6-真正的 Django 博客首页视图

在此之前我们已经编写了 Blog 的首页视图，并且配置了 URL 和模板，让 Django 能够正确地处理 HTTP 请求并返回合适的 HTTP 响应。不过我们仅仅在首页返回了一句话：欢迎访问我的博客。这是个 Hello World 级别的视图函数，我们需要编写真正的首页视图函数，当用户访问我们的博客首页时，他将看到我们发表的博客文章列表。

1、首页视图函数

上一节我们阐明了 Django 的开发流程。即首先配置 URL，把 URL 和相应的视图函数绑定，一般写在 urls.py 文件里，然后在工程的 urls.py 文件引入。其次是编写视图函数，视图中需要渲染模板，我们也在 settings.py 中进行了模板相关的配置，让 Django 能够找到需要渲染的模板。最后把渲染完成的 HTTP 响应返回就可以了。相关的配置和准备工作都在之前完成了，这里我们只需专心编写视图函数，让它实现我们想要的功能即可。

首页的视图函数其实很简单，代码像这样：

【blog/views.py】

```
from django.shortcuts import render
from .models import Post

def index(request):
    post_list = Post.objects.all().order_by('-created_time')
    return render(request, 'blog/index.html', context={'post_list': post_list})
```

我们曾经在前面的章节讲解过模型管理器 `objects` 的使用。这里我们使用 `all()` 方法从数据库里获取了全部的文章，存在了 `post_list` 变量里。`all` 方法返回的是一个 `QuerySet`（可以理解成一个类似于列表的数据结构），由于通常来说博客文章列表是按文章发表时间倒序排列的，即最新的文章排在最前面，所以我们紧接着调用了 `order_by` 方法对这个返回的 queryset 进行排序。排序依据的字段是 `created_time`，即文章的创建时间。- 号表示逆序，如果不加 - 则是正序。 接着如之前所做，我们渲染了 blog\index.html 模板文件，并且把包含文章列表数据的 `post_list` 变量传给了模板。

2、处理静态文件

我们的项目使用了从网上下载的一套博客模板。这里面除了 HTML 文档外，还包含了一些 CSS 文件和 JavaScript 文件以让网页呈现出我们现在看到的样式。同样我们需要对 Django 做一些必要的配置，才能让 Django 知道如何在开发服务器中引入这些 CSS 和 JavaScript 文件，这样才能让博客页面的 CSS 样式生效。

按照惯例，我们把 CSS 和 JavaScript 文件放在 blog 应用的 static\ 目录下。因此，先在 blog 应用下建立一个 static 文件夹。同时，为了避免和其它应用中的 CSS 和 JavaScript 文件命名冲突（别的应用下也可能有和 blog 应用下同名的 CSS 、JavaScript 文件），我们再在 static\ 目录下建立一个 blog 文件夹，把下载的博客模板中的 css 和 js 文件夹连同里面的全部文件一同拷贝进这个目录。最终我们的 blog 应用目录结构应该是这样的：

```
blog\
  __init__.py
  static\
    blog\
      css\
        .css 文件...
      js\
        .js 文件...
  admin.py
  apps.py
  migrations\
    __init__.py
  models.py
  tests.py
  views.py
```

CSS 样式文件的路径在 link 标签的 href 属性里，而 JavaScript 文件的路径在 script 标签的 src 属性里。我们需要把它们改成正确的路径。把代码改成下面样子，正确地引入 static 文件下的 CSS 和 JavaScript 文件：

【templates/blog/index.html】

```
{% load staticfiles %}

<!DOCTYPE html>

<html>

<head>

    <title>Black &amp;amp; White</title>



    <!-- meta -->
```

```
    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1">



    <!-- css -->

    <link rel="stylesheet"
href="http://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min
.css">

    <link rel="stylesheet" href="{% static
'blog/css/bootstrap.min.css' %}">

    <link rel="stylesheet" href="{% static 'blog/css/pace.css' %}">

    <link rel="stylesheet" href="{% static
'blog/css/custom.css' %}">



    <!-- js -->
                    source
    <script src="{% static 'blog/js/jquery-
2.1.3.min.js' %}"></script>

    <script src="{% static 'blog/js/bootstrap.min.js' %}"></script>

    <script src="{% static 'blog/js/pace.min.js' %}"></script>

    <script src="{% static
'blog/js/modernizr.custom.js' %}"></script>

</head>
```

