# 文章详情页面

## 😀 1、功能需求：

- 文章底部可以对文章点赞点踩。
- 作者不可以给自己的文章进行点赞点踩。
- 未登录进行点赞点踩会显示“请登录字样”
  - 点击“请登录”会跳转登录页面
- 文章底部可以对文章进行评论。
  - 已登录，会显示评论框
  - 未登录不显示评论框，并会显示注册，登录字样。点击即可跳转对应页面。
- 点击提交评论，会在评论楼底部显示该评论，并且是加粗效果。刷新页面后，该评论恢复正常。
- 点击回复，会自动聚焦到评论框内，并且在内显示@评论人。

## 2、技术实现-点赞点踩：

- 前端：
  - 给点赞标签的属性设为class="diggit action"
  - 给点踩标签的属性设为class="buryit action"
  - 信息提示标签属性设为class="digword"
  - 然后给".action"绑定点击事件，然后通过半段点击的标签class属性是否含有"diggit"来判断此点击时点赞还是点踩。
  - 通过ajax将相关信息传给后端，比如点赞文章的主键，让后端知道给哪篇文章点了赞。
  - 因为点赞是针对于已登录用户的操作，所以后端可以自动识别点赞者是否登录，如果登录，默认就是他点的赞。
  - 点击后，后端会将信息封装在"msg"，中返回回来。然后前端，将msg信息显示出来即可。
    - $(".action").click(function(){
      - let isUp=$(this).hasClass("diggit"); //判断此标签class是否含有"diggit",如果含有返回true，否则返回fasle.
      - let div=$(this);

- $.ajax({
    - url:"up_and_down",
    - type:"post",
    - data{
        - "article":文章.pk, //文章主键值
        - "isUp": isUp, //告诉后端是点赞还是点踩
        - "csrfmiddlewaretoken":{{ csrf_token},
    - },
    - success:function(args){
        - if (args.code==1000){ //首先判断状态码，判断点赞或点踩是否成功，1000表示成功
            - $(".digword").text(args.msg); //显示提示信息
            - let oldnum=div.children().text(); //获取原来的点赞数,返回值是一个字符串
            - div.childred().text(Number(oldnum)+1); //在原来的基础上+1
        - }else{
            - $("."diggword").text(args.msg); //显示报错信息
- })
- })

- **后端：**
    - 获取前端传来的信息：
        - article_id=request.POST.get("article_id")
        - isUp=request.POST.get("isUp")
        - isUp=json.loads(isUp)
    - 判断用户是否登录：request.is_authenticated
        - 如果False，点赞失败，msg="用户未登录"
    - 获取点赞文章的作者：article_author
        - 如果文章作者和点赞者是一个人：request,user==article
            - 点赞失败，msg="不可以给自己的文章点赞"
    - 判断当前作者是否已经给当前文章点过赞：is_click
        - 如果已经点过赞，点赞失败，msg="已经点过赞"

- 判断isUp是true还是False，true则是点赞，false则是点踩
- 更新数据库
- reteurn JsonResponse(back_dic)

### 3、技术实现-文章评论：

- 前端：
  - 评论楼渲染使用BootStrap框架的列表组实现class="list-group"
  - 评论框使用<textarea>标签。
  - 如果用户登录显示评论框，否则显示注册登录字样，并且点击可以实现页面跳转。
    - 跳转的url使用了url别名：{% url "register" %},{% url "login" %}
  - 给提交按钮绑定点击事件
    - 点击或获取用户评论内容，然后使用ajax传给后端
    - 并对评论内容做一定的修改，然后将其临时渲染到评论楼，当页面刷新时，后端会将次评论传给前端
- 后端：
  - 获取前端传入的数据，然后添加到数据库。
  - 返回状态信息：return JsonResponse(back_dic)

### 4、技术实现-回复评论：

- 前端：
  - 评论楼渲染时，给回复标签添加两个自定义的属性：
    - username: #标注评论人的姓名
    - comment_id: #标注评论内容的主键
  - 给回复标签绑定点击事件
  - 然后获取评论人的名称，以及其他信息
  - 然偶将相关信息，存放到评论框中，并给评论框获取焦点。
- 后端：
  - 然后填写评论，提交即可，具体内容和评论后端一样，这里不再阐述。

# 内容回顾

## 1）侧边栏制作

```
"""
1.当一个页面的局部需要再多个页面使用并且还需要传参数

自定义inclusion_tag步骤
    1.在应用下创建名字必须叫templatetags文件夹
    2.文件夹内创建任意名称的py文件
    3.py文件内先书写固定的两行代码
    from django import template
    register = template.Library()

    @register.inclusion_tag(left_menu.html)
    def index():
 准备上述页面需要的数据
        return locals()
"""
```

