20-标签云

我们博客的文章（Post）模型除了通过 ForeignKey 关联了 Category（分类）外，还通过 ManyToMany 关联了 Tag（标签）。在我们的侧边栏可以看到一个标签云效果的全部标签列表。现在我们来给博客实现这个效果，**让 Django 从数据库中获取全部标签的数据列表，然后在模板中显示它们，并且点击相应的标签，就可以显示该标签下的全部文章列表。**

1、获取标签列表

很明显的能够发现，标签和之前我们开发的分类功能是十分类似的，唯一的不同是一篇文章（Post）只能指定一个分类，但是却可以指定多个标签。回顾一下我们获取博客侧边栏的分类列表时是怎么做的呢？我们自定义了一个模板标签函数 `get_categories`。具体的代码是：

【blog/templatetags/blog_tags.py】

```
@register.simple_tag
def get_categories():
    # 记得在顶部引入 count 函数
    return Category.objects.annotate(num_posts=Count('post')).filter(num_posts__gt=0)
```

然后我们在模板中使用这个模板标签获取到文章数大于 0 的分类列表，并渲染显示它。

【templates/base.html】

```
<div class="widget widget-category">
  <h3 class="widget-title">分类</h3>
  {% get_categories as category_list %}
  <ul>
    {% for category in category_list %}
    <li>
      <a href="{% url 'blog:category' category.pk %}">{{ category.name }}
        <span class="post-count">({{ category.num_posts }})</span>
      </a>
    </li>
    {% empty %}
    暂无分类！
    {% endfor %}
  </ul>
</div>
```

事实上，**标签云的实现方法和分类列表完全一样。**我们**定义一个 `get_tags` 模板标签，获取到文章数大于 0 的标签列表，然后在模板中渲染显示它。**代码如下，你可以看到代码和分类功能的代码几乎是一样的，只是把 Category（分类）换成了 Tag（标签）。

【blog/templatetags/blog_tags.py】

```
from ..models import Post, Category, Tag

@register.simple_tag
def get_tags():
  # 记得在顶部引入 Tag model
  return Tag.objects.annotate(num_posts=Count('post')).filter(num_posts__gt=0)
```

然后在模板中循环显示这些标签：

【templates/base.html】

```
<div class="widget widget-tag-cloud">
 <h3 class="widget-title">标签云</h3>
 {% get_tags as tag_list %}
 <ul>
  {% for tag in tag_list %}
  <li>
   <a href="#">{{ tag.name }}</a>
  </li>
  {% empty %}
  暂无标签！
  {% endfor %}
 </ul>
</div>
```

关于自定义模板标签以及使用方法，请参考 “页面侧边栏：使用自定义模板标签”一章。

在 Django 后台添加一些标签，并且为发表的文章指定这些标签，就可以看到博客的侧边栏显示出这些标签了。

2、显示某个标签下的文章列表

同样的，显示某个标签下的文章列表和我们之前做的点击分类后显示该分类下的文章列表是一样的。回顾一下显示分类下的文章列表时的做法，经典的 Django 三部曲。首先是定义视图函数，然后编写模板文件，最后将视图函数和 URL 模式绑定。标签和分类是完全一样的步骤，因此稍微修改一下分类相关的代码就可以用于标签了。

（1）标签视图函数

【blog/views.py】

```
class TagView(ListView):
    model = Post
    template_name = 'blog/index.html'
    context_object_name = 'post_list'

    def get_queryset(self):
        tag = get_object_or_404(Tag, pk=self.kwargs.get('pk'))
        return super(TagView, self).get_queryset().filter(tags=tag)
```

和 CategoryView 一样，我们使用了类视图。代码几乎和 CategoryView 是一样的，因此这里不再详细说明，具体请参考 CategoryView 部分的代码和说明。

（2）模板

由于显示的是文章列表，因此我们直接复用了用于显示文章列表的 index.html 模板。

（3）绑定 URL

同样的，URL 模式和分类也是完全类似的，这里不再多做解释：

【blog/urls.py】

```
app_name = 'blog'
urlpatterns = [
    # 其它 URL 模式...
    url(r'^category/(?P<pk>[0-9]+)/$', views.CategoryView.as_view(), name='category'),
    url(r'^tag/(?P<pk>[0-9]+)/$', views.TagView.as_view(), name='tag'),
]
```

（4）设置标签跳转链接

设置一下标签的超链接，这样点击标签后就可以跳转到该标签下的文章列表页面了。这里用到了 {% url %} 模板标签，其用法和分类的超链接一模一样，这里就不再过多解释，请参考上边给出的一些文章。

```
<a href="{% url 'blog:tag' tag.pk %}">{{ tag.name }}</a>
```

3、在文章详情页显示标签

上边获取的是全部标签的列表，当在文章详情页面时，我们希望显示的这篇文章所属的标签，具体该怎么做呢？这里我只说明几个关键的点，然后给出一个大致的实现思

路。既然你已经通过教程学习到了这里，相信你对 Django 已经有了一定了解了，根据提示并稍加思考，相信你一定可以很好地完成这个功能。

首先来回顾一下文章（Post）和标签（Tag）的模型：

【blog/models.py】

```
class Post(models.Model):
    title = models.CharField(max_length=70)
    body = models.TextField()
    category = models.ForeignKey('Category')
    tags = models.ManyToManyField(Tag, blank=True)
    # 其它属性...

    def __str__(self):
        return self.title

class Tag(models.Model):
    name = models.CharField(max_length=100)
```

可以看到 Post 下有一个 tags 属性，这个属性通过多对多的关系关联着 Tag。回顾一下我们是如何获取某篇文章 post 对应的分类的？我们直接通过访问 category 属性来获得分类，即通过 post.categoty 来获取 post 的分类。那要获得某篇文章 post 对应的标签，可不可以通过 post.tags 来获取呢？思路是正确的，不过和获取分类稍微有点不同。由于 Post 和 Categoty 是一对多的关系（ForeignKey），所以 post.categoty 是唯一的。但是 Post 和 Tag 是多对多的关系（ManyToManyField），那么 post.tags 就有可能有多个值。所以 Django 没有让 post.tags 返回全部标签，而是返回了一个**模型管理器（类似于 objects）**，然后我们可以**调用这个模型管理器的 all 方法**，来获取这篇 post 下的全部标签列表了。

因此大体思路就清晰了，我们可以在文章的详情页模板中，通过 post.tags.all() 获取到这篇 post 下的标签列表。但是要注意**模板中调用方法需要去掉括号**：

【templates/blog/detail.html】

```
        <div class="entry-content clearfix">
            {{ post.body|safe }}
            <div class="widget-tag-cloud">
                <ul>
                    标签：
                    {% for tag in post.tags.all %}
                        <li><a href="{% url 'blog:tag' tag.pk %}"># {{ tag.name }}</a></li>
                    {% endfor %}
                </ul>
            </div>
        </div>
```