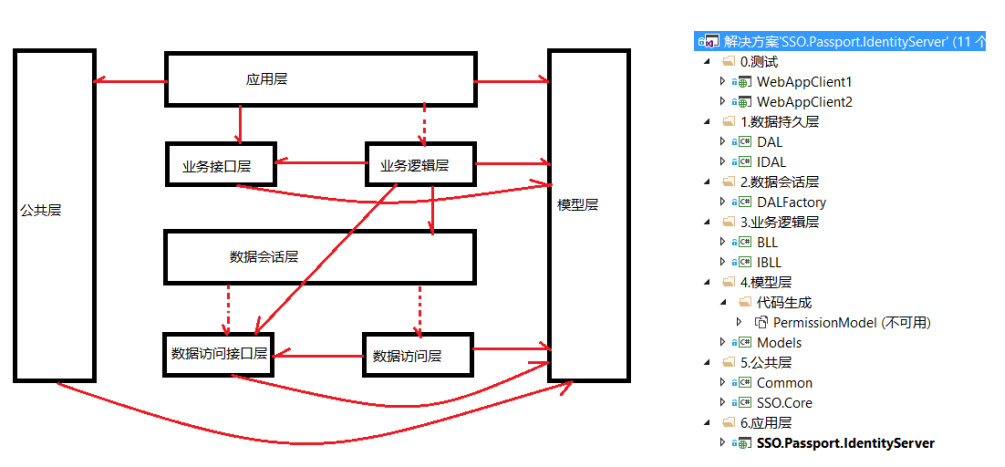
### 项目架构

核心架构和之前的开源项目SSO.Passport.IdentityServer保持一致



实线代表实际引用，虚线代表通过反射的方式引用；

应用层的真实引用只需要引用Common和Model以及IBLL即可，从而达到项目间的解耦，但由于生成和发布的原因，导致需要反射引用的类库不能生成到应用层目录下，所以应用层也全部引用了运行时需要的所有项目；

反射引用需要在应用层的web.config中配置DalNameSpace和DalPath以及BllPath，若未配置，则取程序相应的预设值；

**如何将应用层的数据请求传到数据层的？**

应用程序启动时将通过反射的方式加载业务逻辑层BLL的程序集，并创建相应的接口实例存到autofac容器中，当应用层发起数据请求时，将通过IBLL的接口实例进行处理，依靠BLL调用数据会话层DALFactory来反射创建数据访问层DAL对应的接口实例IDAL，IDAL依靠DAL进行真实的数据库操作，最后将数据返回到应用层。

