springboot整合shiro-实现自己的登出(十六)

原文地址，转载请注明出处： *https://blog.csdn.net/qq_34021712/article/details/84722252* *©王赛超*

前面所有的博客登出都是使用的 shiro 默认自带的登出,使用方式也很简单，不用我们去实现退出功能，只要去访问一个退出的 url （该 url 是可以不存在），由 LogoutFilter 拦截住，清除 session 。(如果没有什么特殊需求,我建议直接使用 shiro 的登出) 具体如下：

```
        // 配置访问权限 必须是LinkedHashMap，因为它必须保证有序
        // 过滤链定义，从上向下顺序执行，一般将 /**放在最为下边 --> : 这是一个坑，一不小心代码就不好使了
        LinkedHashMap<String, String> filterChainDefinitionMap = new LinkedHashMap<>();
        //配置不登录可以访问的资源，anon 表示资源都可以匿名访问
        //配置记住我或认证通过可以访问的地址
        filterChainDefinitionMap.put("/login", "anon");
        filterChainDefinitionMap.put("/", "anon");
        filterChainDefinitionMap.put("/css/**", "anon");
        filterChainDefinitionMap.put("/js/**", "anon");
        filterChainDefinitionMap.put("/img/**", "anon");
        filterChainDefinitionMap.put("/druid/**", "anon");
        //解锁用户专用 测试用的
        filterChainDefinitionMap.put("/unlockAccount","anon");
        filterChainDefinitionMap.put("/Captcha.jpg","anon");
        //logout是shiro提供的过滤器
        filterChainDefinitionMap.put("/logout", "logout");
        //此时访问/user/delete需要delete权限,在自定义Realm中为用户授权。
        //filterChainDefinitionMap.put("/user/delete", "perms[\"user:delete\"]");

        //其他资源都需要认证  authc 表示需要认证才能进行访问 user表示配置记住我或认证通过可以访问的地址
        //如果开启限制同一账号登录,改为 put("/**", "kickout,user");
```


只要拦截到访问 /logout 的请求,就会被走 logout 对应的 LogoutFilter ， 自动登出。

为什么要实现自定义登出?

shiro 的默认登出也会清理用户的 session 信息,并且也会清理掉 redis 中缓存的用户 身份认证 和 权限认证 的相关信息，但是为什么还要实现自定义登出呢?

但是有时候,我们还有自己的一些业务需要处理,比如说前面做的 限制用户的登录次数 和 并发登录的人数,使用 shiro 默认登出这些 key 是不会被删除的,假如我们想要删除这些 key 呢? 或者说 在用户登出时,要记录日志 当前用户在线时长,这个时候 我们就需要自定义登出。

登出的两种实现方式

第一种是 不配置 默认的logout,自己写一个Controller方法对外提供http接口,在该Controller方法中实现 登出的逻辑。

```
/**
 * 登出  这个方法没用到,用的是shiro默认的logout
 * @param session
 * @param model
 * @return
 */
@RequestMapping("/logout")
public String logout(HttpSession session, Model model) {
    Subject subject = SecurityUtils.getSubject();
    subject.logout();
    model.addAttribute("msg","安全退出！");
    return "login";
}
```

第二种是 继承LogoutFilter过滤器，并重写preHandle方法。

第一步：自定义实现LogoutFilter

```
package com.springboot.test.shiro.config.shiro;

import org.apache.shiro.SecurityUtils;
import org.apache.shiro.subject.PrincipalCollection;
import org.apache.shiro.subject.Subject;
import org.apache.shiro.web.filter.authc.LogoutFilter;
import org.apache.shiro.web.mgt.DefaultWebSecurityManager;

import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

/**
 * @author: wangsaichao
 * @date: 2018/11/27
 * @description: 自定义 LogoutFilter
 */
public class ShiroLogoutFilter extends LogoutFilter {

```

```
    /**
     * 自定义登出,登出之后,清理当前用户redis部分缓存信息
     * @param request
     * @param response
     * @return
     * @throws Exception
     */
    @Override
    protected boolean preHandle(ServletRequest request, ServletResponse response) throws Exception {

        //登出操作 清除缓存  subject.logout() 可以自动清理缓存信息, 这些代码是可以省略的  这里只是做个笔记 表示这种方式也可以清除
        Subject subject = getSubject(request,response);
        DefaultWebSecurityManager securityManager = (DefaultWebSecurityManager) SecurityUtils.getSecurityManager();
        ShiroRealm shiroRealm = (ShiroRealm) securityManager.getRealms().iterator().next();
        PrincipalCollection principals = subject.getPrincipals();
        shiroRealm.clearCache(principals);

        //登出
        subject.logout();

        //获取登出后重定向到的地址
        String redirectUrl = getRedirectUrl(request,response,subject);
        //重定向
        issueRedirect(request,response,redirectUrl);
        return false;
    }

}
```

第二步：ShiroConfig中配置 shiroLogoutFilter

```
/**
 * 配置LogoutFilter
 * @return
 */
public ShiroLogoutFilter shiroLogoutFilter(){
    ShiroLogoutFilter shiroLogoutFilter = new ShiroLogoutFilter();
    //配置登出后重定向的地址，等出后配置跳转到登录接口
    shiroLogoutFilter.setRedirectUrl("/login");
    return shiroLogoutFilter;
}
```

