17-Django Pagination 简单分页

当博客上发布的文章越来越多时，通常需要进行分页显示，以免所有的文章都堆积在一个页面，影响用户体验。Django 内置的 Pagination 能够帮助我们实现简单的分页功能。

1、Paginator 类的常用方法

分页功能由 Django 内置的 **Paginator** 类提供。这个类位于 django/core/paginator.py，需要使用它时，只需在适当的地方导入这个类即可。下面的代码摘自 Django 的官方文档中 Pagination 的示例。可以在 python 命令行中执行：

```
from django.core.paginator import Paginator
```

只需**实例化一个 `Paginator` 对象，并在实例化时传入一个需要分页的列表对象，就可以得到分页后的对象数据。**

```
# 对 item_list 进行分页，每页包含 2 个数据。
>>> item_list = ['john', 'paul', 'george', 'ringo']
>>> p = Paginator(item_list, 2)
```

取特定页的数据：

```
# 取第 2 页的数据
>>> page2 = p.page(2)
>>> page2.object_list
['george', 'ringo']
```

查询特定页的当前页码数：

```
>>> page2.number
2
```

查看分页后的总页数：

```
>>> p.num_pages
2
```

查看某一页是否还有上一页，以及查询该页上一页的页码：

```
# 查询第二页是否还有上一页
>>> page2.has_previous()
True

# 查询第二页上一页的页码
>>> page2.previous_page_number()
1
```

查看某一页是否还有下一页，以及查询该页下一页的页码：

```
# 查询第二页是否还有下一页
>>> page2.has_next()
False

# 查询第二页下一页的页码
>>> page2.next_page_number()
django.core.paginator.EmptyPage: That page contains no results
```

更多方法和属性请参阅 Django Pagination 的官方文档。

2、用 Paginator 给文章列表分页（V）

使用上面的一些方法，我们可以实现一个类似于 Django 官方博客一样的简单分页效果，效果如下。

这里，Django 的官方文档中给出了一个在视图函数中对列表进行分页的示例，这个视图函数获取一个联系人列表并对其分页：

```
from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger
from django.shortcuts import render

def listing(request):
    contact_list = Contacts.objects.all()
    paginator = Paginator(contact_list, 25) # 每页显示 25 个联系人

    page = request.GET.get('page')
    try:
        contacts = paginator.page(page)
    except PageNotAnInteger:
        # 如果用户请求的页码号不是整数，显示第一页
        contacts = paginator.page(1)
    except EmptyPage:
        # 如果用户请求的页码号超过了最大页码号，显示最后一页
        contacts = paginator.page(paginator.num_pages)

    return render(request, 'list.html', {'contacts': contacts})
```

这就是在视图函数中使用分页的代码逻辑，你可以把它当做一个模板应用于自己的任何需要分页的视图函数。不过在我们的博客项目中，我们不必写这些代码了。回顾在类视图中的内容，我们已将视图函数转换成了类视图。而**类视图 ListView** 已经帮我们

写好了上述的分页逻辑，我们**只需通过指定 paginate_by 属性来开启分页功能即可**，即在类视图中指定 paginate_by 属性的值：

【blog/views.py】

```
class IndexView(ListView):
    model = Post
    template_name = 'blog/index.html'
    context_object_name = 'post_list'
    # 指定 paginate_by 属性后开启分页功能，其值代表每一页包含多少篇文章
    paginate_by = 10
```

这里我们设置了每 10 篇文章一页，当然你的测试数据可能不够。为了看到分页效果，你可以把这个数值减小。这样首页的文章列表就已经分好页了。

3、在模板中设置分页导航（T）

接下来便是在模板中设置分页导航，比如上一页、下一页的按钮，以及显示一些页面信息。我们这里设置和 Django 官方博客那样的分页导航样式（具体的样式见上图）。**ListView** 传递了以下和分页有关的**模板变量**供我们在模板中使用：

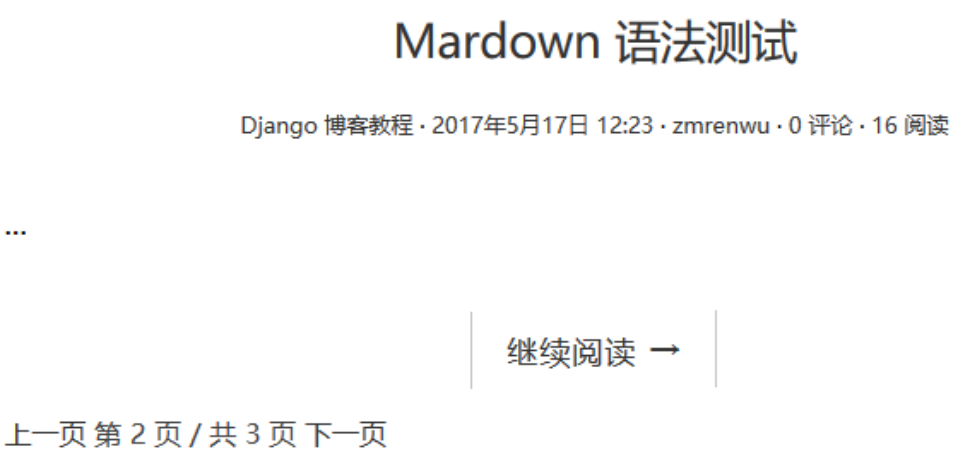
- **paginator** ，即 Paginator 的实例。
- **page_obj** ，当前请求页面分页对象。
- **is_paginated**，是否已分页。只有当分页后页面超过两页时才算已分页。
- **object_list**，请求页面的对象列表，**和 post_list 等价**。所以在模板中循环文章列表时可以选 post_list ，也可以选 object_list。

模板中使用示例：

【templates/blog/index.html】

```
{% if is_paginated %}
<div class="pagination-simple">
 <!-- 如果当前页还有上一页，显示一个上一页的按钮 -->
 {% if page_obj.has_previous %}
  <a href="?page={{ page_obj.previous_page_number }}">上一页</a>
 {% endif %}
 <!-- 显示当前页面信息 -->
 <span class="current">第 {{ page_obj.number }} 页 / 共 {{ paginator.num_pages }} 页</span>
 <!-- 如果当前页还有下一页，显示一个下一页的按钮 -->
 {% if page_obj.has_next %}
  <a href="?page={{ page_obj.next_page_number }}">下一页</a>
 {% endif %}
</div>
{% endif %}
```

其中 {{ }} 模板变量中的内容，其含义已在文章开头部分的 Paginator 类的常用方法中已有介绍。最终我们得到如下的分页效果：



当然这只是一个简单示例，分页导航处的视觉效果并不是很好看，你可以自行为其添加 CSS 样式使其看上去更加美观。