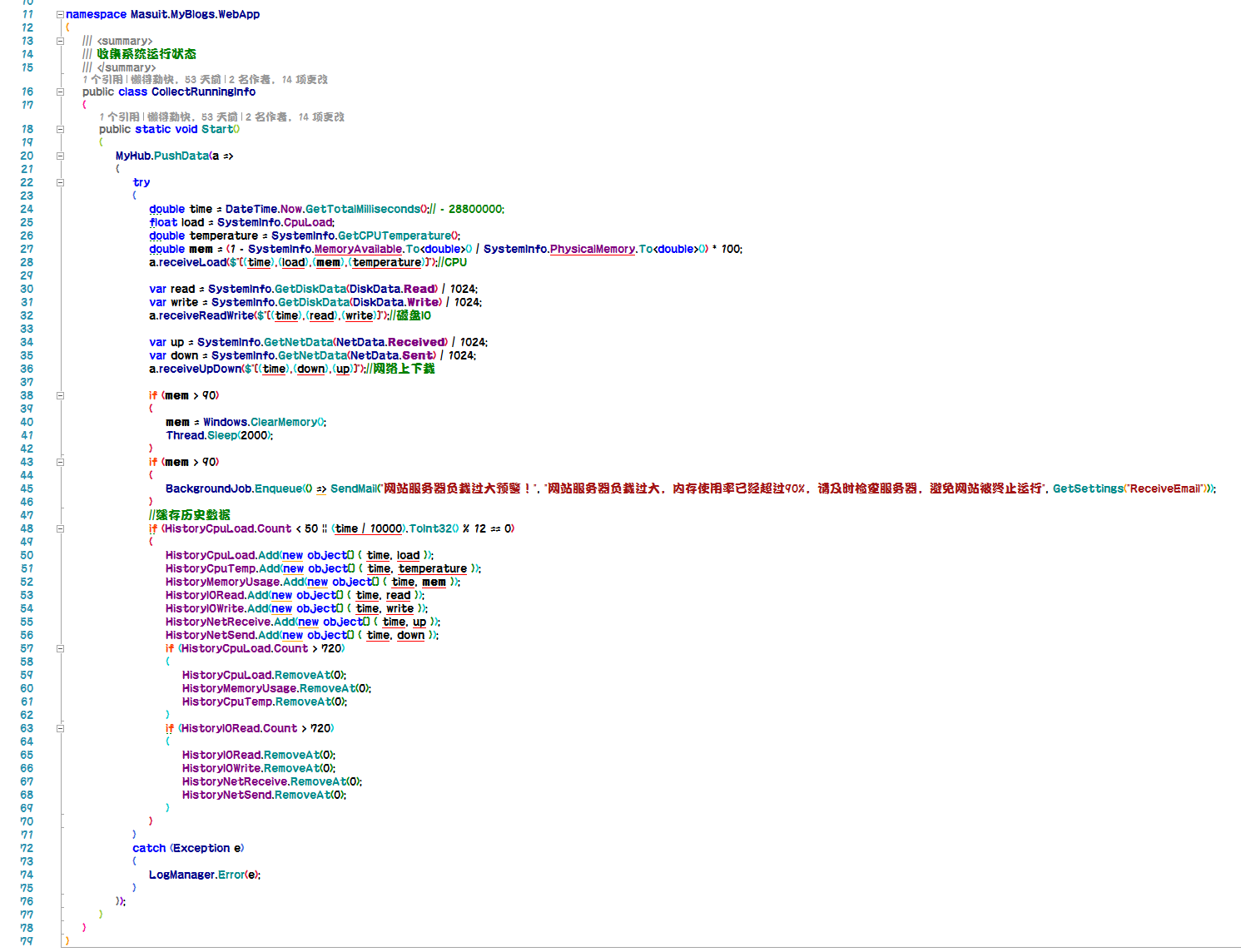
### 核心功能点技术实现

文章防XSS：通过[Masuit.Tools](https://github.com/ldqk/Masuit.Tools)封装的防XSS方法实现富文本html的过滤