## 2）点踩点赞

```
# 前端页面
 1.拷贝博客园点赞点踩前端样式
    html代码 + css代码
  2.如何判断用户到底点击了哪个图标
 恰巧页面上只有两个图标，所以给两个图标标签添加一个公共的样式类
 然后给这个样式类绑定点击事件
 再利用this指代的就是当前被操作对象 利用hasClass判断是否有某个特定的类属性，从而判断出到底是两个图标中的哪一个
  3.书写ajax代码朝后端提交数据

  4.后端逻辑书写完毕之后 前端针对点赞点踩动作实现需要动态展示提示信息
    1.前端点赞点踩数字自增1 需要注意数据类型的问题
          Number(old_num) + 1
      2.用户没有登陆 需要展示没有登陆提示 并且登陆可以点击跳转
        html()
        |safe
        mark_safe()
# 后端逻辑
 1.先判断用户是否登陆
    request.user.authenticated()
  2.再判断当前文字是否是当前用户自己写的
 通过文章主键值获取文章对象
 之后利用orm查询获取文章对象对应的用户对象与request.user比对
  3.判断当前用户是否已经给当前文章点了
 利用article_obj文章对象和request.user用户对象去点赞点踩表中筛选数据如果有数据则点过没有则没点
  4.操作数据库 需要注意要同时操作两张表
   # 前端发送过来的是否点赞是一个字符串 需要你自己转成布尔值或者用字符串判断
```

```
    is_up = json.loads(is_up)
    F模块
 """ 总结:在书写较为复杂的业务逻辑的时候，可以先按照一条线书写下去 之后再去弥补其他线路情况 类似于先走树的主干 之后再分散 """
```

## 3）评论

```
# 先写根评论
 先吧整体的评论功能跑通 再去填补
  1.书写前端获取用户评论的标签
 可能点赞点踩有浮动带来的影响
        clearfix
  2.点击评论按钮发送ajax请求

  3.后端针对评论单独开设url处理
 后端逻辑其实非常的简单非常的少

  4.针对根评论涉及到前端的两种渲染方式
   1.DOM操作临时渲染评论楼
 需要用到模版字符串
        // 临时渲染评论楼
                        let userName = '{{ request.user.username }}';
                        let temp = `
                        <li class="list-group-item">

                            <span>${userName}</span>
                            <span><a href="#" class="pull-right">回复</a></span>
                            <div>
                                ${conTent}
                            </div>
                            </li>
                        `
                        // 将生成好的标签添加到ul标签内
                        $('.list-group').append(temp);
    2.页面刷新永久(render)渲染
 后端直接获取当前文章对应的所有评论 传递给html页面即可
 前端利用for循环参考博客园评论楼样式渲染评论

    3.评论框里面的内容需要清空
# 再考虑子评论
 从回复按钮入手
 点击回复按钮发生了哪些事
     1.评论框自动聚焦      .focus()
      2.评论框里面自动添加对应评论的评论人姓名
        @username\n
 思考:
```

```
        1.根评论和子评论点的是同一个按钮
        2.根评论和子评论的区别
    其实之前的ajax代码只需要添加一个父评论id即可

    点击回复按钮之后 我们应该获取到根评论对应的用户名和主键值
    针对主键值 多个函数都需要用 所以用全局变量的形式存储

    针对子评论内容 需要切割出不是用户写的 @username\n
                        // 获取用户评论的内容
                let conTent = $('#id_comment').val();
                // 判断当前评论是否是子评论 如果是 需要将我们之前手动渲染的@username去除
                if(parentId){
                    // 找到\n对应的索引 然后利用切片 但是前片顾头不顾尾 所以索引+1
                    let indexNum = conTent.indexOf('\n') + 1;
                    conTent = conTent.slice(indexNum)  // 将indexNum之前的所有数据切除 只保留后
    面的部分
                }

    后端parent字段本来就可以为空，所以传不传值都可以直接存储数据

    前端针对子评论再渲染评论楼的时候需要额外的判断
            {% if comment.parent_id %}
                <p>@{{ comment.parent.user.username }}</p>
          {% endif %}
                {{ comment.content }}
    前端parentId字段每次提交之后需要手动清空
```

