18-Django Pagination 完善分页

在上一章中，我们实现了一个简单的分页导航效果。但效果有点差强人意，我们只能点上一页和下一页的按钮进行翻页。比较完善的分页效果应该像下面这样，但想实现这样一种效果，Django Pagination 内置的 API 已无能为力。本文将通过拓展 Django Pagination 来实现下图这样比较完善的分页效果。

分页效果概述

一个比较完善的分页效果应该具有以下特性，就像上图展示的那样，很多网站都采用了类似这种的分页导航方式。

- 始终显示第一页和最后一页。
- 当前页码高亮显示。
- 显示当前页码前后几个连续的页码。
- 如果两个页码号间还有其它页码，中间显示省略号以提示用户。

1、拓展 Pagination

在此之前，我们已将首页文章列表的视图函数转为了类视图，并且使用了类视图 `ListView` 中已经为我们写好的分页代码来达到分页的目的（详情请查看文章开头处给出的链接）。为了实现如下所展示的分页效果，接下来就需要在 `ListView` 的基础上进一步拓展分页的逻辑代码。

先来分析一下导航条的组成部分，可以看到整个分页导航条其实可以分成 七个部分：

1. 第 1 页页码，这一页需要始终显示。
2. 第 1 页页码后面的省略号部分。但要注意如果第 1 页的页码号后面紧跟着页码号 2，那么省略号就不应该显示。
3. 当前页码的左边部分，比如这里的 3-6。
4. 当前页码，比如这里的 7。

5. 当前页码的右边部分，比如这里的 8-11。
6. 最后一页页码前面的省略号部分。但要注意如果最后一页的页码号前面跟着的页码号是连续的，那么省略号就不应该显示。
7. 最后一页的页码号。

因此我们的思路是，在视图里将以上七步中所需要的数据生成，然后传递给模板并在模板中渲染显示即可。整个视图的代码如下，由于代码比较长，所以代码实现的功能直接在代码块中注释，就不在文章中进一步说明了。推荐使用大屏幕阅读器获取更好的阅读体验。

