Chapter2 Templates(模板)

1. 什么是模板

上次课程中的前端页面中只是简单的显示了语句'Hello world!'. 如果想在该页面显示动态信息，例如：需要显示

```
user = {'username': 'Miguel'}
```

改进:不再返回(显示)一个字符串，而是一个HTML页面，则需要做下列修改：

```
# app/routes.py: Return complete HTML page from view function

from app import app

@app.route('/')
@app.route('/index')
def index():
    user = {'username': 'Miguel'}
    return '''
<html>
    <head>
        <title>Home Page - Microblog</title>
    </head>
    <body>
        <h1>Hello, ''' + user['username'] + '''!</h1>
    </body>
</html>'''
```

虽然以上改进已经取得了长足进步，但是缺陷在哪里??

如果能把前端页面代码(html,css,js)与服务器端逻辑代码(这里的路由函数)分散在不同文件中、分开处理，就不是很美好？

Templates help achieve this separation between presentation and business logic. In Flask, templates are written as separate files, stored in a templates folder that is inside the application package.

```
mkdir app/templates
```

```
<!-- app/templates/index.html: Main page template -->
<html>
    <head>
        <title>{{ title }} - Microblog</title>
    </head>
    <body>
```

```
        <h1>Hello, {{ user.username }}!</h1>
    </body>
</html>
```

上述代码中，重点在于{{...}}这个动态内容占位符。 占位符里的内容只有在运行时，动态由server端传过来，生成出来。

```
# app/routes.py: Use render\_template() function

from flask import render_template
from app import app

@app.route('/')
@app.route('/index')
def index():
    user = {'username': 'NXNUCC'}
    return render_template('index.html', title='首页', user=user)
```

render_template() 函数，第一个参数为模板名称，其余参数为模板参数列表，在模板渲染时替换模板中的变量。

2. Jinja2 Conditional Statements(条件)

上例中已经看到Jinjia2中可以在模板渲染时，用真实的值替换占位符中的变量的语法。还有更强大的条件语句判断，如下：

{% ... %}

```
<-- app/templates/index.html: Conditional statement in template -->

<html>
    <head>
        {% if title %}
        <title>{{ title }} - 微博客</title>
        {% else %}
        <title>欢迎访问微博客!</title>
        {% endif %}
    </head>
    <body>
        <h1>Hello, {{ user.username }}!</h1>
    </body>
</html>
```

3. Jinja2 Loops(循环)

```
# app/routes.py: Fake posts in view function
from flask import render_template
from app import app

@app.route('/')
@app.route('/index')
def index():
    user = {'username' : 'NXNUCC'}
    posts = [
        {
            'author' : {'username': 'Chris Paul'},
            'body' : 'half chanmpion?'
        },
        {
            'author':{'username': 'Stephen Curry'},
            'body' : 'excuce me? Paul'
        },
        {
            'author' : {'username': 'James Harden'},
            'body' : 'I am the MVP of 2017-2018 season'
        }
    ]
    return render_template('index.html', title='首页', user=user, posts=posts)
```

上例中，用户发表的内容是一个list, list中的每个元素是一个字典，字典有author和body两个key 继续重构以上的HTML模板代码

```
<-- app/templates/index.html: Conditional statement in template -->

<html>
    <head>
        {% if title %}
        <title>{{ title }} - 微博客</title>
        {% else %}
        <title>欢迎访问微博客!</title>
        {% endif %}
    </head>
    <body>
        <h1>Hello, {{ user.username }}!</h1>

        {% for post in posts %}
        <div><p>{{ post.author.username }} says: <b>{{ post.body }}</b></p></div>
        {% endfor %}

    </body>
</html>
```

Jinja2提供了{% for %} ... {% endfor %}的控制结构，可以方面的进行遍历操作。

4. Jinja2 Template Inheritance

前端页面中有很多区域在不同页面中是相同的，比如：顶部导航、登入登出区域、页面底部版权区域等。如果在每个页面都写相同的重复代码，显然不符合'懒人'的风格。怎么解决呢? 可以使用Template Inheritance

可以定义一个页面，将公共、静态的内容写入其中，其他页面引用(Inheritance)它。

```
<!-- app/templates/base.html: Base template with navigation bar -->
<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">
    {% if title %}
    <title>{{ title }} - 微博客</title>
    {% else %}
    <title>欢迎访问微博客!</title>
    {% endif %}
</head>
<body>
    <div>微博客: <a href="/index">首页</a></div>
    <hr>
    {% block content %} {% endblock %}
</body>
</html>
```

在模板中，使用`block`控制语句，定义了一个区域，这个区域留给继承自该模板的模板插入相应代码。注意`block`需要给一个唯一的name，比如上例中的`content`

继续重构首页模板

```
<!-- app/templates/index.html: Inherit from base template -->
{% extends "base.html" %}

{% block content %}

    <h1>你好, {{user.username}}!</h1>
    {% for post in posts %}
        <div>
            <p>
                {{post.author.username}} says: <b>{{post.body}}</b>
            </p>
        </div>
    {% endfor %}
{% endblock %}
```

The `extends` statement establishes the inheritance link between the two templates, so that Jinja2 knows that when it is asked to render `index.html` it needs to embed it inside `base.html`. The two templates have

matching block statements with name content, and this is how Jinja2 knows how to combine the two templates into one.