springboot整合shiro-关于登出时,redis中缓存没有清理干净的问题(十七)

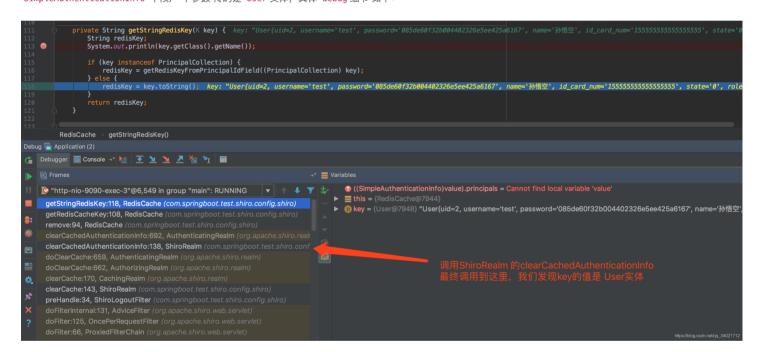
原文地址,转载请注明出处: https://blog.csdn.net/qq 34021712/article/details/84722724 ⑥王赛超

如果是跟着我的 shiro 系列博客敲下来的,其实还有一个 bug ,这是一个网友遇到的,他在登出的时候,发现 redis 中当前用户身份认证缓存没有清理掉,之前在 springboot整合shiro-ehcache缓存(五) 中测试添加权限之后,清理的是所有用户的缓存,所以没有发现这个问题。

还记得上一篇博客: springboot整合shiro-实现自己登出(十五), 我们在登出方法中,清理了当前用户的 身份认证 和 权限认证 的 缓存信息,最后发现有一个 key 没有清理 掉。如下图:



为什么 该 key 没有清除掉呢? 经过 debug 发现,在清理 身份认证 缓存的时候,调用了 ShiroRealm 的 clearCachedAuthenticationInfo 最终调用到 RedisCache 的 remove 方法,但是传过来 key 却是 User 实体,为什么会是 User 实体,就是因为在 ShiroRealm 的 doGetAuthenticationInfo 方法返回值 SimpleAuthenticationInfo 中,第一个参数 传的是 User 实体,具体 debug 细节 如下:



而在删除 用户 权限缓存时,却没有这个问题,删除缓存时,传入的 key 为 SimplePrincipalCollection 最终调用 getRedisKeyFromPrincipalIdField 根据你在 ShiroConfig 中 配置 RedisCacheManager 指定的那个字段作为缓存的前缀,根据反射获取该字段的值 并返回,具体的配置信息,可以参考之前博客的源码:

```
vate String getStringRedisKey(K key) { key: "
String redisKey; redisKey: "test"
System.out.println(key.getClass().getName());
                                    if (key instanceof PrincipalCollection) {
    redisKey = getRedisKeyFromPrincipalIdField((PrincipalCollection) key);
                                               redisKev:
                                                                                        redisKey: "test"
                         private String getRedisKeyFromPrincipalIdField(PrincipalCollection key) {
   String redisKey;
              RedisCache > getStringRedisKey()
 Application (2)
Debugger 

Console → 

Consol
                                                                                                                                                                                                                         →" U Variables
                                                                                                                                                                                                                                                   ((SimpleAuthenticationInfo)value).principals = Cannot find local variable 'value
🌠 "http-nio-9090-exec-3"@6,549 in group "main": RUNNING 🔻 🛊 🍸
                                                                                                                                                                                                                                                 getStringRedisKey:120, RedisCache (com.springboot.test.shiro.config.shiro)
   getRedisCacheKey:108, RedisCache (c
      clearCachedAuthorizationInfo:129, ShiroRealm (
                                                                                                                              ache.shiro.realm)
 clearCache:170, CachingRealm (org.apache.shiro.realm)
clearCache:1743, ShiroRealm (org.apache.shiro.realm)
preHandle:34, ShiroRealm (com.springboot.test.shiro.config.shiro)
doFilterInternal:131, AdviceFilter (org.apache.shiro.web.servlet)
doFilter:125, OncePerRequestFilter (org.apache.shiro.web.servlet)
```

解决方案

第一种, 判断key为User实体,强转并获取用户名

```
private String getRedisCacheKey(K key) {
    if (key == null) {
        return null;
    }
    return this.keyPrefix + getStringRedisKey(key);
}

private String getStringRedisKey(K key) {
    String redisKey;
    System.out.println(key.getClass().getName());

    if (key instanceof PrincipalCollection) {
        redisKey = getRedisKeyEromPrincipalIdEield((PrincipalCollection) key);
    } else if(key instanceof User){
        redisKey = ((User)key).getUsername();
    } else {
        redisKey = key.toString();
    }
    return redisKey;
}
```