【blog/views.py】

```
class IndexView(ListView):
    model = Post
    template_name = 'blog/index.html'
    context_object_name = 'post_list'
    paginate_by = 10

    def get_context_data(self, **kwargs):
        """
        在视图函数中将模板变量传递给模板是通过给 render 函数的 context 参数传递一个字典实现的，
        例如 render(request, 'blog/index.html', context={'post_list': post_list}),
        这里传递了一个 {'post_list': post_list} 字典给模板。
        在类视图中，这个需要传递的模板变量字典是通过 get_context_data 获得的，
        所以我们复写该方法，以便我们能够自己再插入一些我们自定义的模板变量进去。
        """

        # 首先获得父类生成的传递给模板的字典。
        context = super().get_context_data(**kwargs)

        # 父类生成的字典中已有 paginator、page_obj、is_paginated 这三个模板变量，
        # paginator 是 Paginator 的一个实例，
        # page_obj 是 Page 的一个实例，
        # is_paginated 是一个布尔变量，用于指示是否已分页。
        # 例如如果规定每页 10 个数据，而本身只有 5 个数据，其实就用不着分页，此时 is_paginated=False。
        # 关于什么是 Paginator，Page 类在 Django Pagination 简单分页：http://zmrenwu.com/post/34/ 中已有详细说明。
        # 由于 context 是一个字典，所以调用 get 方法从中取出某个键对应的值。
        paginator = context.get('paginator')
        page = context.get('page_obj')
        is_paginated = context.get('is_paginated')

        # 调用自己写的 pagination_data 方法获得显示分页导航条需要的数据，见下方。
        pagination_data = self.pagination_data(paginator, page, is_paginated)
```

```
        # 将分页导航条的模板变量更新到 context 中，注意 pagination_data 方法返回的也是一个字典。
        context.update(pagination_data)

        # 将更新后的 context 返回，以便 ListView 使用这个字典中的模板变量去渲染模板。
        # 注意此时 context 字典中已有了显示分页导航条所需的数据。
        return context

    def pagination_data(self, paginator, page, is_paginated):
        if not is_paginated:
            # 如果没有分页，则无需显示分页导航条，不用任何分页导航条的数据，因此返回一个空的字典
            return {}

        # 当前页左边连续的页码号，初始值为空
        left = []

        # 当前页右边连续的页码号，初始值为空
        right = []

        # 标示第 1 页页码后是否需要显示省略号
        left_has_more = False

        # 标示最后一页页码前是否需要显示省略号
        right_has_more = False

        # 标示是否需要显示第 1 页的页码号。
        # 因为如果当前页左边的连续页码号中已经含有第 1 页的页码号，此时就无需再显示第 1 页的页码号，
        # 其它情况下第一页的页码是始终需要显示的。
        # 初始值为 False
        first = False

        # 标示是否需要显示最后一页的页码号。
        # 需要此指示变量的理由和上面相同。
        last = False

        # 获得用户当前请求的页码号
        page_number = page.number

        # 获得分页后的总页数
        total_pages = paginator.num_pages

        # 获得整个分页页码列表，比如分了四页，那么就是 [1, 2, 3, 4]
        page_range = paginator.page_range

        if page_number == 1:
            # 如果用户请求的是第一页的数据，那么当前页左边的不需要数据，因此 left=[]（已默认为空）。
```

```
            # 此时只要获取当前页右边的连续页码号，
            # 比如分页页码列表是 [1, 2, 3, 4]，那么获取的就是 right = [2, 3]。
            # 注意这里只获取了当前页码后连续两个页码，你可以更改这个数字以获取更多页码。切片时如果溢出自动截断
            right = page_range[page_number:page_number + 2]

            # 如果最右边的页码号比最后一页的页码号减去 1 还要小，
            # 说明最右边的页码号和最后一页的页码号之间还有其它页码，因此需要显示省略号，通过 right_has_more 来指示。
            if right[-1] < total_pages - 1:
                right_has_more = True

            # 如果最右边的页码号比最后一页的页码号小，说明当前页右边的连续页码号中不包含最后一页的页码
            # 所以需要显示最后一页的页码号，通过 last 来指示
            if right[-1] < total_pages:
                last = True

        elif page_number == total_pages:
            # 如果用户请求的是最后一页的数据，那么当前页右边就不需要数据，因此 right=[]（已默认为空），
            # 此时只要获取当前页左边的连续页码号。
            # 比如分页页码列表是 [1, 2, 3, 4]，那么获取的就是 left = [2, 3]
            # 这里只获取了当前页码后连续两个页码，你可以更改这个数字以获取更多页码。
            left = page_range[(page_number - 3) if (page_number - 3) > 0 else 0:page_number - 1]

            # 如果最左边的页码号比第 2 页页码号还大，
            # 说明最左边的页码号和第 1 页的页码号之间还有其它页码，因此需要显示省略号，通过 left_has_more 来指示。
            if left[0] > 2:
                left_has_more = True

            # 如果最左边的页码号比第 1 页的页码号大，说明当前页左边的连续页码号中不包含第一页的页码，
            # 所以需要显示第一页的页码号，通过 first 来指示
            if left[0] > 1:
                first = True
        else:
            # 用户请求的既不是最后一页，也不是第 1 页，则需要获取当前页左右两边的连续页码号，
            # 这里只获取了当前页码前后连续两个页码，你可以更改这个数字以获取更多页码。
            left = page_range[(page_number - 3) if (page_number - 3) > 0 else 0:page_number - 1]
            right = page_range[page_number:page_number + 2]

            # 是否需要显示最后一页和最后一页前的省略号
            if right[-1] < total_pages - 1:
                right_has_more = True
```

```
            if right[-1] < total_pages:
                last = True

            # 是否需要显示第 1 页和第 1 页后的省略号
            if left[0] > 2:
                left_has_more = True
            if left[0] > 1:
                first = True

        data = {
            'left': left,
            'right': right,
            'left_has_more': left_has_more,
            'right_has_more': right_has_more,
            'first': first,
            'last': last,
        }

        return data
```

2、模板中设置分页导航

接下来便是在模板中设置分页导航了，将导航条的七个部分的数据一一展现即可，示例代码如下：

【templates/blog/index.html】

```
    {% if is_paginated %}
        <div class="pagination">
            <ul>
                {% if first %}
                    <li><a href="?page=1">1</a></li>
                {% endif %}
                {% if left %}
                    {% if left_has_more %}
                        <li><span>...</span></li>
                    {% endif %}
                    {% for i in left %}
                        <li><a href="?page={{ i }}">{{ i }}</a></li>
                    {% endfor %}
                {% endif %}
                <li class="current"><a
href="?page={{ page_obj.number }}">{{ page_obj.number }}</a></li>
                {% if right %}
                    {% for i in right %}
                        <li><a href="?page={{ i }}">{{ i }}</a></li>
                    {% endfor %}
                    {% if right_has_more %}
```

```
                        <li><span>...</span></li>
                    {% endif %}
                {% endif %}
                {% if last %}
                    <li><a
href="?page={{ paginator.num_pages }}">{{ paginator.num_pages }}</a></li>
                {% endif %}
            </ul>
        </div>
    {% endif %}
```

多添加几篇文章，在示例中就可以看到分页效果了。要使分页导航更加美观，通过设置其 CSS 样式即可：