登录

2019年04月19日 阅读 204

关注

Flask框架从入门到精通之模板初识（五）

> 知识点：1、模板 2、变量 3、标签 4、注释 5、过滤器 6、自定义过滤器

一、概况

我们目前在视图函数函数中返回的都一段普通的字符串，一张网页需要用到html、css、js等标签，才能展现的更漂亮。所以目前咱们只返回字符串还达不到，那么我们就需要模板。模板其实就已经写好的html、css、js，你只需要往里面填充要展示的内容即可。

二、模板

在我们项目的目录下新建一个名为 `templates` 的文件夹，并在此文件夹新建一个名为 `index.html` 的html文件。内容如下：

复制代码

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

<h1>我们是模板-Hello</h1>

</body>
</html>
```

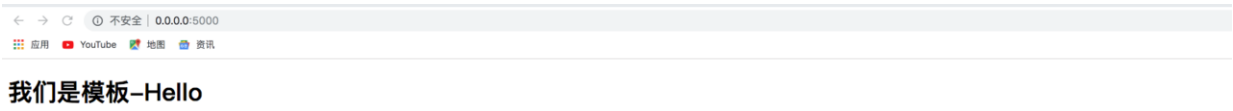
模板我们已经写好，在视图需要引用 `render_template` 函数来使用。

复制代码

```
from flask import Flask
from flask import render_template

app = Flask(__name__)


@app.route('/')  # 代表首页
def index():  # 视图函数

    return render_template('index.html')  # 加载并渲染模板


if __name__ == '__main__':
    # 0.0.0.0代表任何能代表这台机器的地址都可以访问
```

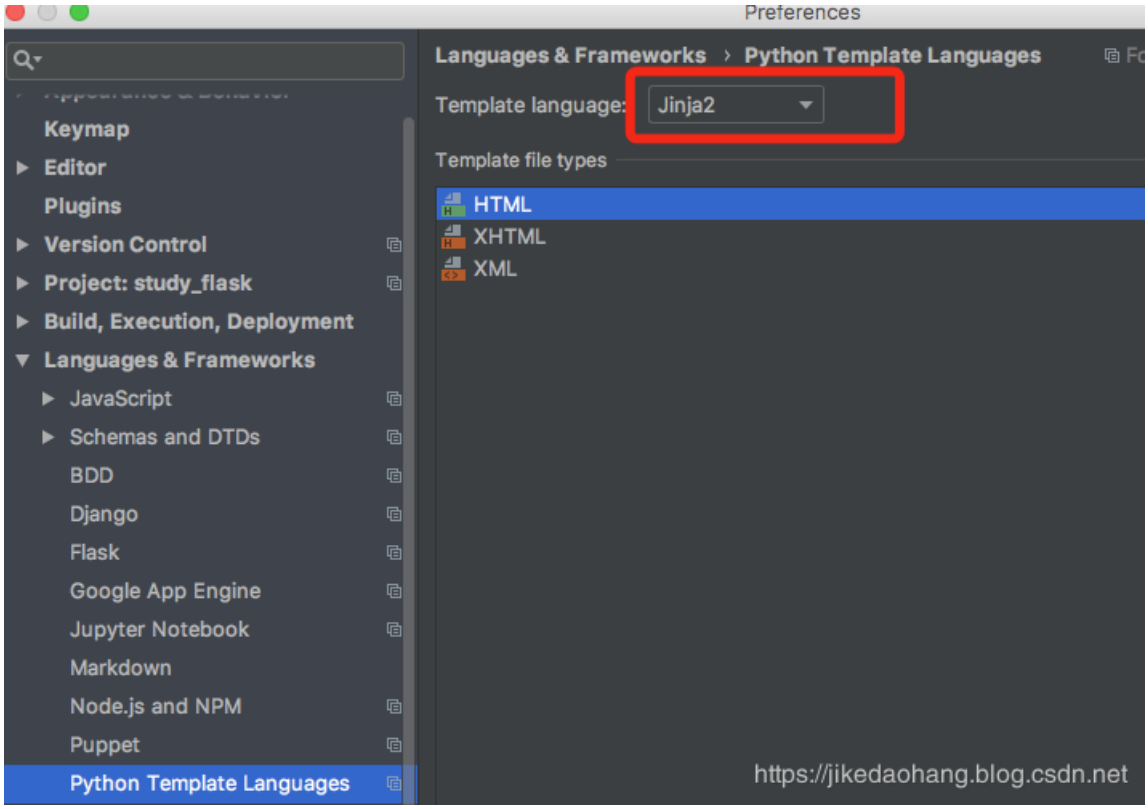
```
app.run(host='0.0.0.0', port=5000)  # 运行程序
```

我们在浏览器访问试一下：



三、变量

现在模板的内容是写死的，这肯定满足不了咱们，我们希望把后台返回的内容渲染在到模板上。Flass 使用 Jinja2 这个模板引擎来渲染模板。我们先把pycharm里的模板语言改成 Jinja2



我们在后台模拟一些数据，返回给模板渲染，render_template 函数第二个参数就是我们要返回的内容。

复制代码

