内容概要

- 个人站点页面搭建
- 侧边栏展示功能
  - 标签
  - 分类
  - 日期
- 侧边栏筛选功能
- 将侧边栏制作成inclusion_tag

个人站点

```
# 全是每个用户都可以有自己的站点样式
<link rel="stylesheet" href="/media/css/{{ blog.site_theme }}/">



id      content               create_time                  month
1   111         2020-11-11     2020-11
2   222         2020-11-12     2020-11
3   333         2020-11-13     2020-11
4   444         2020-11-14     2020-11
5   555         2020-11-15     2020-11
"""
django官网提供的一个orm语法
    from django.db.models.functions import TruncMonth
-官方提供
            from django.db.models.functions import TruncMonth
            Sales.objects
            .annotate(month=TruncMonth('timestamp'))  # Truncate to month and add to select
list
            .values('month')  # Group By month
            .annotate(c=Count('id'))  # Select the count of the grouping
            .values('month', 'c')  # (might be redundant, haven't tested) select month and
count


时区问题报错
TIME_ZONE = 'Asia/Shanghai'
USE_TZ = True
"""
```

侧边栏筛选功能

```
https://www.cnblogs.com/jason/tag/Python/                标签
https://www.cnblogs.com/jason/category/850028.html 分类
https://www.cnblogs.com/jason/archive/2016/10.html 日期
```

```
https://www.cnblogs.com/jason/tag/1/                标签
https://www.cnblogs.com/jason/category/1 分类
https://www.cnblogs.com/jason/archive/2020-11/ 日期
```

```python
def site(request,username,**kwargs):
    """
    :param request:
    :param username:
    :param kwargs: 如果该参数有值 也就意味着需要对article_list做额外的筛选操作
    :return:
    """
    # 先校验当前用户名对应的个人站点是否存在
    user_obj = models.UserInfo.objects.filter(username=username).first()
    # 用户如果不存在应该返回一个404页面
    if not user_obj:
        return render(request,'errors.html')
    blog = user_obj.blog
    # 查询当前个人站点下的所有的文章
    article_list = models.Article.objects.filter(blog=blog)  # queryset对象 侧边栏的筛选其实就是对article_list再进一步筛选
    if kwargs:
        # print(kwargs)  # {'condition': 'tag', 'param': '1'}
        condition = kwargs.get('condition')
        param = kwargs.get('param')
        # 判断用户到底想按照哪个条件筛选数据
        if condition == 'category':
            article_list = article_list.filter(category_id=param)
        elif condition == 'tag':
            article_list = article_list.filter(tags__id=param)
        else:
            year,month = param.split('-')  # 2020-11  [2020,11]
            article_list = article_list.filter(create_time__year=year,create_time__month=month)


    # 1 查询当前用户所有的分类及分类下的文章数
    category_list = models.Category.objects.filter(blog=blog).annotate(count_num=Count('article__pk')).values_list('name','count_num','pk')
    # print(category_list)  # <QuerySet [('jason的分类一', 2), ('jason的分类二', 1), ('jason的分类三', 1)]>

    # 2 查询当前用户所有的标签及标签下的文章数
    tag_list = models.Tag.objects.filter(blog=blog).annotate(count_num=Count('article__pk')).values_list('name','count_num','pk')
    # print(tag_list)  # <QuerySet [('tank的标签一', 1), ('tank的标签二', 1), ('tank的标签三', 2)]>

    # 3 按照年月统计所有的文章
    date_list = models.Article.objects.filter(blog=blog).annotate(month=TruncMonth('create_time')).values('month').annotate(count_num=Count('pk')).values_list('month','count_num')
    # print(date_list)

    return render(request,'site.html',locals())
```