3个步骤：
1）【blog/views.py】
添加视图函数
2）【blog/urls.py】
绑定路由和视图函数
3）【base.html】
修改链接。改为url

11-分类与归档

侧边栏已经正确地显示了最新文章列表、归档、分类等信息。现在来完善归档和分类功能，当用户点击归档下的某个日期或者分类下的某个分类时，跳转到文章列表页面，显示该日期或者分类下的全部文章。

1、归档页面

要显示某个归档日期下的文章列表，思路和显示主页文章列表是一样的，回顾一下主页视图的代码：

【blog/views.py】

```
def index(request):
    post_list = Post.objects.all().order_by('-created_time')
    return render(request, 'blog/index.html', context={'post_list': post_list})
```

主页视图函数中我们通过 `Post.objects.all()` 获取全部文章，而在我们的归档和分类视图中，我们不再使用 `all` 方法获取全部文章，而是使用 `filter` 来根据条件过滤。先来看归档视图：

【blog/views.py】

```
def archives(request, year, month):
    post_list = Post.objects.filter(created_time__year=year,
                                    created_time__month=month
                                    ).order_by('-created_time')
    return render(request, 'blog/index.html', context={'post_list': post_list})
```

这里我们使用了模型管理器（objects）的 `filter` 函数来过滤文章。由于是按照日期归档，因此这里根据文章发表的年和月来过滤。具体来说，就是根据 `created_time` 的 `year` 和 `month` 属性过滤，筛选出文章发表在对应的 year 年和 month 月的文章。注意这里 `created_time` 是 Python 的 `date` 对象，其有一个 `year` 和 `month` 属性，我们在上一章使用过这个属性。**Python 中类实例调用属性的方法通常是 `created_time.year`，但是由于这里作为函数的参数列表，所以 Django 要求我们把点替换成了两个下划线，即 `created_time__year`。**同时和 index 视图中一样，我们对返回的文章列表进行了排序。此外由于归档的下的文章列表的显示和首页是一样的，因此我们直接渲染了 index.html 模板。

写好视图函数后就是配置好 URL：

【blog/urls.py】

```
from django.conf.urls import url

from . import views

app_name = 'blog'
urlpatterns = [
   url(r'^$', views.index, name='index'),
   url(r'^post/(?P<pk>[0-9]+)/$', views.detail, name='detail'),
   + url(r'^archives/(?P<year>[0-9]{4})/(?P<month>[0-9]{1,2})/$', views.archives,
name='archives'),
]
```

这个归档视图对应的 URL 的正则表达式和 detail 视图函数对应的 URL 是类似的，这在之前我们讲过。两个括号括起来的地方是两个命名组参数，Django 会从用户访问的 URL 中自动提取这两个参数的值，然后传递给其对应的视图函数。例如如果用户想查看 2017 年 3 月下的全部文章，他**访问 /archives/2017/3/，那么 archives 视图函数的实际调用为：archives(request, year=2017, month=3)。**

在模板找到归档列表部分的代码，修改超链接的 href 属性，让用户点击超链接后跳转到文章归档页面：

【templates/base.html】

```
{% for date in date_list %}
<li>
 <a href="{% url 'blog:archives' date.year date.month %}">
  {{ date.year }} 年 {{ date.month }} 月
 </a>
</li>
{% endfor %}
```

这里 {% url %} 这个模板标签的作用是**解析视图函数 blog:archives 对应的 URL 模式**，并把 URL 模式中的年和月替换成 date.year，date.month 的值。（**相当于 Flask 里模板中的{{url_for()}}标签**）例如 blog:archives 表示 blog 应用下的 archives 函数，这个函数对应的 URL 模式为 ^archives/(?P<year>[0-9]{4})/(?P<month>[0-9]{1,2})/$，假设 date.year=2017，date.month=5，那么 {% url 'blog:archives' date.year date.month %} 模板标签返回的值为/archives/2017/5/。

为什么要使用 {% url %} 模板标签呢？事实上，我们把超链接的 href 属性设置为 /archives/{{ date.year }}/{{ date.month }}/ 同样可以达到目的，但是这种写法是硬编码的。虽然现在 blog:archives 视图函数对应的 URL 模式是这种形式，但是如果哪天这个模式改变了呢？如果使用了硬编码的写法，那你需要把每一处 /archives/{{ date.year }}/{{ date.month }}/ 修改为新的模式。但如果使用了 {% url %} 模板标签，则不用做任何修改（相当于 Flask 中 url_for 的好处）。

测试一下，点击侧边栏归档的日期，跳转到归档页面，发现报了个错误，提示没有安装 pytz（python time zone，时区设置）。激活虚拟环境，使用 `pip install pytz` 安装即可。

重启一下开发服务器，再次测试，发现可以显示归档下的文章列表了。

2、分类页面

同样的写好分类页面的视图函数：

【blog/views.py】

```
import markdown

from django.shortcuts import render, get_object_or_404

# 引入 Category 类
from .models import Post, Category

def category(request, pk):
    cate = get_object_or_404(Category, pk=pk)
    post_list = Post.objects.filter(category=cate).order_by('-created_time')
    return render(request, 'blog/index.html', context={'post_list': post_list})
```

这里我们首先根据传入的 pk 值（也就是被访问的分类的 id 值）从数据库中获取到这个分类。`get_object_or_404` 函数和 detail 视图中一样，其作用是如果用户访问的分类不存在，则返回一个 404 错误页面以提示用户访问的资源不存在。然后我们通过 `filter` 函数过滤出了该分类下的全部文章。同样也和首页视图中一样对返回的文章列表进行了排序。

URL 配置如下：

【blog/urls.py】

```
from django.conf.urls import url

from . import views

app_name = 'blog'
urlpatterns = [
    url(r'^$', views.index, name='index'),
    url(r'^post/(?P<pk>[0-9]+)/$', views.detail, name='detail'),
    url(r'^archives/(?P<year>[0-9]{4})/(?P<month>[0-9]{1,2})/$', views.archives,
name='archives'),
    + url(r'^category/(?P<pk>[0-9]+)/$', views.category, name='category'),
]
```

这个分类页面对应的 URL 模式和文章详情页面对应的 URL 模式十分类似，你可以自己分析分析它是如何工作的，在此就不赘述了。

修改相应模板：

【templates/base.html】

```
{% for category in category_list %}
<li>
 <a href="{% url 'blog:category' category.pk %}">{{ category.name }}</a>
</li>
{% endfor %}
```

同样，{% url %} 模板标签的用法和写归档页面时的用法是一样的。现在尝试点击相应的链接，就可以跳转到归档或者分类页面了。