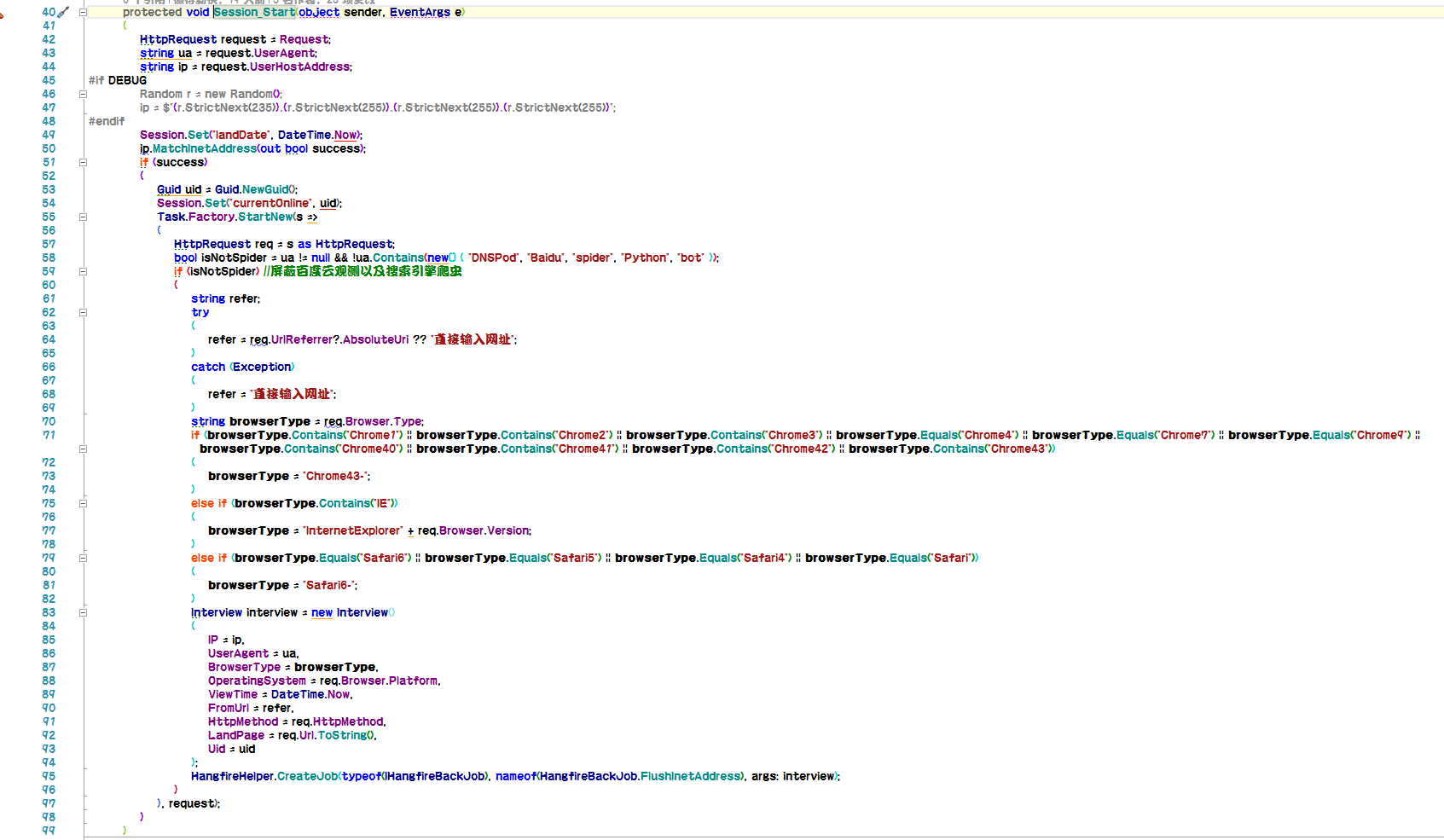
post.Content.HtmlSantinizerStandard()

文章图片上传：将上传到服务器的图片转发到新浪图床，如果上传失败，降级上传到sm.ms等其他图床。毕竟是免费的，可能不是长久之计，有条件的童鞋可以去申请专用的OSS服务器。