第二种,将ShiroRealm 的 doGetAuthenticationInfo 方法返回值 SimpleAuthenticationInfo 中,第一个参数,传username,不要传User实体

第一步: 修改ShiroRealm

修改 shiroRealm 中的 doGetAuthenticationInfo 验证用户身份的最后一句 返回值

return new SimpleAuthenticationInfo(user.getUsername(), user.getPassword(),new MyByteSource(user.getUsername()),getName());

```
protected \ Authentication Info \ do Get Authentication Info \ (Authentication Token) \ throws \ Authentication Exception \ \{ (Authentication Token) \ throws \ Authentication Exception \ \{ (Authentication Token) \ throws \ Authentication Exception \ \{ (Authentication Token) \ throws \ Authentication Exception \ \{ (Authentication Token) \ throws \ Authentication Exception \ \{ (Authentication Token) \ throws \ Authentication Exception \ \{ (Authentication Token) \ throws \ Authentication Exception \ \{ (Authentication Token) \ throws \ Authentication Exception \ \{ (Authentication Token) \ throws \ Authentication Exception \ \{ (Authentication Token) \ throws \ Authentication Exception \ \{ (Authentication Token) \ throws \ Authentication Exception \ \{ (Authentication Token) \ throws \ Authentication Exception \ \{ (Authentication Token) \ throws \ Authentication Exception \ \{ (Authentication Token) \ throws \ Authentication Exception \ \{ (Authentication Token) \ throws \ Authentication Exception \ \{ (Authentication Exception Exception \ throws \ Authentication \ thro
    2
    3
                               //获取用户名密码 第一种方式
                               //String username = (String) authenticationToken.getPrincipal();
    4
     5
                              //String password = new String((char[]) authenticationToken.getCredentials());
    6
     7
                              //获取用户名 密码 第二种方式
    8
                               UsernamePasswordToken usernamePasswordToken = (UsernamePasswordToken) authenticationToken;
                               String username = usernamePasswordToken.getUsername();
    9
10
                               String password = new String(usernamePasswordToken.getPassword());
11
 12
                               //从数据库查询用户信息
                               User user = this.userMapper.findByUserName(username);
13
14
                               //可以在这里直接对用户名校验,或者调用 CredentialsMatcher 校验
15
16
                               if (user == null) {
```

```
throw new UnknownAccountException("用户名或密码错误!"):
17
18
19
       //这里将 密码对比 注销掉。否则 无法锁定 要将密码对比 交给 密码比较器
20
       //if (!password.equals(user.getPassword())) {
       //
             throw new IncorrectCredentialsException("用户名或密码错误!");
21
22
       //}
23
       if ("1".equals(user.getState())) {
           throw new LockedAccountException("账号已被锁定,请联系管理员!");
24
25
26
27
       SimpleAuthenticationInfo info = new SimpleAuthenticationInfo(user.getUsername(), user.getPassword(),new MyByteSource(user.getUsern
28
       return info:
29
```

并且修改shiroRealm中的 doGetAuthorizationInfo 方法,SecurityUtils.getSubject().getPrincipal()之前 返回的是User实体,现在 就是我们上一步中 放进去的用户名。并且这里需要再单独根据 username去数据库查询 User

```
protected \ Authorization Info \ do Get Authorization Info \ (Principal Collection \ principal Collection) \ \{ protected \ Authorization \ Principal Collection \ principal Collectio
   1
   2
                           System.out.println("查询权限方法调用了!!");
   3
   4
                           //这里获取到的就是 上面方法放进去的username了
   5
                           String username = (String)SecurityUtils.getSubject().getPrincipal();
    6
                           //需要单独根据 username 从数据库查询用户信息
   7
   8
                           User user = this.userMapper.findBvUserName(username):
   Q
10
                          //获取用户鱼色
11
                          Set<Role> roles =this.roleMapper.findRolesByUserId(user.getUid());
12
                           //添加角色
13
                           SimpleAuthorizationInfo authorizationInfo = new SimpleAuthorizationInfo();
                           for (Role role : roles) {
14
15
                                       authorizationInfo.addRole(role.getRole());
16
17
                          //获取用户权限
18
19
                           Set<Permission> permissions = this.permissionMapper.findPermissionsByRoleId(roles);
20
21
                           for (Permission permission:permissions) {
                                       authorizationInfo.addStringPermission(permission.getPermission()):
22
23
24
25
                           return authorizationInfo;
             }
26
```