---

---

# urls.py

```
#点赞点踩
re_path(r"^up_or_down/",views.up_or_down),
# 评论
re_path(r"^comment/",views.comment),
```

# article_detail.html

```
{% extends "base.html" %}
```

```
{% block css %}
<style>
/* ****************************点赞点踩样式****************************************** */
    #div_digg {
        float: right;
        margin-bottom: 10px;
        margin-right: 30px;
        font-size: 12px;
        width: 128px;
        text-align: center;
        margin-top: 10px;
    }

    .diggit {
        float: left;
        width: 46px;
        height: 52px;
        background: url(/static/img/upup.gif) no-repeat;
        text-align: center;
        cursor: pointer;
        margin-top: 2px;
        padding-top: 5px;
    }

    .buryit {
        float: right;
        margin-left: 20px;
        width: 46px;
        height: 52px;
        background: url(/static/img/downdown.gif) no-repeat;
        text-align: center;
        cursor: pointer;
        margin-top: 2px;
        padding-top: 5px;
    }

    .diggword {
        margin-top: 5px;
        margin-left: 0;
        font-size: 12px;
        color: #808080;
    }
</style>
{% endblock css %}
```

```
{% block content %}
    <!--****************************************文章主要内容
****************************************-->
    <h1>{{ article_obj.title }}</h1>
    <div class="article_content">
        {{  article_obj.content|safe }}
    </div>

    <!--**********************************点赞点踩
**********************************************-->
<!--点赞点踩样式开始-->
<div class="clearfix">
    <div id="div_digg">
        <div class="diggit action" >
            <span class="diggnum" id="digg_count">{{ article_obj.up_num }}</span>
        </div>
        <div class="buryit action">
            <span class="burynum" id="bury_count">{{ article_obj.down_num }}</span>
        </div>
        <div class="clear"></div>
        <div class="diggword" id="digg_tips" style="color: red">
        </div>
    </div>
</div>
    <!--点赞点踩样式结束-->

<!--**********************************评 论 楼**************************************-
->
    <!--评论楼渲染开始-->
    <div class="">
        <ul class="list-group">
            {% for comment in comment_list %}
            <li class="list-group-item" >
            <span>#{{ forloop.counter }}楼</span>
            <span>{{ comment.comment_time|date:"Y-m-d h:i:s" }}</span>
            <span>{{ comment.user.username }}</span>
            <span><a  class="pull-right replay" username="{{ request.user.username }}"
comment_id="{{ comment.pk }}">回复</a></span>
            <div>
                <!--判断当前评论是否是子评论，如果是子评论需要渲染对应的评论人的人名-->
                {% if comment.parent_id %}
                    <p>@{{ comment.parent.user.username }}</p>
                {% endif %}
                {{ comment.content }}
            </div>
         </li>
        {% endfor %}
```

```
        </ul>
    </div>
    <!--评论楼渲染结束-->

<!--**********************************文章评论**************************************-->
    <!--文章评论样式开始-->
    <div class="">
    <p><span class="glyphicon glyphicon-comment">发表评论</span></p>
        {% if request.user.is_authenticated %}
    <div class="">
        <textarea name="comment" id="id_comment" cols="60" rows="10"></textarea>
    </div>
            {% else %}
            <li><a href="{% url "reg" %}">注册</a></li> <!--使用的是url别名-->
            <li><a href="{% url "login" %}">登录</a></li>
        {% endif %}
        <button class="btn btn-primary" id="id_submit">提交评论</button>
        <span style="color: red" id="error"></span>    <!--提示信息-->
    </div>
    <!--文章评论样式开始-->
{% endblock %}

{% block js %}
<script>
/*
 ****************************************** 点 赞 点 踩
******************************************
 */
    //给所有的action绑定事件
    $(".action").click(function () {
        let isUp=$(this).hasClass("diggit");  //判断该标签class属性是否含有"diggit"这个值
        let $div=$(this);
        //朝后端发送ajax请求
        $.ajax({
            url:"/up_or_down/",
            type:"post",
            data:{
                "article_id":"{{ article_obj.pk }}",
                "isUp":isUp,
                "csrfmiddlewaretoken":"{{ csrf_token }}"
            },
            success:function (args) {
                if(args.code==1000){
                    $("#digg_tips").text(args.msg);
                    //将前端的数
                    //先获取到之前的数字
                    let oldNum=$div.children().text();  //文本 字符类型
```

```
                    $div.children().text(Number(oldNum)+1);
                }else{
                    $("#digg_tips").html(args.msg);
                }

            }
        })
    });

/*
 **********************************文章评论*******************************************
 */
    //设置一个全局的parentId字段
    let parentId=null;
    //用户点击评论给后盾发送ajax请求
    $("#id_submit").click(function () {
        //获取用户评论的内容
        let conTent=$("#id_comment").val();
        //判断当前评论是否是子评论 如果是我们要将手动渲染的@username去掉
        if(parentId){
            //先找到\n对应的所有值， 然后利用切片 但是切片顾前不顾尾 所以索引+1
            let indexNum=conTent.indexOf("\n")+1;
            conTent=conTent.slice(indexNum)     //将indxNum之前的所有数据切除，只保留后面的部分
        }

        $.ajax({
            url:"/comment/",
            type:"post",
            data:{
              "article_id":"{{ article_obj.pk }}",
                "parent_id":parentId,   //如果parentId没有值，置为null
              "content":conTent,
                "csrfmiddlewaretoken":"{{ csrf_token }}"
            },
            success:function (args) {
                if(args.code==1000){
                    $("#error").text(args.msg);
                    //1、将评论框里的内容清空
                    $("#id_comment").val("");
                    //2、临时渲染评论楼
                    let userName="{{ request.user.username }}";
                    let t=  `
                       <li class="list-group-item" >
                        <b>${userName}</b>
                        <span><a href="" class="pull-right">回复</a></span>pi
                        <div>
```

```
                            <b>${conTent}</b>
                        </div>
                        </li>
                    `;
                    //将生成好的标签，添加到ul标签内
                    $(".list-group").append(t);
                    //清空全局的parentId
                    parentId=null;
                }
            }

        })
    })

/*
 ************************************评论回复************************************
 */
//给回复按钮绑定点击事件
$(".replay").click(function () {
    //需要评论对应的评论人姓名 还需要评论的主键值
    //获取用户名
    let commentUserName=$(this).attr("username");
    //获取主键值 直接修改全局的变量名
     parentId=$(this).attr("comment_id");
    //将信息塞给评论框
    $("#id_comment").val("@"+commentUserName+"\n").focus();
})
</script>
{% endblock js %}
```