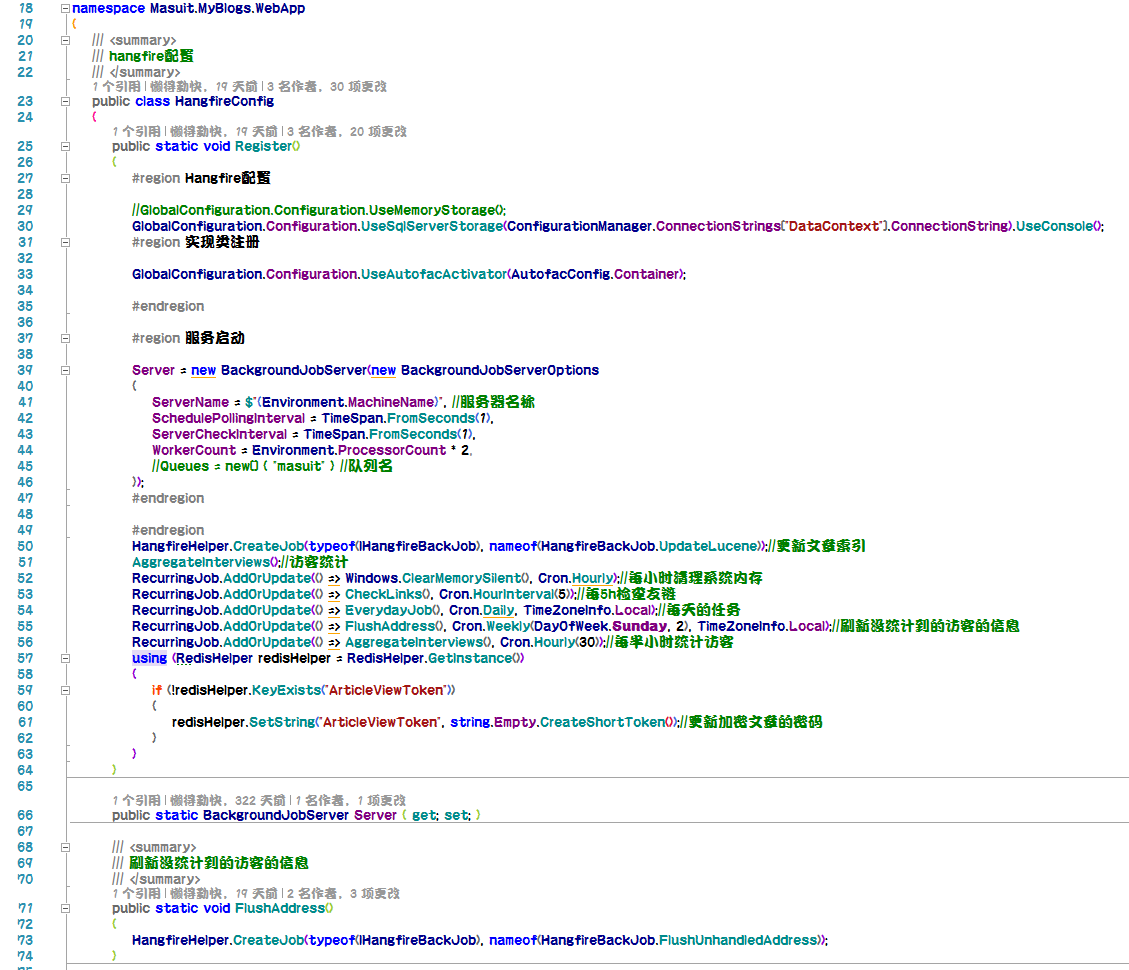
服务器监控：通过[Masuit.Tools](https://github.com/ldqk/Masuit.Tools)封装的SystemInfo类实现服务器硬件资源的实时监控，结合SignalR主动推送，图表使用highchart动态更新。



访客统计：在Session开始时接收客户端的IP和UA等信息，通过hangfire来调度，获取到IP地址的详细信息后写入到数据库，hangfire每小时统计历史数据，包括每日访客走势，独立IP，新增访客，以及每个访客浏览过哪些页面。



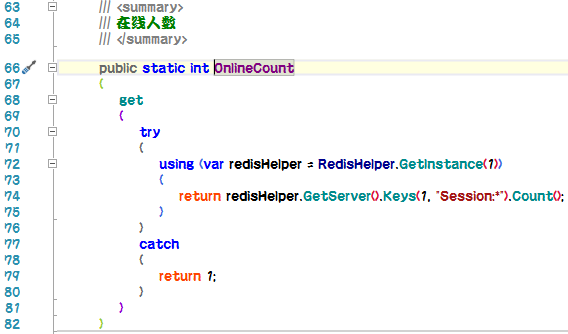
分布式任务调度：通过hangfire实现，结合hangfire控制面板，实现后台任务的监控。



