```
    return render_template('index.html',form=form)
if __name__ == '__main__':
    app.run(debug=True)
```

这样一写感觉整个的逼格就提升了好几个档次。在表单拓展当中需要注意必须得设置SECRET_KEY的值，这个值随便你设置为多少（在前面的session当中我们也提到过SECRET_KEY）

在上面的代码当中，我把需要注释的地方全部都注释了，大家可以仔细去阅读一些注释。由于篇幅有限，这里不对Flask-WTF的一些具体用法做阐述。

精选留言

暂无...