我们把引用路径放在了一个奇怪的符号里，例如：href="{% static 'blog/css/bootstrap.min.css' %}"。用 {% %} 包裹起来的叫做模板标签。我们前面说过用 {{ }} 包裹起来的叫做模板变量，其作用是在最终渲染的模板里显示由视图函数传过来的变量值。而这里我们使用的模板标签的功能则类似于函数，例如这里的 static 模板标签，它把跟在后面的字符串 'css/bootstrap.min.css' 转换成正确的文件引入路径。这样 css 和 js 文件才能被正确加载，样式才能正常显示。

为了能在模板中使用 {% static %} 模板标签，别忘了在最顶部 {% load staticfiles %} 。**static 模板标签位于 staticfiles 模块中，只有通过 load 模板标签将该模块引入后，才能在模板中使用 {% static %} 标签。**

替换完成后你可以刷新页面并看看网页的源代码，看一看 {% static %} 模板标签在页面渲染后究竟被替换成了什么样的值。例如我们可以看到

```
<link rel="stylesheet" href="{% static 'blog/css/pace.css' %}">
```

这一部分最终在浏览器中显示的是：

```
<link rel="stylesheet" href="/static/blog/css/pace.css">
```

这正是 pace.css 文件所在的路径，其它的文件路径也被类似替换。可以看到 {% static %} 标签的作用实际就是把后面的字符串加了一个 /static/ 前缀，比如 `{% static 'blog/css/pace.css' %}` 最终渲染的值是 `/static/blog/css/pace.css`。而 /static/ 前缀是我们在 settings.py 文件中通过 `STATIC_URL = '/static/'` 指定的。事实上，如果我们直接把引用路径写成 `/static/blog/css/pace.css` 也是可以的，那么为什么要使用 {% static %} 标签呢？想一下，目前 URL 的前缀是 /static/，如果哪一天因为某些原因，我们需要把 /static/ 改成 /resource/，如果你是直接写的引用路劲而没有使用 static 模板标签，那么你可能需要改 N 个地方。如果你使用了 static 模板标签，那么只要在 settings.py 处改一个地方就可以了，即把 `STATIC_URL = '/static/'` 改成 `STATIC_URL = '/resource/'`。

有时候按 F5 刷新后页面还是很乱，这可能是因为浏览器缓存了之前的结果。按 Shift + F5（有些浏览器可能是 Ctrl + F5）强制刷新浏览器页面即可。

注意这里有一个 CSS 文件的引入

```
<link rel="stylesheet"
href="http://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css">
```

我们没有使用模板标签，因为这里的引用的文件是一个外部文件，不是我们项目里 static\blog\css\ 目录下的文件，因此无需使用模板标签。

正确引入了静态文件后样式显示正常了。

修改模板

（1）设置 header

```
<body>

<div class="container">

    <header id="site-header">

        <div class="row">

            <div class="col-md-4 col-sm-5 col-xs-8">

                <div class="logo">

                    <h1><a href="index.html"><b>Black</b> &amp;
White</a></h1>

                </div>

            </div><!-- col-md-4 -->

            <div class="col-md-8 col-sm-7 col-xs-4">

                <nav class="main-nav" role="navigation">

                    <div class="navbar-header">

                        <button type="button" id="trigger-overlay"
class="navbar-toggle">

                            <span class="ion-navicon"></span>

                        </button>

                    </div>



                    <div class="collapse navbar-collapse" id="bs-
example-navbar-collapse-1">

                        <ul class="nav navbar-nav navbar-right">

                            <li class="cl-effect-11"><a
href="index.html" data-hover="首页">首页</a></li>
```

```
                            <li class="cl-effect-11"><a href="full-
width.html" data-hover="博客">博客</a></li>

                            <li class="cl-effect-11"><a
href="about.html" data-hover="关于">关于</a></li>

                            <li class="cl-effect-11"><a
href="contact.html" data-hover="联系">联系</a></li>

                        </ul>

                    </div><!-- /.navbar-collapse -->

                </nav>

                <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="#">

                            <input type="search" placeholder="搜索"
required>

                            <button type="submit"><span class="ion-
ios-search-strong"></span></button>

                        </form>

                    </div>

                </div>

            </div><!-- col-md-8 -->

        </div>

    </header>

</div>
```