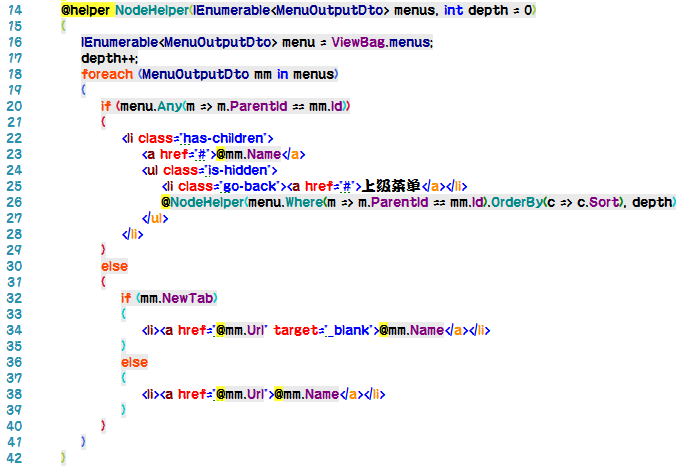
基于Redis的Session实现：程序产生一个SessionID后写到客户端的Cookie，这个SessionID对应了Redis的一个List，而List里面的每个键则是用户每个Session里的每个元素，滑动过期20min，Masuit.Tools封装实现。



