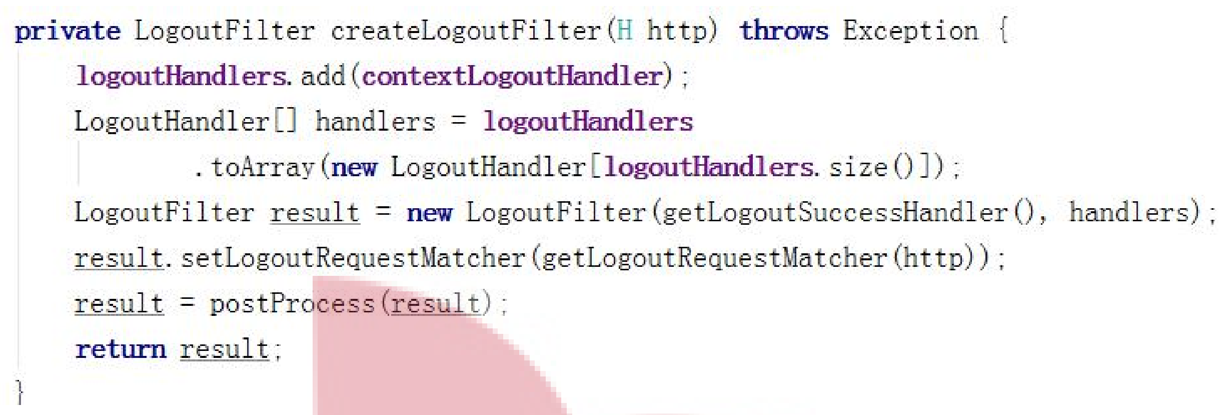
logout其他常用配置源码解读

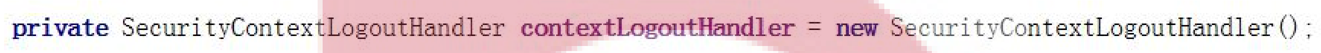
addLogoutHandler(LogoutHandler)

默认是 contextLogoutHandler



```java
private LogoutFilter createLogoutFilter(H http) throws Exception {
    logoutHandlers.add(contextLogoutHandler);
    LogoutHandler[] handlers = logoutHandlers
            .toArray(new LogoutHandler[logoutHandlers.size()]);
    LogoutFilter result = new LogoutFilter(getLogoutSuccessHandler(), handlers);
    result.setLogoutRequestMatcher(getLogoutRequestMatcher(http));
    result = postProcess(result);
    return result;
}
```

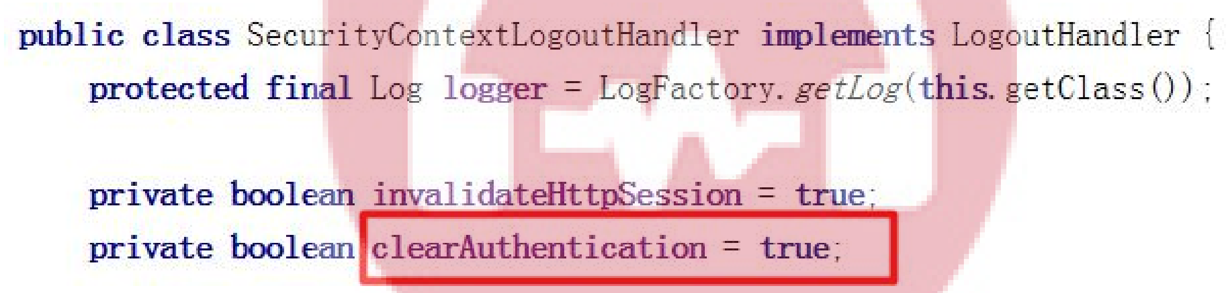
默认实例内容



```java
private SecurityContextLogoutHandler contextLogoutHandler = new SecurityContextLogoutHandler();
```

clearAuthentication(boolean)

