23-简单全文搜索

搜索是一个复杂的功能，但对于一些简单的搜索任务，我们可以使用 Django Model 层提供的一些内置方法来完成。现在我们来为我们的博客提供一个简单的搜索功能。

0、概述

博客文章通常包含标题和正文两个部分。当用户输入某个关键词进行搜索后，我们希望为用户显示标题和正文中含有被搜索关键词的全部文章。整个搜索的过程如下：

1. 用户在搜素框中输入搜索关键词，假设为 “django” ，然后用户点击了搜索按钮提交其输入的结果到服务器。
2. 服务器接收到用户输入的搜索关键词 “django” 后去数据库查找文章标题和正文中含有该关键词的全部文章。
3. 服务器将查询结果返回给用户。

整个过程就是这样，下面来看看 Django 如何用实现这些过程。

1、将关键词提交给服务器

先来回顾一下我们的 Django 博客的 Post（文章）模型：

【blog/models.py】

```
class Post(models.Model):
    # 标题
    title = models.CharField(max_length=70)
    # 正文
    body = models.TextField()

    # 其他属性...

    def __str__(self):
        return self.title
```

先看到第 1 步，用户在搜索框输入搜索关键词，因此我们要在博客的导航条后为用户提供一个搜索表单，HTML 表单代码大概像这样：

【base.html】

```
                <div id="header-search-box">
```

```
                    <a id="search-menu" href="#"><span id="search-icon" class="ion-ios-search-strong"></span></a>
                    <div id="search-form" class="search-form">
                        <form role="search" method="get" id="searchform" action="{% url 'blog:search' %}">
                            <input type="search" name="q" placeholder="搜索" required>
                            <button type="submit"><span class="ion-ios-search-strong"></span></button>
                        </form>
                    </div>
                </div>
```

用户输入了搜索关键词并点击了搜索按钮后，数据就被发送给了 Django 后台服务器。表单的 `action` 属性的值为 {% url 'blog:search' %}（虽然我们还没有写这个视图函数），表明用户提交的结果将被发送给 blog 应用下 search 视图函数对应的 URL。

2、查找含有搜索关键词的文章

搜索的功能将由 search 视图函数提供，代码写在 blog/views.py 里：

【blog/views.py】

```
from django.db.models import Q

def search(request):
    q = request.GET.get('q')
    error_msg = ''

    if not q:
        error_msg = "请输入关键词"
        return render(request, 'blog/index.html', {'error_msg': error_msg})

    post_list = Post.objects.filter(Q(title__icontains=q) | Q(body__icontains=q))
    return render(request, 'blog/index.html', {'error_msg': error_msg,
                                               'post_list': post_list})
```

首先我们使用 `request.GET.get('q')` 获取到用户提交的搜索关键词。用户通过表单 get 方法提交的数据 Django 为我们保存在 **request.GET** 里，这是一个类似于 Python 字典的对象，所以我们使用 `get` 方法从字典里取出键 q 对应的值，即用户的搜索关键词。这里字典的键之所以叫 q 是因为我们的表单中搜索框 input 的 name 属性的值是 q，如果修改了 name 属性的值，那么这个键的名称也要相应修改。

接下来我们做了一个小小的校验，如果用户没有输入搜索关键词而提交了表单，我们就无需执行查询，我们就在模板中渲染一个错误提示信息。

如果用户输入了搜索关键词，我们就通过 `filter` 方法从数据库里过滤出符合条件的所有文章。这里的过滤条件是 `title__icontains=q`，即 title 中包含（contains）关键字 q，前缀 i 表示不区分大小写。这里 `icontains` 是查询表达式（Field lookups），我们在之前也使用过其他类似的查询表达式，其用法是在模型需要筛选的属性后面跟上两个下划线。Django 内置了很多查询表达式，建议过一遍 Django 官方留个印象，了解每个表达式的作用，以后碰到相关的需求就可以快速定位到文档查询其用途 Field lookups。

此外我们这里从 from django.db.models 中引入了一个新的东西：Q 对象。Q 对象用于包装查询表达式，其作用是为了提供复杂的查询逻辑。例如这里 `Q(title__icontains=q) | Q(body__icontains=q)` 表示标题（title）含有关键词 q 或者正文（body）含有关键词 q ，或逻辑使用 `|` 符号。如果不用 Q 对象，就只能写成 `title__icontains=q, body__icontains=q`，这就变成标题（title）含有关键词 q 且正文（body）含有关键词 q，就达不到我们想要的目的。

3、渲染搜索结果

接下来就是渲染搜索结果页面，这里我们复用了 index.html 模板，唯一需要修改的地方就是当有错误信息时，index.html 应该显示错误信息。只需要在文章列表前加个 `error_msg` 模板变量即可：

【templates/blog/index.html】

```
{% extends 'base.html' %}
{% block main %}
 {% if error_msg %}
  <p>{{ error_msg }}</p>
 {% endif %}

 {% for post in post_list %}
  ...
 {% empty %}
  <div class="no-post">暂时还没有发布的文章！</div>
 {% endfor %}
{% endblock main %}
```

4、绑定 URL

有了视图函数后记得把视图函数映射到相应了 URL，如下。

blog/urls.py

```
urlpatterns = [
  # 其他 url 配置
```

```
    url(r'^search/$', views.search, name='search'),
]
```

大功告成，在导航栏尝试输入一些关键词，看看效果吧！

当然这样的搜索功能是非常简略的，难以满足一些复杂的搜索需求。编写一个搜索引擎是一个大工程，好在 django-haystack 这款第三方 app 为我们完成了全部工作。使用它我们可以实现更加复杂的搜索功能，比如全文检索、按搜索相关度排序、关键字高亮等等类似于百度搜索的功能，功能十分强大。当然其使用也会复杂一些，下一篇教程将向大家介绍 django-haystack 的使用方法。