Django 模板

在前面的几节中我们都是用简单的 **django.http.HttpResponse** 来把内容显示到网页上，本节将讲解如何使用渲染模板的方法来显示内容。

1. 创建一个 qf\_tmpl 项目，和一个 名称为 learn 的应用，并且

django-admin.py startproject qf\_tmpl

cd qf\_tmpl

python manage.py startapp learn

2. 把 learn 加入到 settings.INSTALLED\_APPS中

INSTALLED\_APPS = (

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'learn',

)

3. 打开 learn/views.py 写一个首页的视图

from django.shortcuts import render

def home(request):

    return render(request, 'home.html')

4. 在 learn目录下新建一个 templates 文件夹，里面新建一个 home.html

**默认配置下，Django 的模板系统会自动找到app下面的templates文件夹中的模板文件。**

目录的结构是这样的：

qf\_tmpl

├── learn

│   ├── \_\_init\_\_.py

│   ├── admin.py

│   ├── migrations

│   │   └── \_\_init\_\_.py

│   ├── models.py

│   ├── templates

│   │   └── home.html

│   ├── tests.py

│   └── views.py

├── manage.py

└── qf\_tmpl

    ├── \_\_init\_\_.py

    ├── settings.py

    ├── urls.py

    └── wsgi.py

4 directories, 12 files

5. 在 home.html 中写一些内容

<!DOCTYPE html>

<html>

<head>

    <title>欢迎光临</title>

</head>

<body>

欢迎来到千峰

</body>

</html>

6. 将视图函数对应到网址，更改 qf\_tmpl/urls.py

Django 1.8.x 及以上：

from django.conf.urls import include, url

from django.contrib import admin

from learn import views as learn\_views

urlpatterns = [

    url(r'^$', learn\_views.home, name='home'),

    url(r'^admin/', include(admin.site.urls)),

]

8. 运行开发服务器，看看效果

python manage.py runserver

模板补充知识：

网站模板的设计，一般的，我们做网站有一些通用的部分，比如 **导航，底部，访问统计代码等等**

**nav.html, bottom.html, tongji.html**

可以写一个 base.html 来包含这些通用文件（include)

<!DOCTYPE html>

<html>

<head>

    <title>{% block title %}默认标题{% endblock %} - 千峰</title>

</head>

<body>

{% include 'nav.html' %}

{% block content %}

<div>这里是默认内容，所有继承自这个模板的，如果不覆盖就显示这里的默认内容。</div>

{% endblock %}

{% include 'bottom.html' %}

{% include 'tongji.html' %}

</body>

</html>

如果需要，写足够多的 **block** 以便继承的模板可以重写该部分，**include** 是包含其它文件的内容，就是把一些网页共用的部分拿出来，重复利用，改动的时候也方便一些，还可以把广告代码放在一个单独的html中，改动也方便一些，在用到的地方include进去。其它的页面继承自 **base.html** 就好了，继承后的模板也可以在 block 块中 include 其它的模板文件。

比如我们的首页 home.html，继承或者说扩展(extends)原来的 **base.html，**可以简单这样写，重写部分代码（默认值的那一部分不用改）

{% extends 'base.html' %}

{% block title %}欢迎光临首页{% endblock %}

{% block content %}

{% include 'ad.html' %}

这里是首页，欢迎光临

{% endblock %}

**注意：模板一般放在app下的templates中，Django会自动去这个文件夹中找。但 假如我们每个app的templates中都有一个 index.html，当我们在views.py中使用的时候，直接写一个 render(request, 'index.html')，Django 能不能找到当前 app 的 templates 文件夹中的 index.html 文件夹呢?（答案是不一定能，有可能找错）**

**Django 模板查找机制： Django 查找模板的过程是在每个 app 的 templates 文件夹中找（而不只是当前 app 中的代码只在当前的 app 的 templates 文件夹中找）。各个 app 的 templates 形成一个文件夹列表，Django 遍历这个列表，一个个文件夹进行查找，当在某一个文件夹找到的时候就停止，所有的都遍历完了还找不到指定的模板的时候就是 Template Not Found （过程类似于Python找包）。这样设计有利当然也有弊，有利是的地方是一个app可以用另一个app的模板文件，弊是有可能会找错了。所以我们使用的时候在 templates 中建立一个 app 同名的文件夹，这样就好了。**

