

目录

Python教程

Python简介

安装Python

第一个Python程序

Python基础

函数

高级特性

函数式编程

模块

面向对象编程

面向对象高级编程

错误、调试和测试

IO编程

进程和线程

正则表达式

常用内建模块

常用第三方模块

virtualenv

图形界面

网络编程

电子邮件

访问数据库

Web开发

HTTP协议简介

HTML简介

WSGI接口

使用Web框架

使用模板

异步IO

实战

FAQ

期末总结

关于作者

廖雪峰

自己的Python课程

Python商业爬虫全解码

让天下没有爬不到的数据！

Python爬虫

+

数据分析

Python机器学习

+

深度学习

.....

找廖雪峰老师

廖雪峰老师

自己的Java课程

Java高级架构师

更专业 更权威

源码分析专题

+

微服务架构专题

高并发分布式专题

+

性能优化专题

.....

找廖雪峰老师

unicef

联合国儿童基金会

童年的梦想很美好

他渴望有机会去实现

帮助困境儿童圆梦 >>

unicef

联合国儿童基金会

童年的梦想很美好

他渴望有机会去实现

帮助困境儿童圆梦 >>

使用模板

阅读 160095

Web框架把我们从WSGI中拯救出来了。现在，我们只需要不断地编写函数，带上URL，就可以继续Web App的开发了。

但是，Web App不仅仅是处理逻辑，展示给用户的页面也非常重要。在函数中返回一个包含HTML的字符串，简单的页面还可以，但是，想想新浪首页的6000多行的HTML，你确信能在Python的字符串中正确地写出来？反正我是做不到。

俗话说得好，不懂前端的Python工程师不是好的产品经理。有Web开发经验的同学都明白，Web App最复杂的部分就在HTML页面。HTML不仅要正确，还要通过CSS美化，再加上复杂的JavaScript脚本来实现各种交互和动画效果。总之，生成HTML页面的难度很大。

由于在Python代码里拼字符串是不现实的，所以，模板技术出现了。

使用模板，我们需要预先准备一个HTML文档，这个HTML文档不是普通的HTML，而是嵌入了一些变量和指令，然后，根据我们传入的数据，替换后，得到最终的HTML，发送给用户：

浏览器请求 GET /Michael

name = 'Michael'

app.py

@app.route('/<name>')

def home(name):

return render\_template('home.html', name=name)

模板

<html>

<body>

<p>Hello, {{ name }}!</p>

</body>

</html>

变量 {{ name }}

替换为'Michael'

输出

用户看到的

<html>

<body>

<p>Hello, Michael!</p>

</body>

</html>

这就是传说中的MVC：Model-View-Controller，中文名“模型-视图-控制器”。

Python处理URL的函数就是C：Controller，Controller负责业务逻辑，比如检查用户名是否存在，取出用户信息等等；包含变量 `{{ name }}` 的模板就是V：View，View负责显示逻辑，通过简单地替换一些变量，View最终输出的就是用户看到的HTML。

MVC中的Model在哪？Model是用来传给View的，这样View在替换变量的时候，就可以从Model中取出相应的数据。

上面的例子中，Model就是一个 `dict`：

```
{ 'name': 'Michael' }
```

只是因为Python支持关键字参数，很多Web框架允许传入关键字参数，然后，在框架内部组装出一个 `dict` 作为Model。

现在，我们把上次直接输出字符串作为HTML的例子用高端大气上档次的MVC模式改写一下：

```
from flask import Flask, request, render_template

app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def home():
    return render_template('home.html')

@app.route('/signin', methods=['GET'])
def signin_form():
    return render_template('form.html')

@app.route('/signin', methods=['POST'])
def signin():
    username = request.form['username']
    password = request.form['password']
    if username=='admin' and password=='password':
        return render_template('signin-ok.html', username=username)
    return render_template('form.html', message='Bad username or password', username=username)

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

Flask通过 `render_template()` 函数来实现模板的渲染。和Web框架类似，Python的模板也有很多种。Flask默认支持的模板是jinja2，所以我们先直接安装jinja2：

