非法字符过滤---动态代理版

动态代理实现技术：
    JDK：要求目标对象和代理对象，实现同一个接口

步骤：
    1.创建目标对象（过滤器已提供）
    2.通过Proxy工具，生产代理类，创建代理对象
    3.放行代理对象

代码实现流程：

```
@WebFilter("/illegalServet")
public class IllegalFilter2 implements Filter {

    List<String> list = new ArrayList<>();

    // 加载非法词库
    public void init(FilterConfig config) throws ServletException {

        InputStream is = null;
        InputStreamReader inputStreamReader = null;
        BufferedReader bufferedReader = null;

        try {
            // 1.通过类加载器，读取文件，获取io流
            is =
IllegalFilter2.class.getClassLoader().getResourceAsStream("illegal_words.txt");
            // 2.转换为字符流
            inputStreamReader = new InputStreamReader(is, "utf-8");
            // 3.转换字符高效缓冲流
            bufferedReader = new BufferedReader(inputStreamReader);
            // 4.遍历输出到list集合
            String line = null;

            while ((line = bufferedReader.readLine()) != null) {
                list.add(line);
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // 5.释放资源
            try {
                bufferedReader.close();
                inputStreamReader.close();
                is.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    // 拦截敏感词汇
```

```
    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain
chain) throws ServletException, IOException {
        // 设置请求解码方式
        req.setCharacterEncoding("utf-8");
        // 设置响应编码方式
        resp.setContentType("text/html;charset=utf-8");

        // 1.创建目标对象（过滤器已提供   ServletRequest req）
        // 2.通过Proxy工具，生产代理类，创建代理对象
        /*
                类加载器：目标对象的加载器
                接口数组：目标对象的接口数组
                处理器接口：创建匿名内部类，实现增强业务逻辑
         */
        ServletRequest requestProxy= (ServletRequest)
Proxy.newProxyInstance(req.getClass().getClassLoader(),
req.getClass().getInterfaces(), new InvocationHandler() {
            /*
                proxy：生产的代理对象
                method：当前执行具体方法对象
                args：当前方法具体传递参数数组
             */
            @Override
            public Object invoke(Object proxy, Method method, Object[] args)
throws Throwable {

                // 目标对象实现具体的功能
                Object object = method.invoke(req, args);

                // 如果执行的是 getParameter方法 实现对内容增强 过滤
                if("getParameter".equals(method.getName())){
                    // 认为这个结果就是字符串
                    String msg = (String) object;
                    // 遍历词库
                    for (String illegal : list) {
                        if(msg.contains(illegal)){
                            msg=msg.replaceAll(illegal, "***");
                        }
                    }
                    // 返回增强后结果
                    return msg;
                }

                // 返回结果
                return object;
            }
        });
        // 3.放行代理对象
        chain.doFilter(requestProxy, resp);
    }

    public void destroy() {

    }

}
```