**这就需要把每个app中的 templates 文件夹中再建一个 app 的名称，仅和该app相关的模板放在 app/templates/app/ 目录下面，**

**例如：项目 qf 有两个 app，分别为 tutorial 和 tryit**

qf

├── tutorial

│   ├── \_\_init\_\_.py

│   ├── admin.py

│   ├── models.py

│   ├── templates

│   │   └── tutorial

│   │       ├── index.html

│   │       └── search.html

│   ├── tests.py

│   └── views.py

├── tryit

│   ├── \_\_init\_\_.py

│   ├── admin.py

│   ├── models.py

│   ├── templates

│   │   └── tryit

│   │       ├── index.html

│   │       └── poll.html

│   ├── tests.py

│   └── views.py

├── manage.py

└── qf

    ├── \_\_init\_\_.py

    ├── settings.py

    ├── urls.py

    └── wsgi.py

这样，使用的时候，模板就是 "tutorial/index.html" 和 "tryit/index.html" 这样有app作为名称的一部分，就不会混淆。

# Django 模板进阶

## **本节主要讲** **Django模板中的循环，条件判断，常用的标签，过滤器的使用。**

1. 列表，字典，类的实例的使用
2. 循环：迭代显示列表，字典等中的内容
3. 条件判断：判断是否显示该内容，比如判断是手机访问，还是电脑访问，给出不一样的代码。
4. **标签**：for，if 这样的功能都是标签。
5. **过滤器**：管道符号后面的功能，比如{{ var**|**length }}，求变量长度的 length 就是一个过滤器。

如果需要将一个或多个变量共享给多个网页或者所有网页使用，比如在网页上显示来访者的IP，这个可以使用 Django 上下文渲染器 来做。

**实例一，显示一个基本的字符串在网页上**

views.py

# -\*- coding: utf-8 -\*-

from django.shortcuts import render

def home(request):

    string = u"我在千峰学习Django，用它来建网站"

    return render(request, 'home.html', {'string': string})

在视图中我们传递了一个字符串名称是 string 到模板 home.html，在模板中这样使用它：

home.html

{{ string }}

app/templates/home.html 来测试后面的示例)

**实例二，讲解了基本的 for 循环 和 List内容的显示**

views.py

def home(request):

    TutorialList = ["HTML", "CSS", "jQuery", "Python", "Django"]

    return render(request, 'home.html', {'TutorialList': TutorialList})

在视图中我们传递了一个List到模板 home.html，在模板中这样使用它：

home.html

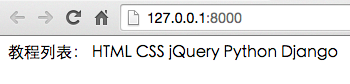
教程列表：

{% for i in TutorialList %}

{{ i }}

{% endfor %}

for 循环要有一个结束标记，上面的代码假如我们对应的是首页的网址（自己修改urls.py），显示在网页上就是：



**简单总结一下：**一般的变量之类的用 {{ }}（变量），功能类的，比如循环，条件判断是用 {%  %}（标签）

**实例三，显示字典中内容：**

views.py

def home(request):

    info\_dict = {'site': u'千峰', 'content': u'各种IT技术教程'}

    return render(request, 'home.html', {'info\_dict': info\_dict})

home.html

站点：{{ info\_dict.site }} 内容：{{ info\_dict.content }}

在模板中取字典的键是用点info\_dict.site，而不是Python中info\_dict['site']

还可以这样遍历字典：

{% for key, value in info\_dict.items %}

    {{ key }}: {{ value }}

{% endfor %}

其实就是遍历这样一个 List:  **[('site', u'千峰'), ('content', u'各种IT技术教程')]**

**实例四，在模板进行 条件判断和 for 循环的详细操作：**

**views.py**

def home(request):

    List = map(str, range(100))# 一个长度为100的 List

    return render(request, 'home.html', {'List': List})

假如我们想用逗号将这些元素连接起来：

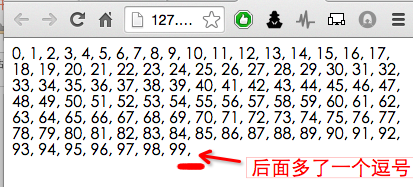
home.html

{% for item in List %}

    {{ item }},

{% endfor %}

效果如下：



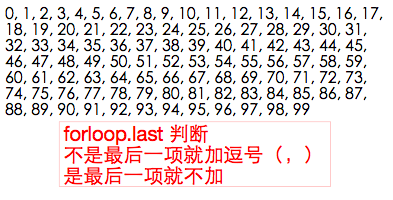
我们会发现最后一个元素后面也有一个逗号，这样肯定不爽，如果判断是不是遍历到了最后一个元素了呢？