第三步：将配置的shiroLogout给ShiroFilterFactoryBean

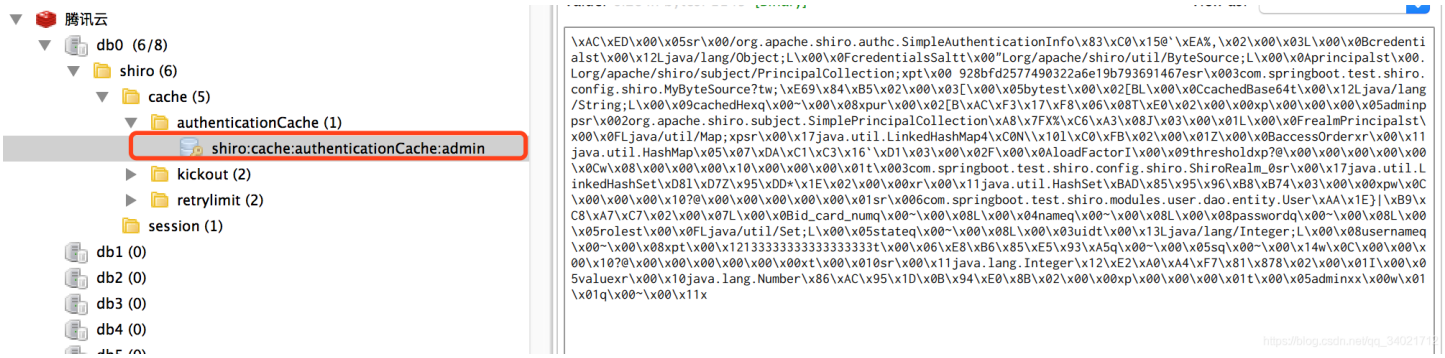
```
    /**
     * ShiroFilterFactoryBean 处理拦截资源文件问题。
     * 注意：初始化ShiroFilterFactoryBean的时候需要注入：SecurityManager
     * Web应用中,Shiro可控制的Web请求必须经过Shiro主过滤器的拦截
     * @param securityManager
     * @return
     */
    @Bean(name = "shirFilter")
    public ShiroFilterFactoryBean shiroFilter(@Qualifier("securityManager") SecurityManager securityManager) {

        ShiroFilterFactoryBean shiroFilterFactoryBean = new ShiroFilterFactoryBean();

        //必须设置 SecurityManager,Shiro的核心安全接口
        shiroFilterFactoryBean.setSecurityManager(securityManager);
        //这里的/login是后台的接口名,非页面，如果不设置默认会自动寻找Web工程根目录下的"/login.jsp"页面
        shiroFilterFactoryBean.setLoginUrl("/");
        //这里的/index是后台的接口名,非页面,登录成功后要跳转的链接
        shiroFilterFactoryBean.setSuccessUrl("/index");
        //未授权界面,该配置无效，并不会进行页面跳转
        shiroFilterFactoryBean.setUnauthorizedUrl("/unauthorized");

        LinkedHashMap<String, Filter> filtersMap = new LinkedHashMap<>();
        //限制同一帐号同时在线的个数
        filtersMap.put("kickout", kickoutSessionControlFilter());

        //配置自定义登出 覆盖 logout 之前默认的LogoutFilter
        filtersMap.put("logout", shiroLogoutFilter());


        shiroFilterFactoryBean.setFilters(filtersMap);
        // 配置访问权限 必须是LinkedHashMap，因为它必须保证有序
        // 过滤链定义，从上向下顺序执行，一般将 /**放在最为下边 --> : 这是一个坑，一不小心代码就不好使了
        LinkedHashMap<String, String> filterChainDefinitionMap = new LinkedHashMap<>();
        //配置不登录可以访问的资源，anon 表示资源都可以匿名访问
```

```
        //配置记住我或认证通过可以访问的地址
        filterChainDefinitionMap.put("/login", "anon");
        filterChainDefinitionMap.put("/", "anon");
        filterChainDefinitionMap.put("/css/**", "anon");
        filterChainDefinitionMap.put("/js/**", "anon");
        filterChainDefinitionMap.put("/img/**", "anon");
        filterChainDefinitionMap.put("/druid/**", "anon");
        //解锁用户专用 测试用的
        filterChainDefinitionMap.put("/unlockAccount","anon");
        filterChainDefinitionMap.put("/Captcha.jpg","anon");
        //logout是shiro提供的过滤器,这是走自定义的 shiroLogoutFilter 上面有配置
        filterChainDefinitionMap.put("/logout", "logout");
        //此时访问/user/delete需要delete权限,在自定义Realm中为用户授权。
        //filterChainDefinitionMap.put("/user/delete", "perms[\"user:delete\"]");

        //其他资源都需要认证  authc 表示需要认证才能进行访问 user表示配置记住我或认证通过可以访问的地址
        //如果开启限制同一账号登录,改为 .put("/**", "kickout,user");
        filterChainDefinitionMap.put("/**", "kickout,user");

        shiroFilterFactoryBean.setFilterChainDefinitionMap(filterChainDefinitionMap);

        return shiroFilterFactoryBean;
    }
```

上面的代码中 `filtersMap.put("logout", shiroLogoutFilter());` 表示 `logout` 走自定义的 `filter` 不再走默认的 `LogoutFilter` 。

可能出现的问题

`redis` 中的身份认证缓存，无法清除
无论是使用 `subject.logout();` 还是使用 `shiroRealm` 中的自定义方法 `shiroRealm.clearCache(principals);` 都只能清除 权限的缓存信息 却无法清除 身份认证的缓存信息 ，如下：



为什么会造成这个问题？ 我会在下一章中讲解为什么会出现这个问题。

具体参考下一篇博客：springboot整合shiro-关于登出时,redis中缓存没有清理干净的问题(十七)