```
from flask import Flask
from flask import render_template

app = Flask(__name__)


@app.route('/')  # 代表首页
def index():  # 视图函数

    ctx = {
        "name": '老王',
        "age": 12,
        "hobby": ["下棋", '电影'],
        "test": {"a": 1, "b": 2}
    }

    return render_template('index.html', **ctx)  # 加载并渲染模板
    # 下面这种也可以
```

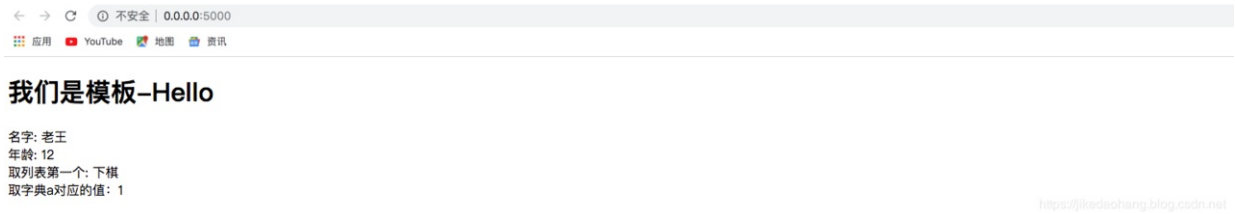
```
    # return render_template('index.html', name='laowang', age=12, hobby=["下棋", '电影'], test={"a": 1, "b": 2})  # 加载并渲染模板


if __name__ == '__main__':
    # 0.0.0.0代表任何能代表这台机器的地址都可以访问
    app.run(host='0.0.0.0', port=5000)  # 运行程序
```

在模板中用模板语言解析出我们的数据,解析变量我们{{}},这种语法我们叫变量代码块：

复制代码

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

<h1>我们是模板-Hello</h1>

名字: <span>{{ name }}</span> <br>
年龄: <span>{{ age }}</span> <br>
取列表第一个: <span>{{ hobby[0] }}</span><br>
取字典a对应的值:<span>{{ test.a }}</span><br>


</body>
</html>
```