# views.py

```
def article_detail(request,username,article_id):
    user_obj=models.UserInfo.objects.get(username=username)
    blog=user_obj.blog
    #先获取文章对象

article_obj=models.Article.objects.get(pk=article_id,blog__userinfo__username=username)
    if not article_obj:
        return render(request,"errors.html")
    #获取当前文章所有的评论内容
    comment_list=models.Comment.objects.filter(article=article_obj)
```

```
    return render(request,"article_detail.html",locals())

#点赞点踩
def up_or_down(request):
    '''
        1、校验当前用户是否登录
        2、判断当前文章是否是当前用户自己写的（用户不可以给自己文章点赞点踩）
        3、判断当前用户是否已经给当前文章点过了。只能点一次。
        4、操纵数据库
    '''
    if request.method=="POST":
        back_dic={
            "code":1000,
            "msg":"",
        }
        #1、判断当前用户是否登录
        if request.user.is_authenticated:     #用户已经登录
            article_id=request.POST.get("article_id") #文章主键值
            isUp=request.POST.get("isUp")   #返回值是一个字符串，值是"true"或者"false"
            isUp=json.loads(isUp)           #将普通字符串，转换为python格式的数据类型，相当于
去掉双引号操作
            #2、判断当前文章是否是当前用户自己写的。
            article_obj=models.Article.objects.get(pk=article_id)
            if not article_obj.blog.userinfo==request.user:
                #3、校验当前用户是否已经点了

is_click=models.UpAndDown.objects.filter(user=request.user,article=article_obj)
                if not is_click:
                    #4、操作数据库，纪录数据 要同步操作普通字段
                    # 判断当前用户是点了赞还是踩 从而决定给哪个字段加一
                    if isUp:
                        #给点赞数加一

models.Article.objects.filter(pk=article_id).update(up_num=F("up_num")+1)
                        back_dic["msg"]="点赞成功"
                    else:
                        #给点踩数加一

models.Article.objects.filter(pk=article_id).update(down_num=F("down_num")+1)
                        back_dic["msg"]="点踩成功"
                    # 操作点赞点踩表

models.UpAndDown.objects.create(user=request.user,article=article_obj,is_up=isUp)
                else:
                    back_dic["code"]=2000
                    back_dic["msg"]="你已经点过了，不能在点了"
            else:
```

```
                back_dic["code"]=3000
                back_dic["msg"]="不能给自己点赞"
        else:
            back_dic["code"]=4000
            back_dic["msg"]="<p><a href='/login/'>请先登录</a></p>"
        print("*********执行完毕*************")
        return JsonResponse(back_dic)

from django.db  import transaction
def comment(request):
    #自己可以评论自己的文章
    if request.method=="POST":
        back_dic={"code":1000,"msg":""}
        if request.user.is_authenticated:
            article_id=request.POST.get("article_id")
            content=request.POST.get("content")
            parent_id=request.POST.get("parent_id")
            #直接操作评论表存储数据  两张表
            with transaction.atomic():

models.Article.objects.filter(pk=article_id).update(comment_num=F("comment_num")+1)

models.Comment.objects.create(user=request.user,article_id=article_id,content=content,par
ent_id=parent_id)
                back_dic["msg"]="评论成功"
        else:
            back_dic["code"]=2000
            back_dic["msg"]="未登录"
        return JsonResponse(back_dic)
```