我的个人网站由来，以及部署时遇到的问题！

作者：wanglei0622
原文链接：https://ld246.com/article/1461296614373
来源网站：链滴

刚开始工作的时候，我的学习工作笔记是记在本子上的，然后移到有道云笔记，大概在第二年的候转移到CSDN博客，在那上面开始我的技术博客生涯。
一些知识点太久不用就会忘记，在博客上写下来，也是一个很不错的知识整理，还能给别人提供助。

工作以来就一直希望能有自己的网站，刚开始是因为技术水平不够，不知道如何下手，后来水平了，我对搭建自己的个人网站又没有之前那么的有热情了
自己写一个吧，还是要花点时间，其实我还是太懒了，这事就一直这么耽搁下来了。

直到前两天，不知怎么就看到了丁亮的博客，被他首页<span style="text-decoration: underline; color: #ff0000;" GitHub 上 Star 数最多的 Java 博客 Solo<pan> 给吸引了，从github上把solo的代码拉下来，本地部署一，
这不正是我需要的功能嘛，完全覆盖我的需求，花一天时间测试以后，我搭建自己博客热情又燃烧起来了，说干就干，域名早就买好放在那半年了，先借用公司的
服务器，部署上线。

谁想部署到服务器后测试出问题了，我采用的是nginx+tomcat的集群部署方案，部署服务器后，测试发现有功能的请求链接403了，删除功能、编辑功能都不能用，
想看日志，发现solo的日志只配置了在控制台输出，太不科学了，算了后面再给他加上log文件输出,先从tomcat的日志里看，发现程序并没有报错，那就只能看代码了，
找到删除的请求目标执行方法，代码如下

<pre class="brush: java">
/**
 * Removes an article by the specified request.
 *
 * <p>
 * Renders the response with a json object, for example,
 * <pre>
 * {
 *   "sc": boolean,
 *   "msg": ""
 * }
 * </pre>
 * </p>
 *
 * @param context the specified http request context
 * @param request the specified http servlet request
 * @param response the specified http servlet response
 * @param articleId the specified article id
 * @throws Exception exception
 */
@RequestProcessing(value = "/console/article/{articleId}", method = HTTPRequestMethod
DELETE)
public void removeArticle(final HTTPRequestContext context, final HttpServletRequest requ
st, final HttpServletResponse response,
        final String articleId) throws Exception {
    if (!userQueryService.isLoggedIn(request, response)) {
        response.sendError(HttpServletResponse.SC_FORBIDDEN);
        return;
    }
</pre>

```
    final JSONRenderer renderer = new JSONRenderer();

    context.setRenderer(renderer);

    final JSONObject ret = new JSONObject();

    renderer.setJSONObject(ret);

    try {
        if (!articleQueryService.canAccessArticle(articleId, request)) {
            ret.put(Keys.STATUS_CODE, false);
            ret.put(Keys.MSG, langPropsService.get("forbiddenLabel"));

            return;
        }

        articleMgmtService.removeArticle(articleId);

        ret.put(Keys.STATUS_CODE, true);
        ret.put(Keys.MSG, langPropsService.get("removeSuccLabel"));
    } catch (final Exception e) {
        LOGGER.log(Level.ERROR, e.getMessage(), e);

        final JSONObject jsonObject = new JSONObject();

        renderer.setJSONObject(jsonObject);
        jsonObject.put(Keys.STATUS_CODE, false);
        jsonObject.put(Keys.MSG, langPropsService.get("removeFailLabel"));
    }
}
```

</pre>
在方法体开头的时候（userQueryService.isLoggedIn(request, response)）了登录验证，这里再吐下槽，登录验证不应该提取出来么，在每个方法中都写一遍我也是醉了，
这里最有可能返回403，果断加log，看是否是因为session丢失导致的通不过登录验证，最后发请求根本没有进入到这个方法里面来，而是在这之前就被拦截了，但是solo的权限过滤器
并没有拦截删除请求，那就只有org.b3log.latke.servlet.DispatcherServlet这个solo使用的开源架latke有可能了，是这是框架的控制器，不太可能做这事啊，因为框架本地测试
是没有问题的，那我只能怀疑是nginx+tomcat的问题了，于是我绕过nginx，直接访问tomcat端口，测试居然通过了，没有再报403，那就是nginx的问题了，仔细一看nginx的配置
并没有发现什么问题，之前其他的项目都是这么配置的啊，难道是latke框架不支持？觉得不应该啊，只能继续用排除法了，将nginx最有可能的请过滤注释掉，如下代码
<pre class="brush: bash">upstream wanglei_xxxxx {
    server 127.0.0.1:xxxxx;
}
server {
    listen   80 ;
    server_name wanglei0622.cn www.wanglei0622.cn;
    #注释掉下面的请求类型过滤
      #if ($request_method !~* GET|HEAD|POST) {
    #   return 403;
    #}


```
```

```
    rewrite ^(.*)\;jsessionid=(.*)$  $1    break;
    location ~ ^/(WEB-INF)/ {
        deny all;
    }
    location ~ \.(gif|jpg|jpeg|png|ico|rar|css|js|zip|txt|flv|swf|doc|ppt|xls|pdf)$ {
         root /data/xxxxx;
         access_log off;
         expires 24h;
    }
    location / {
        proxy_pass http://wanglei_xxxxx;
        proxy_redirect          off;
        proxy_set_header        Host $host;
        proxy_set_header        X-Real-IP $remote_addr;
        proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
        client_max_body_size    10m;
        client_body_buffer_size 128k;
        proxy_connect_timeout   300;
        proxy_send_timeout      300;
        proxy_read_timeout      300;
        proxy_buffer_size       4k;
        proxy_buffers           4 32k;
        proxy_busy_buffers_size 64k;
        proxy_temp_file_write_size 64k;
    }
}<br />然后一看页面删除操作的JS代码，如下</pre>
<pre class="brush: js;light: true">/**
     * @description 删除文章
     * @param {String} id 文章 id
     * @param {String} fromId 文章来自草稿夹(draft)/文件夹(article)
     * @param {String} title 文章标题
     */
    del: function (id, fromId, title) {
        var isDelete = confirm(Label.confirmRemoveLabel + Label.articleLabel + '"' + title + '"?');
        if (isDelete) {
            $("#loadMsg").text(Label.loadingLabel);
            $("#tipMsg").text("");


        $.ajax({
            url: latkeConfig.servePath + "/console/article/" + id,
            type: "DELETE",
            cache: false,
            success: function (result, textStatus) {
                $("#tipMsg").text(result.msg);
                if (!result.sc) {
                    $("#loadMsg").text("");
                    return;
                }

                admin[fromId + "List"].getList(1);
            }
        });
    }
```

}

请求类型居然是 type: "DELETE" ，我也是醉了,哪位前端大神写的，不走寻常路啊！是够坑的，幸亏我找BUG经验丰富，二分法快速定位问题位置，排除法确定问题。太有成就感了！
接下来就等域名备案通过了！</pre>