Django 的模版

> **Organization**: 千锋教育 Python 教学部 **Date**: 2019-03-10 **Author**: 张旭

一、思考: 前面显示页面的时候有什么缺点

1. HTML 写死在 Python 的代码中，非常不利于修改
2. 页面上相同的部分需要写好多次，HTML 可复用性差

二、什么是模版

模板是一个文本，用于分离文档的表现形式和内容。 模板定义了占位符以及各种用于规范文档该如何显示的各部分基本逻辑（模板标签）。 模板通常用于产生 HTML，但是 Django 的模板也能产生任何基于文本格式的文档。

让我们从一个简单的例子模板开始。 该模板描述了一个向某个与公司签单人员致谢 HTML 页面。 可将其视为一个格式信函:

```
<html>
    <head>
        <title>Ordering notice</title>
    </head>

    <body>
        <h1>Ordering notice</h1>

        <p>Dear {{ person_name }},</p>

        <p>
            Thanks for placing an order from {{ company }}. It's scheduled to
            ship on {{ ship_date|date:"F j, Y" }}.
        </p>

        <p>Here are the items you've ordered:</p>

        <ul>
            {% for item in item_list %}
            <li>{{ item }}</li>
            {% endfor %}
        </ul>

        {% if ordered_warranty %}
        <p>Your warranty information will be included in the packaging.</p>
        {% else %}
        <p>
```

```
            You didn't order a warranty, so you're on your own when the
products
            inevitably stop working.
        </p>
        {% endif %}

        <p>Sincerely,<br />{{ company }}</p>
    </body>
</html>
```

该模板是一段添加了些许变量和模板标签的基础 HTML 。 让我们逐步分析一下:

1. 变量: 用两个大括号括起来的文字（例如 `{{ person_name }}` ）称为 变量(variable) 。这意味着在此处插入指定变量的值。
2. 标签: 被大括号和百分号包围的文本(例如 `{% if ordered_warranty %}`)是 模板标签(template tag) 。标签(tag)定义比较明确，即: 仅通知模板系统完成某些工作的标签。
3. for 循环

   这个例子中的模板包含一个 for 标签（ `{% for item in item_list %}` ）和一个 if 标签（ `{% if ordered_warranty %}` ）

   for 标签类似 Python 的 for 语句，可让你循环访问序列里的每一个项目。
4. if 标签: if 是用来执行逻辑判断的。

   在这里，tag 标签检查 `ordered_warranty` 值是否为 True。

   如果是，模板系统将显示 `{% if ordered_warranty %}` 和 `{% else %}` 之间的内容；否则将显示 `{% else %}` 和 `{% endif %}` 之间的内容。 `{% else %}` 是可选的。
5. filter 过滤器:

   上文这个模板的第二段中有一个关于 filter 过滤器的例子，它是一种最便捷的转换变量输出格式的方式。

   如这个例子中的 `{{ship_date|date:"F j, Y" }}` ，我们将变量 ship_date 传递给 date 过滤器，同时指定参数"F j,Y"。date 过滤器根据参数进行格式输出。 过滤器是用管道符(|)来调用的。

Django 模板含有很多内置的 tags 和 filters,我们将陆续进行学习.

三、如何使用模版

可以这么说，上文展示的模版仅仅是一个页面的框架，如何将有效的内容填充到模版里面呢?

这里用到的就是 render 函数, 由他将数据和模版渲染在一起，组成一个完整的 HTML 文本。

```
from django.shortcuts import render
```

render 函数有三个主要参数, 按位置分别是: 请求对象, 模版名, 上下文变量

1. 请求对象就是 views 中的 request
2. 模版名是我们需要创建的模版文件
3. 上下文变量就是我们需要填充的数据，一般情况下是一个字典。

具体使用方法如下：

模版 1:

```
<p>the cat named {{ name }}</p>
```

对应代码

```
data = {'name': 'kitty'}
return render(request, 'example.html', data)
```

模版 2:

```
<p>{{ person.name }} is {{ person.age }} years old.'</p>
```

对应代码:

```
data = {'name': 'Bob', 'age': 21}
return render(request, 'example.html', {'person': data})
```

或者:

```
class Person:
    def __init__(self):
        name = 'Bob'
        age = 12

bob = Person()
return render(request, 'example.html', {'person': bob})
```

模版 3:

```
<p>My name is {{ person.name }}.</p>
```

对应代码:

```
class Person:
    def __init__(self):
        first_name = 'Bob'
        last_name = 'Green'
        age = 12

    def name(self):
        return '%s %s' % (self.first_name, self.last_name)

bob = Person()
return render(request, 'example.html', {'person': bob})
```

模版 4:

```
<ul>
    <li>Person 1 is {{ items.0 }}.</li>
    <li>Person 2 is {{ items.1 }}.</li>
    <li>Person 3 is {{ items.2 }}.</li>
</ul>
```

对应代码:

```
data = ['Bob', 'Lucy', 'Tom']
return render(request, 'example.html', {'items': data})
```

模版 5:

```
<ul>
    {% for item in items %}
    <li>Person 1 is {{ item.0 }}.</li>
    <li>Person 2 is {{ items.1 }}.</li>
    <li>Person 3 is {{ items.2 }}.</li>
    {% endfor %}
</ul>
```

代码:

```
data = ['Bob', 'Lucy', 'Tom']
return render(request, 'example.html', {'items': data})
```

模版 6:

```
{% if person.sex == 'male' %}
<p>Hi, man</p>
{% else %}
<p>Hi, lady</p>
{% endif %}
```

代码

```
class Person:
def __init__(self):
    name = 'Bob'
    sex = 'male'
    age = 12

bob = Person()
return render(request, 'example.html', {'person': bob})
```

四、静态文件的处理

1. 创建 static 目录，并将文件保存到该目录中
2. settings.py 中添加配置

```
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static"),
]
```

3. urls.py 中添加路由

```
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    # ... the rest of your URLconf goes here ...
    path(...)
]

urlpatterns += static(settings.STATIC_URL,
document_root=settings.STATIC_ROOT)
```

4. 在模版中指明文件路径

```
<img src="/static/abc.png" />
```

五、模版嵌套

1. include 与子模版

```
<!-- 文件 base.html 中 -->
<html>
    <body>
        {% include "nav.html" %}

        <h1>{{ title }}</h1>
    </body>
</html>

<!-- 文件 nav.html 中 -->
<div id="nav">
    <ul>
        <li><a href="...">时政</a></li>
        <li><a href="...">军事</a></li>
        <li><a href="...">动漫</a></li>
    </ul>
</div>
```

2. extends 与模版继承

   基础模版 base.html:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
    <head>
        <title>{% block title %}{% endblock %}</title>
    </head>
    <body>
        <h1>My helpful timestamp site</h1>
        {% block content %}{% endblock %} {% block footer %}
        <hr />
        <p>Thanks for visiting my site.</p>
        {% endblock %}
    </body>
</html>
```

   功能模版:

```
{% extends "base.html" %}

{% block title %}
    The current time
{% endblock %}

{% block content %}
    <p>It is now {{ current_date }}.</p>
{% endblock %}
```