19-统计各个分类下的文章数

在我们的博客侧边栏有分类列表，显示博客已有的全部文章分类。现在想在分类名后显示该分类下有多少篇文章，该怎么做呢？最优雅的方式就是使用 Django 模型管理器的 **annotate** 方法。

0、模型回顾

回顾一下我们的模型代码，Django 博客有一个 Post 和 Category 模型，分别表示文章和分类：

【blog/models.py】

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

    def __str__(self):
        return self.title

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

我们知道从数据库取数据都是使用模型管理器 objects 的方法实现的。比如获取全部分类是：`Category.objects.all()`，假设有一个名为 test 的分类，那么获取该分类的方法是：`Category.objects.get(name='test')` 。objects 除了 `all`、`get` 等方法外，还有很多操作数据库的方法，而其中有一个 **annotate** 方法，该方法可以帮我们实现本文所关注的统计分类下的文章数量的功能。

1、数据库数据聚合

`annotate` 方法在底层调用了数据库的**数据聚合函数**，下面使用一个实际的数据库表来帮助我们理解 `annotate` 方法的工作原理。在 Post 模型中我们通过 ForeignKey 把 Post 和 Category 关联了起来，这时候它们的数据库表结构就像下面这样：

Post 表：

id	title	body	category_id

1	post 1	...	1
2	post 2	...	1
3	post 3	...	1
4	post 4	...	2

Category 表：

name	id
category 1	1
category 2	2

这里前 3 篇文章属于 category 1，第 4 篇文章属于 category 2。

当 Django 要查询某篇 post 对应的分类时，比如 post 1，首先查询到它分类的 id 为 1，然后 Django 再去 Category 表找到 id 为 1 的那一行，这一行就是 post 1 对应的分类。反过来，如果要查询 category 1 对应的全部文章呢？category 1 在 Category 表中对应的 id 是 1，Django 就在 Post 表中搜索哪些行的 category_id 为 1，发现前 3 行都是，把这些行取出来就是 category 1 下的全部文章了。同理，这里 `annotate` 做的事情就是把全部 Category 取出来，然后去 Post 查询每一个 Category 对应的文章，查询完成后只需算一下每个 category id 对应有多少行记录，这样就可以统计出每个 Category 下有多少篇文章了。把这个统计数字保存到每一条 Category 的记录就可以了（当然**并非保存到数据库**，在 Django ORM 中是**保存到 Category 的实例的属性中**，每个实例对应一条记录）。

2、使用 Annotate

以上是原理方面的分析，具体到 Django 中该如何用呢？在我们的博客中，获取侧边栏的分类列表的方法写在模板标签 `get_categories` 里，因此我们修改一下这个函数，具体代码如下：

【blog/templatetags/blog_tags.py】

```
from django.db.models.aggregates import Count
from blog.models import Category

@register.simple_tag
def get_categories():
    # 记得在顶部引入 count 函数
```

```
    # Count 计算分类下的文章数，其接受的参数为需要计数的模型的名称
    return Category.objects.annotate(num_posts=Count('post')).filter(num_posts__gt=0)
```

这个 Category.objects.annotate 方法和 Category.objects.all 有点类似，它**会返回数据库中全部 Category 的记录**，但同时它还会**做一些额外的事情（annotate 的作用就是往模型类中新增一个属性，这里新增了 num_posts 属性，注意并非保存到数据库，在 Django ORM 中是保存到 Category 的实例的属性中）**，在这里我们希望它做的额外事情就是去统计返回的 Category 记录的集合中每条记录下的文章数。代码中的 Count 方法为我们做了这个事，它接收一个**和 Categoty 相关联的模型参数名**（这里是 Post，通过 ForeignKey 关联的），然后它便会统计 Category 记录的集合中每条记录下的与之关联的 Post 记录的行数，也就是文章数，最后把这个值保存到 num_posts 属性中。

此外，我们还对结果集做了一个过滤，使用 filter 方法把 num_posts （**此时 num_posts 已经变成了一个属性，因此可以写在 filter**）的值小于 1 的分类过滤掉。因为 num_posts 的值小于 1 表示该分类下没有文章，没有文章的分类我们不希望它在页面中显示。关于 filter 函数以及查询表达式（双下划线）在分类与归档这一章已经讲过。

3、在模板中引用新增的属性

现在在 Category 列表中每一项都新增了一个 num_posts 属性记录该 Category 下的文章数量，我们就可以在模板中引用这个属性来显示分类下的文章数量了。

【templates/base.html】

```
<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>
```

也就是在模板中通过模板变量 {{ category.num_posts }} 显示 num_posts 的值。开启开发服务器，可以看到分类名后正确地显示了该分类下的文章数了，而没有文章分类则不会在分类列表中出现。

4、将 Annotate 用于其它关联关系

此外，annotate 方法不局限于用于本文提到的统计分类下的文章数，你也可以举一反三，**只要是两个 model 类通过 ForeignKey 或者 ManyToMany 关联起来，那么就可以使用 annotate 方法来统计数量**。比如下面这样一个标签系统：

```
class Post(models.Model):
    title = models.CharField(max_length=70)
    body = models.TextField()
    Tags = models.ManyToManyField('Tag')

    def __str__(self):
        return self.title

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

统计标签下的文章数：

```
from django.db.models.aggregates import Count
from blog.models import Tag

# Count 计算分类下的文章数，其接受的参数为需要计数的模型的名称
tag_list = Tag.objects.annotate(num_posts=Count('post'))
```

关于 annotate 方法官方文档的说明在这里：annotate。同时也建议了解了解 objects 下的其它操作数据库的方法，以便在遇到相关问题时知道去哪里查阅。