用变量 forloop.last 这个变量，如果是最后一项其为真，否则为假，更改如下：

{% for item in List %}

    {{ item }}{% if not forloop.last %},{% endif %}

{% endfor %}



在for循环中还有很多有用的东西，如下：

| **变量** | **描述** |
| --- | --- |
| **forloop.counter** | 索引从 1 开始算 |
| **forloop.counter0** | 索引从 0 开始算 |
| **forloop.revcounter** | 索引从最大长度到 1 |
| **forloop.revcounter0** | 索引从最大长度到 0 |
| **forloop.first** | 当遍历的元素为第一项时为真 |
| **forloop.last** | 当遍历的元素为最后一项时为真 |
| **forloop.parentloop** | 用在嵌套的 for 循环中，  获取上一层 for 循环的 forloop |

当列表中可能为空值时用 for  empty

<ul>

{% for athlete in athlete\_list %}

    <li>{{ athlete.name }}</li>

{% empty %}

    <li>抱歉，列表为空</li>

{% endfor %}

</ul>

**实例五，模板上得到视图对应的网址：**

# views.py

def add(request, a, b):

    c = int(a) + int(b)

    return HttpResponse(str(c))

# urls.py

urlpatterns = patterns('',

    url(r'^add/(\d+)/(\d+)/$', 'app.views.add', name='add'),

)

# template html

{% url 'add' 4 5 %}

这样网址上就会显示出：/add/4/5/ 这个网址，假如我们以后修改 urls.py 中的

r'^add/(\d+)/(\d+)/$'

这一部分，改成另的，比如：

r'^jiafa/(\d+)/(\d+)/$'

这样，我们不需要再次修改模板，当再次访问的时候，网址会自动变成 /jiafa/4/5/

还可以使用 as 语句将内容取别名（相当于定义一个变量），多次使用（但视图名称到网址转换只进行了一次）

{% url 'some-url-name' arg arg2 as the\_url %}

<a href="{{ the\_url }}">链接到：{{ the\_url }}</a>

**实例六，模板中的逻辑操作：**

6.1、   ==, !=, >=, <=, >, < 这些比较都可以在模板中使用，比如：

{% if var >= 90 %}

成绩优秀，千峰你没少去吧！学得不错

{% elif var >= 80 %}

成绩良好

{% elif var >= 70 %}

成绩一般

{% elif var >= 60 %}

需要努力

{% else %}

不及格啊，大哥！多去千峰学习啊！

{% endif %}

（**注意：比较符号前后必须有至少一个空格！**）

and, or, not, in, not in 也可以在模板中使用

假如我们判断 num 是不是在 0 到 100 之间：

{% if num <= 100 and num >= 0 %}

num在0到100之间

{% else %}

数值不在范围之内！

{% endif %}

假如我们判断 'qianfeng' 在不在一个列表变量 List 中：

{% if 'qianfeng' in List %}

千峰在名单中

{% endif %}

**实例七，模板中 获取当前网址，当前用户等：**

如果不是在 views.py 中用的 [**render**](https://docs.djangoproject.com/en/dev/topics/http/shortcuts/#render)函数，是 [render\_to\_response](https://docs.djangoproject.com/en/dev/topics/http/shortcuts/#django.shortcuts.render_to_response) 的话，需要将 request 加入到 上下文渲染器

Django 1.8 及以后 修改 settings.py （Django 1.11 中新建的项目已经默认加上了）

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                ...

                'django.template.context\_processors.request',

                ...

            ],

        },

    },

]

然后在 模板中我们就可以用 request 了。一般情况下，推荐用 [**render**](https://docs.djangoproject.com/en/dev/topics/http/shortcuts/#render) 而不是用 [render\_to\_response](https://docs.djangoproject.com/en/dev/topics/http/shortcuts/#django.shortcuts.render_to_response)

**7.1 获取当前用户：**

{{ request.user }}

如果登陆就显示内容，不登陆就不显示内容：

{% if request.user.is\_authenticated %}

    {{ request.user.username }}，您好！

{% else %}

    请登陆，这里放登陆链接

{% endif %}

**7.2.1 获取当前网址：**

{{ request.path }}

**7.2.2 获取当前 GET 参数：**

{{ request.GET.urlencode }}

**7.2.3 合并到一起用的一个例子：**

<a href="{{ request.path }}?{{ request.GET.urlencode }}">当前网址加参数 </a>