我们在浏览器访问试一下：



模版中的变量代码块可以是任意 Python 类型或者对象，只要它能够被 Python 的 str() 方法转换为一个字符串就可以。

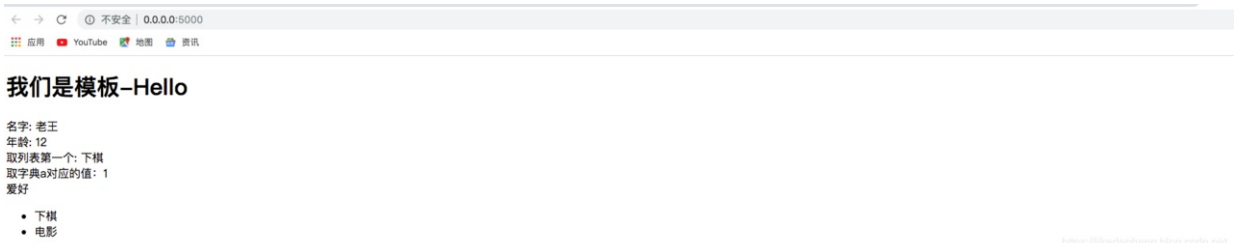
四、标签

用 {%%} 定义的控制语句的代码块，可以实现一些语言层次的功能，比如循环或者if语句。比如我们要上面的所有的爱好变量出来，模板中可以增加如下代码：

复制代码

```
爱好:
<ul>
    {% for h in hobby %}
        <li>{{ h }}</li>
    {% endfor %}

</ul>
```

我们在浏览器访问试一下：



当然我们也可以使用if语句,我们把每行改成不同的颜色:

复制代码

```
爱好:
<ul>
    {% for h in hobby %}
        {% if loop.index == 1 %}
            <li style="background-color: red">{{ h }}</li>
        {% else %}
            <li style="background-color: chartreuse">{{ h }}</li>
        {% endif %}
    {% endfor %}

</ul>
```

其中的 `loop.index` 表示遍历的索引,默认从1开始。我们在浏览器访问试一下:

五、注释

使用 {# #} 进行注释,注释的内容不会在html中被渲染出来。

复制代码

```
{# {{ name }} #}
```

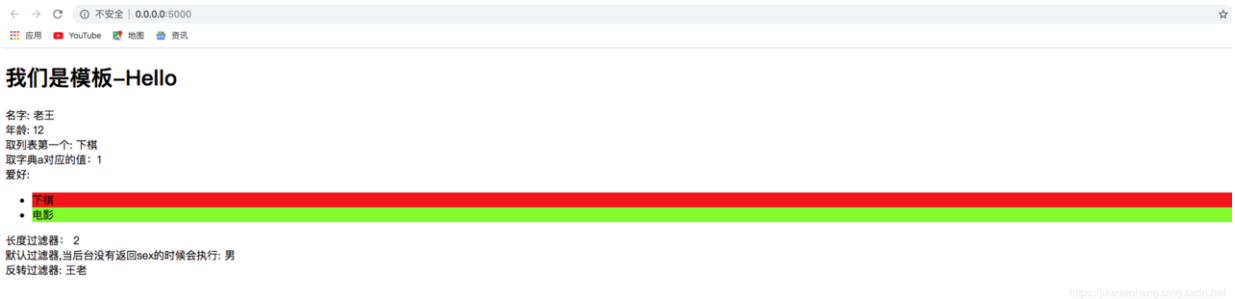
五、过滤器

过滤器顾名思义就是一边进入原始数据,一边出你想要的数据。其本质就是一个函数,比如我们取列表的长度、格式化时间等操作,写法跟linux里面的管道一样。我们在模板中加入一些过滤器试试:

复制代码

```
长度过滤器:
{{ name|length }}<br>

默认过滤器,当后台没有返回sex的时候会执行:
{{ sex|default('男') }}<br>

反转过滤器:
{{ name|reverse }}<br>
```

我们在浏览器访问试一下：



过滤器支撑链式调用：

```
{{ "hello world" | reverse | upper }}
```

语句块操作:

```
{% filter upper %}
    'abc'
{% endfilter %}
```

常用过滤器：

```
禁用转义：{{ '<em>hello</em>' | safe }}
删除标签：{{ '<em>hello</em>' | striptags }}
首字母大写：{{ 'hello' | capitalize }}
所有值小写：{{ 'HELLO' | lower }}
首字母大写：{{ 'hello world' | title }}
字符串反转：{{ 'hello' | reverse }}
字符串截断：{{ 'hello world' | truncate(5) }}

获取列表长度:{{ [1,2,3,4,5,6] | length }}
列表求和:{{ [1,2,3,4,5,6] | sum }}
列表排序:{{ [6,2,3,1,5,4] | sort }}
```