是否清除认证状态，默认为 true

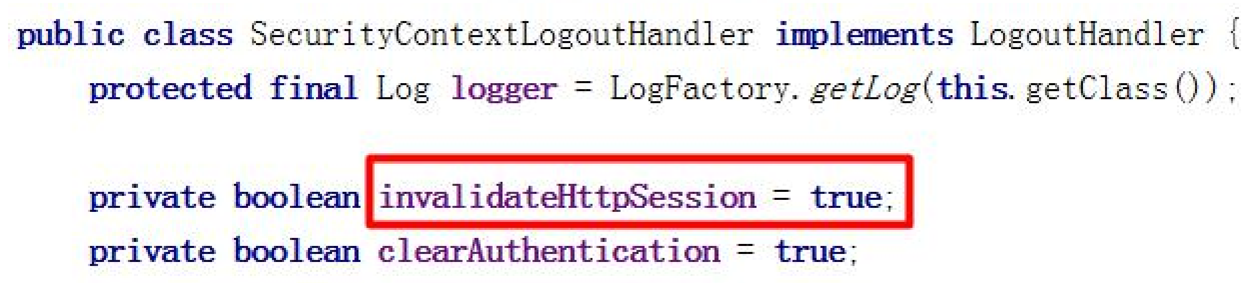


```java
public class SecurityContextLogoutHandler implements LogoutHandler {
    protected final Log logger = LogFactory.getLog(this.getClass());

    private boolean invalidateHttpSession = true;
    private boolean clearAuthentication = true;
```

invalidateHttpSession(boolean)

是否销毁 HttpSession 对象，默认为 true

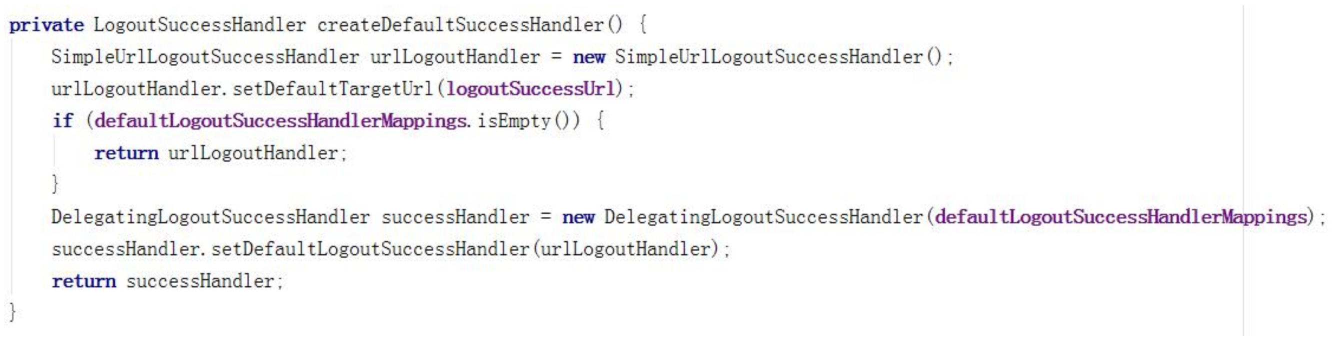


```java
public class SecurityContextLogoutHandler implements LogoutHandler {
    protected final Log logger = LogFactory.getLog(this.getClass());

    private boolean invalidateHttpSession = true;
    private boolean clearAuthentication = true;
```

logoutSuccessHandler(LogoutSuccessHandler)

退出成功处理器



```java
private LogoutSuccessHandler createDefaultSuccessHandler() {
    SimpleUrlLogoutSuccessHandler urlLogoutHandler = new SimpleUrlLogoutSuccessHandler();
    urlLogoutHandler.setDefaultTargetUrl(logoutSuccessUrl);
    if (defaultLogoutSuccessHandlerMappings.isEmpty()) {
        return urlLogoutHandler;
    }
    DelegatingLogoutSuccessHandler successHandler = new DelegatingLogoutSuccessHandler(defaultLogoutSuccessHandlerMappings);
    successHandler.setDefaultLogoutSuccessHandler(urlLogoutHandler);
    return successHandler;
}
```

也可以自己进行定义退出成功处理器。只要实现了 `LogoutSuccessHandler` 接口。与之前讲解的登录成功处理器和登录失败处理器极其类似。

SpringSecurity中的CSRF

从刚开始学习Spring Security时，在配置类中一直存在这样一行代码： `http.csrf().disable();` 如果没有这行代码导致用户无法被认证。这行代码的含义是：关闭 csrf 防护。

什么是CSRF

CSRF（Cross-site request forgery）跨站请求伪造，也被称为“OneClick Attack” 或者 Session Riding。通过伪造用户请求访问受信任站点的非法请求访问。

跨域：只要网络协议，ip 地址，端口中任何一个不相同就是跨域请求。

客户端与服务进行交互时，由于 http 协议本身是无状态协议，所以引入了cookie进行记录客户端身份。在cookie中会存放session id用来识别客户端身份的。在跨域的情况下，session id 可能被第三方恶意劫持，通过这个 session id 向服务端发起请求时，服务端会认为这个请求是合法的，可能发生很多意想不到的事情。