（2）设置 main

```
<div class="content-body">

    <div class="container">

        <div class="row">

            <main class="col-md-8">

                {% for post in post_list %}

                    <article class="post post-{{ post.pk }}">

                        <header class="entry-header">

                            <h1 class="entry-title">

                                <a
href="single.html">{{ post.title }}</a>

                            </h1>

                            <div class="entry-meta">

                                <span class="post-category"><a
href="#">{{ post.category.name }}</a></span>

                                <span class="post-date"><a
href="#"><time class="entry-date"


datetime="{{ post.created_time }}">{{ post.created_time }}</time></a
></span>

                                <span class="post-author"><a
href="#">{{ post.author }}</a></span>

                                <span class="comments-link"><a
href="#">4 评论</a></span>

                                <span class="views-count"><a
href="#">588 阅读</a></span>

                            </div>

                        </header>

                        <div class="entry-content clearfix">

                            <p>{{ post.excerpt }}</p>

                            <div class="read-more cl-effect-14">
```

```
                                <a href="#" class="more-link">继续阅
读 <span class="meta-nav">→</span></a>
                            </div>
                        </div>
                    </article>
                {% empty %}
                    <div class="no-post">暂时还没有发布的文章！</div>
                {% endfor %}

                <!-- 简单分页效果
                <div class="pagination-simple">
                    <a href="#">上一页</a>
                    <span class="current">第 6 页 / 共 11 页</span>
                    <a href="#">下一页</a>
                </div>
                -->
                <div class="pagination">
                    <ul>
                        <li><a href="">1</a></li>
                        <li><a href="">...</a></li>
                        <li><a href="">4</a></li>
                        <li><a href="">5</a></li>
                        <li class="current"><a href="">6</a></li>
                        <li><a href="">7</a></li>
                        <li><a href="">8</a></li>
```

```
                        <li><a href="">...</a></li>
                        <li><a href="">11</a></li>
                    </ul>
                </div>
            </main>
```

（3）设置 aside

```
            <aside class="col-md-4">
                <div class="widget widget-recent-posts">
                    <h3 class="widget-title">最新文章</h3>
                    <ul>
                        <li>
                            <a href="#">Django 博客开发入门教程：前言
</a>
                        </li>
                        <li>
                            <a href="#">Django 博客使用 Markdown 自动
生成文章目录</a>
                        </li>
                        <li>
                            <a href="#">部署 Django 博客</a>
                        </li>
                    </ul>
                </div>
                <div class="widget widget-archives">
```

```
                    <h3 class="widget-title">归档</h3>
                    <ul>
                        <li>
                            <a href="#">2017 年 5 月</a>
                        </li>
                        <li>
                            <a href="#">2017 年 4 月</a>
                        </li>
                        <li>
                            <a href="#">2017 年 3 月</a>
                        </li>
                    </ul>
                </div>

                <div class="widget widget-category">
                    <h3 class="widget-title">分类</h3>
                    <ul>
                        <li>
                            <a href="#">Django 博客教程 <span
class="post-count">(13)</span></a>
                        </li>
                        <li>
                            <a href="#">Python 教程 <span
class="post-count">(11)</span></a>
                        </li>
```

```
                        <li>
                            <a href="#">Django 用户认证 <span
class="post-count">(8)</span></a>
                        </li>
                    </ul>
                </div>

                <div class="widget widget-tag-cloud">
                    <h3 class="widget-title">标签云</h3>
                    <ul>
                        <li>
                            <a href="#">Django</a>
                        </li>
                        <li>
                            <a href="#">Python</a>
                        </li>
                        <li>
                            <a href="#">Java</a>
                        </li>
                        <li>
                            <a href="#">笔记</a>
                        </li>
                        <li>
                            <a href="#">文档</a>
                        </li>
                        <li>
```

```
                            <a href="#">AngularJS</a>
                        </li>
                        <li>
                            <a href="#">CSS</a>
                        </li>
                        <li>
                            <a href="#">JavaScript</a>
                        </li>
                        <li>
                            <a href="#">Snippet</a>
                        </li>
                        <li>
                            <a href="#">jQuery</a>
                        </li>
                    </ul>
                </div>
                <div class="rss">
                    <a href=""><span class="ion-social-rss-
outline"></span> RSS 订阅</a>
                </div>
            </aside>
        </div>
    </div>
</div>
```

（4）设置 footer

```
<footer id="site-footer">
```

```
    <div class="container">
        <div class="row">
            <div class="col-md-12">
                <p class="copyright">&copy 2017 - 采集自<a
href="http://www.cssmoban.com/"
target="_blank" title="模板之家">模板之家</a>
                </p>
            </div>
        </div>
    </div>
</footer>
```

（5）设置移动按钮

```
<!-- Mobile Menu -->
<div class="overlay overlay-hugeinc">
    <button type="button" class="overlay-close"><span class="ion-
ios-close-empty"></span></button>
    <nav>
        <ul>
            <li><a href="index.html">首页</a></li>
            <li><a href="full-width.html">博客</a></li>
            <li><a href="about.html">关于</a></li>
            <li><a href="contact.html">联系</a></li>
        </ul>
```

```
    </nav>
</div>

<script src="{% static 'blog/js/script.js' %}"></script>

</body>
</html>
```