在线人数统计：通过统计Redis里面SessionID的总数实现。在Session结束时会将Redis里面每个访客的记录同步到数据库保存。



无级菜单的加载，数据表依赖ParentId来确定上下级关系，前端通过递归创建html节点实现递归加载，评论和留言的递归加载也是如此。

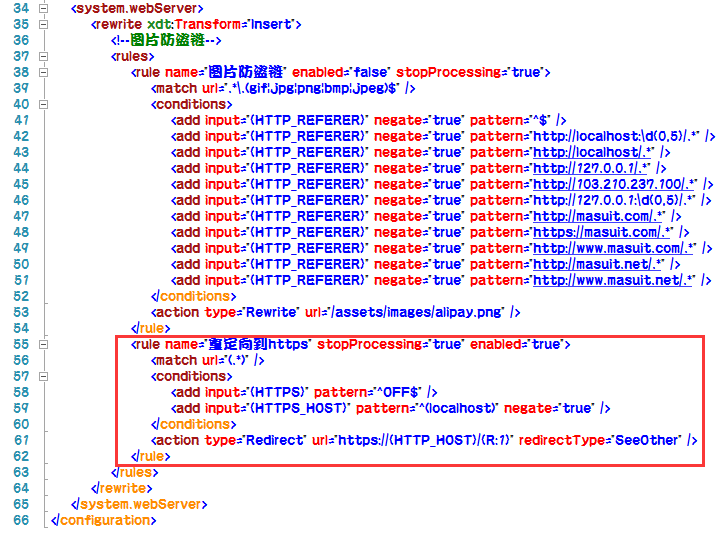


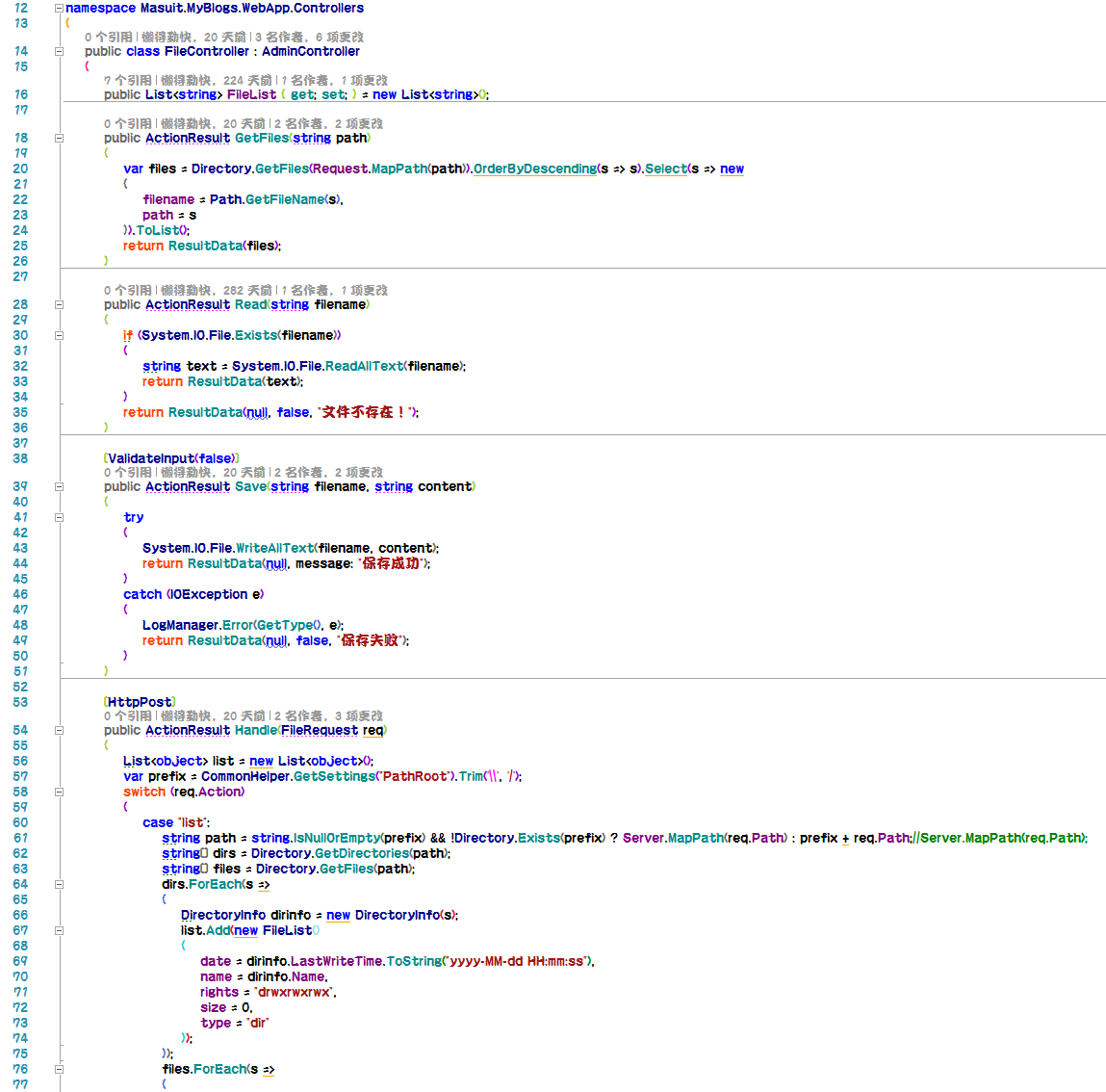
恶意请求拦截：如果同一个客户端每分钟内请求数达到阈值，则将该客户端拦截掉，或者同一客户端请求连续次数上限的错误页，也视为机器人，拦截掉。



服务器文件资源管理器：结合angular-filemanager实现服务器的资源管理器。

https：如果你没有申请https证书，发布时将web.release.config中的这一节去掉





### 项目部署

配置文件：

主要需要配置的是数据库连接字符串，BaiduAK；



同时，BaiduAK参与了数据库的加密，如果你没有BaiduAK，自行到百度地图开放平台申请，免费的。

如果Redis不在本机，需要在配置文件中的ConnectionStrings节下配置，name固定为RedisHosts，值的格式：127.0.0.1:6379,allowadmin=true，若未正确配置，将按默认值“127.0.0.1:6379,allowadmin=true,abortConnect=false”。

IIS：部署时必须将应用程序池的标识设置为LocalSystem，否则无法监控服务器硬件