```
$ pip install jinja2
```

然后，开始编写jinja2模板：

home.html

用来显示首页的模板：

```
<html>
<head>
  <title>Home</title>
</head>
<body>
  <h1 style="font-style:italic">Home</h1>
</body>
</html>
```

form.html

用来显示登录表单的模板：

```
<html>
<head>
  <title>Please Sign In</title>
</head>
<body>
  {% if message %}
  <p style="color:red">{{ message }}</p>
  {% endif %}
  <form action="/signin" method="post">
    <legend>Please sign in:</legend>
    <p><input name="username" placeholder="Username" value="{{ username }}"></p>
    <p><input name="password" placeholder="Password" type="password"></p>
    <p><button type="submit">Sign In</button></p>
  </form>
</body>
</html>
```

signin-ok.html

登录成功的模板：

```
<html>
<head>
  <title>Welcome, {{ username }}</title>
</head>
<body>
  <p>Welcome, {{ username }}!</p>
</body>
</html>
```

登录失败的模板呢？我们在 `form.html` 中加了一点条件判断，把 `form.html` 重用为登录失败的模板。

最后，一定要把模板放到正确的 `templates` 目录下，`templates` 和 `app.py` 在同级目录下：

app.py

templates

form.html

home.html

signin-ok.html

启动 `python app.py`，看看使用模板的页面效果：

Please Sign In

localhost:5000/signin

Bad username or password

Please sign in:

test

Password

Sign In

通过MVC，我们在Python代码中处理M：Model和C：Controller，而V：View是通过模板处理的，这样，我们就成功地把Python代码和HTML代码最大限度地分离了。

使用模板的另一大好处是，模板改起来很方便，而且，改完保存后，刷新浏览器就能看到最新的效果，这对于调试HTML、CSS和JavaScript的前端工程师来说实在是太重要了。

在Jinja2模板中，我们用 `{{ name }}` 表示一个需要替换的变量。很多时候，还需要循环、条件判断等指令语句，在Jinja2中，用 `{% ... %}` 表示指令。

比如循环输出代码：

```
{% for i in page_list %}
  <a href="/page/{{ i }}">{{ i }}</a>
{% endfor %}
```

如果 `page_list` 是一个list： `[[1, 2, 3, 4, 5]]`，上面的模板将输出5个超链接。

除了Jinja2，常见的模板还有：

- Mako：用 `<% ... %>` 和 `$(xxx)` 的一个模板；
- Cheetah：也是用 `<% ... %>` 和 `$(xxx)` 的一个模板；
- Django：Django是一站式框架，内置一个用 `{% ... %}` 和 `{{ xxx }}` 的模板。

小结

有了MVC，我们就分离了Python代码和HTML代码。HTML代码全部放到模板里，写起来更有效率。

源码参考

app.py

读后有收获可以请作者喝咖啡，读后有疑问请加群讨论：

☕

分享

到

微信

还可以分享给朋友：

分享到微博

廖雪峰

官方 独家

Python

商业爬虫全解码

找廖雪峰老师

ACM金牌得主

全球顶尖名企一线数据科学家倾力指导

人工智能与自然语言/计算机视觉课程培训

Artificial Intelligence For NLP/CV Courses

无offer退全款

廖雪峰推荐

JAVA进阶教程

原价1599元

0元领取

阿里云

高性能云服务器首台5折

企业上云事半功倍，最大20Gbps内网带宽，450万PPS

立即选购

5折

Python内部教材+

100G全套学习视频

订阅价：30元免费直播公开课

python大型免费高级进阶公开课

授课模式：在线直播+课后视频，从零基础到中级开发工程师

Python内部教材+

100G全套学习视频

订阅价：30元免费直播公开课

python大型免费高级进阶公开课

授课模式：在线直播+课后视频，从零基础到中级开发工程师

评论

交作业，结合了hmac，mysql

commented at June 6, 2019 6:54 PM. Last updated at: June 6, 2019 6:54 PM.