目前我们看到的只是模板中预先填充的一些数据，我们得让它显示从数据库中获取的文章数据。下面来稍微改造一下模板：

在模板 index.html 中你会找到一系列 article 标签：
templates/blog/index.html

```
...
<article class="post post-1">
 ...
</article>

<article class="post post-2">
 ...
</article>

<article class="post post-3">
 ...
</article>
...
```

这里面包裹的内容显示的就是文章数据了。我们前面在视图函数 index 里给模板传了一个 `post_list` 变量，它里面包含着从数据库中取出的文章列表数据。就像 Python 一样，我们可以在模板中循环这个列表，把文章一篇篇循环出来，然后一篇篇显示文章的数据。要在模板中使用循环，需要使用到前面提到的模板标签，这次使用 {% for %} 模板标签。将 index.html 中多余的 article 标签删掉，只留下一个 article 标签，然后写上下列代码：
templates/blog/index.html

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

```
{% endfor %}
...
```

可以看到语法和 Python 的 for 循环类似，只是被 {% %} 这样一个模板标签符号包裹着。{% empty %} 的作用是当 post_list 为空，即数据库里没有文章时显示 {% empty %} 下面的内容，最后我们用 {% endfor %} 告诉 Django 循环在这里结束了。

你可能不太理解模板中的 post 和 post_list 是什么。post_list 是一个 QuerySet（类似于一个列表的数据结构），其中每一项都是之前定义在 blog\models.py 中的 Post 类的实例，且每个实例分别对应着数据库中每篇文章的记录。因此我们循环遍历 post_list ，每一次遍历的结果都保存在 post 变量里。所以我们使用模板变量来显示 post 的属性值。例如这里的 {{ post.pk }}（pk 是 primary key 的缩写，即 post 对应于数据库中记录的 id 值，该属性尽管我们没有显示定义，但是 Django 会自动为我们添加）。

现在我们可以在循环体内通过 post 变量访问单篇文章的数据了。分析 article 标签里面的 HTML 内容，h1 显示的是文章的标题，

```
<h1 class="entry-title">
  <a href="single.html">Adaptive Vs. Responsive Layouts And Optimal Text Readability</a>
</h1>
```

我们把标题替换成 post 的 title 属性值。注意要把它包裹在模板变量里，因为它最终要被替换成实际的 title 值。

```
<h1 class="entry-title">
  <a href="single.html">{{ post.title }}</a>
</h1>
```

下面这 5 个 span 标签里分别显示了分类（category）、文章发布时间、文章作者、评论数、阅读量。

```
<div class="entry-meta">
 <span class="post-category"><a href="#">Django 博客教程</a></span>
 <span class="post-date"><a href="#"><time class="entry-date"
                        datetime="2012-11-09T23:15:57+00:00">2017 年 5 月 11 日
</time></a></span>
 <span class="post-author"><a href="#">追梦人物</a></span>
 <span class="comments-link"><a href="#">4 评论</a></span>
 <span class="views-count"><a href="#">588 阅读</a></span>
</div>
```

再次替换掉一些数据，由于评论数和阅读量暂时没法替换，因此先留着，我们在之后实现了这些功能后再来修改它，目前只替换分类、文章发布时间、文章作者：

```
<div class="entry-meta">
 <span class="post-category"><a href="#">{{ post.category.name }}</a></span>
 <span class="post-date"><a href="#"><time class="entry-date"

datetime="{{ post.created_time }}">{{ post.created_time }}</time></a></span>
```

```
 <span class="post-author"><a href="#">{{ post.author }}</a></span>
 <span class="comments-link"><a href="#">4 评论</a></span>
 <span class="views-count"><a href="#">588 阅读</a></span>
</div>
```

这里 p 标签里显示的是摘要

```
<div class="entry-content clearfix">
 <p>免费、中文、零基础，完整的项目，基于最新版 Django 1.10 和 Python 3.5。带你从零开始一步步开发属于自己的博客网站，帮助你以最快的速度掌握 Django
  开发的技巧...</p>
 <div class="read-more cl-effect-14">
  <a href="#" class="more-link">继续阅读 <span class="meta-nav">→</span></a>
 </div>
</div>
```

替换成 post 的摘要：

```
<div class="entry-content clearfix">
 <p>{{ post.excerpt }}</p>
 <div class="read-more cl-effect-14">
  <a href="#" class="more-link">继续阅读 <span class="meta-nav">→</span></a>
 </div>
</div>
```

再次访问首页，它显示：暂时还没有发布的文章。接下来我们就实际写几篇文章保存到数据库里，看看显示的效果究竟如何。