六、自定义过滤器

虽然系统给我提供了很多过滤器，但是需求这种东西是永远满足不了，所以这个时候就需要我们自定义过滤器。上面也说了过滤器的本质其实就一个函数：

```
from flask import Flask
from flask import render_template
import time
from datetime import datetime

app = Flask(__name__)


@app.route('/')  # 代表首页
def index():  # 视图函数

    ctx = {
```

```
        "name": '老王',
        "age": 12,
        "hobby": ["下棋", '电影'],
        "test": {"a": 1, "b": 2},
        "time": datetime.now()  # 返回时间

    }

    return render_template('index.html', **ctx)  # 加载并渲染模板
    # 下面这种也可以
    # return render_template('index.html', name='laowang', age=12, hobby=["下棋", '电影'], test={"a": 1, "b": 2})  # 加载并渲染模板


# 自定义过滤器
#def handletime(time):
#    return time.strftime('%Y-%m-%d %H:%M')

# 自定义有参过滤器
def handletime(time,mode):
    return time.strftime(mode)


app.jinja_env.filters['handletime'] = handletime  # 注册过滤器

if __name__ == '__main__':
    # 0.0.0.0代表任何能代表这台机器的地址都可以访问
    app.run(host='0.0.0.0', port=5000)  # 运行程序
```

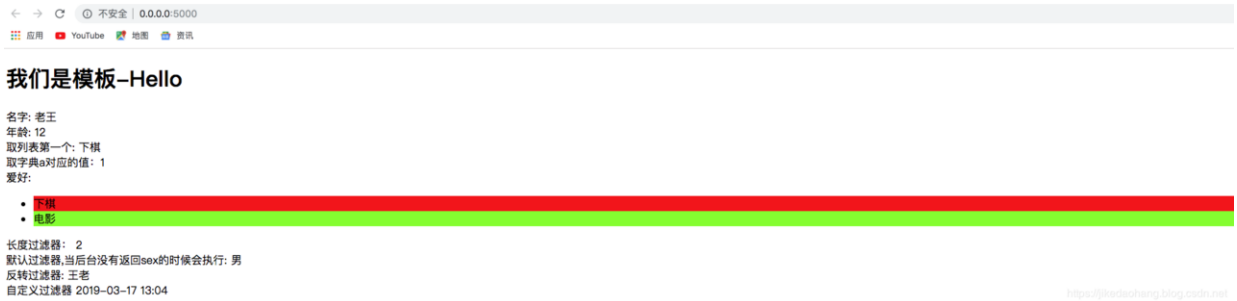
模板中使用过滤器：

复制代码

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

<h1>我们是模板-Hello</h1>

名字: <span>{{ name }}</span> <br>
年龄: <span>{{ age }}</span> <br>
取列表第一个: <span>{{ hobby[0] }}</span><br>
取字典a对应的值:<span>{{ test.a }}</span><br>

爱好:
<ul>
    {% for h in hobby %}
        {% if loop.index == 1 %}
            <li style="background-color: red">{{ h }}</li>
        {% else %}
            <li style="background-color: chartreuse">{{ h }}</li>
        {% endif %}
    {% endfor %}

</ul>


长度过滤器:
{{ name|length }}<br>
```

```
默认过滤器,当后台没有返回sex的时候会执行:
{{ sex|default('男') }}<br>

反转过滤器:
{{ name|reverse }}<br>

自定义有参过滤器
{{ time|handletime('%Y-%m-%d %H:%M') }}


</body>
</html>
```

我们在浏览器访问试一下：



欢迎关注我的公众号：

文章分类

后端

文章标签

Flask

相关推荐