第二步: 修改项目中 其他地方 使用 (User)SecurityUtils.getSubject().getPrincipal();的 代码

因为 我们在第一部中 已经将 Principal 从 User 实体 改为了 username 所以这些相应的都需要修改,这里要看你们代码中都是在那里使用的了。

第三步: 修改之前 index.html页面中的34行 <shiro:principal property="username"/>

因为 principal 为 用户名,已经不是实体了,这里再指定 property 就会报如下 异常:

```
2 | 2018/11/30 15:13:28.861 org.thymeleaf.TemplateEngine [] ERROR [THYMELEAF][http-nio-9090-exec-5] Exception processing template "index": 2 | 2018/11/30 15:13:28.862 o.s.web.servlet.DispatcherServlet [] DEBUG Error rendering view [org.thymeleaf.spring4.view.ThymeleafView@7578 ] org.thymeleaf.exceptions.TemplateProcessingException: Error during execution of processor 'at.pollux.thymeleaf.shiro.processor.element
```

```
2018/11/30 15:13:28.861 org.thymeleaf.TemplateEngine [] ERROR [THYMELEAF] [http-nio-9090-exec-5] Exception processing template "index": Error during execution of processor 'at.pollux .thymeleaf.shiro.processor.element.HasPermissionElementProcessor' (index:15)
2018/11/30 15:13:28.862 o.s.web.servlet.DispatcherServlet [] DEBUG Error rendering view [org.thymeleaf.spring4.view.ThymeleafView@7578de10] in DispatcherServlet with name 'dispatcherServlet' org.thymeleaf.exceptions.TemplateProcessingException: Error during execution of processor 'at.pollux.thymeleaf.shiro.processor.element.HasPermissionElementProcessor' (index:15) at org.thymeleaf.dom.Node.applyNextProcessor(Node.java:1017) at org.thymeleaf.dom.Node.applyNextProcessor(Node.java:1017) at org.thymeleaf.dom.Node.processNode(Node.java:1017) at org.thymeleaf.dom.Node.orgorcessNode(Node.java:1017) at org.thymeleaf.dom.Node.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoule.shoul
```

具体如下图,只需要使用 <shiro:principal></shiro:principal> 就可以了。

```
<!-- 用户没有身份验证时显示相应信息,即游客访问信息 -->
<shiro:guest为客显示的信息</pre>
<!-- 用户已经身份验证/记住我登录后显示相应的信息 -->
<shiro:user>用户已经身分验证/这住我登录后显示相应的信息 -->
<shiro:user>用户已经身分验证通过,即Subject.login登录成功,不是记住我登录的 -->
<shiro:authenticated>不是记住我登录</shiro:authenticated>br/>
<!-- 用户已经身份验证通过,即Subject.login登录成功,不是记住我登录的 -->
<shiro:authenticated>不是记住我登录</shiro:authenticated>br/>
<!-- 显示用户身份信息,通常为登录帐号信息,默认调用Subject.getPrincipal()获取,即Primary Principal -->
<shiro:principal></shiro:principal></shiro:notipal></shiro:notipal></shiro:notipal></shiro:notAuthenticated>->
<!-- 相当于Subject.getPrincipals(),oneByType(String.class) -->
<shiro:orincipal tvbe="iava.lang.String"/>>br/>
<!-- 相当于(String)Subject.getPrincipals()).getUsername() 因为principal 已经改为用户名,所以这一行注释律 -->
<!-- 如果当即Subject.getPrincipals()).getUsername() 因为principal 已经改为用户名,所以这一行注释 -->
<shiro:hasAnyRoles name="admin,vip">-> (shiro:hasAnyRoles name="haeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,nee"/aeai,n
```

第四步:修改RedisCache 类中的 getStringRedisKey 方法

直接返回 key-toString 下面的 getRedisKeyFromPrincipalIdField 方法 也可以直接删除了,因为不再使用它了,原本 它存在的意义 就是为了解决 principal 放的是 User 实体。

```
private String getStringRedisKey(K key) {
      String redisKey;
System.out.println(key.getClass().getName());
      redisKey = getRedisKeyFromPrincipalIdField((PrincipalCollection) key);
} else if(key instanceof User){
  redisKey = ((User)key).getUsername();
                                                                             型的代码可以直接使用 key.toString了,
    return key.toString();
private String getRedisKeyFromPrincipalIdField(PrincipalCollection key) {
    String redisKey
    Object principalObject = key.getPrimaryPrincipal();
    Method pincipalIdGetter =
    Method[] methods = principalObject.getClass().getDeclaredMethods();
for (Method m:methods) {
         if (RedisCacheManager.DEFAULT_PRINCIPAL_ID_FIELD_NAME.equals(this.principalIdFieldName)
                 °) 33
                           AuthCacheKey".equals(m.getName()) || "getId".equals(m.getName()))) {
             pincipalIdGetter = m;
         if (m.getName().equals("get" + this.principalIdFieldName.substring(0, 1).toUpperCase() + this.principalIdFieldName.substring(1))) {
             pincipalIdGetter = m;
       (pincipalIdGetter == null) {
             w new PrincipalInstanceException(principalObject.getClass(), this.principalIdFieldName);
```

第五步: 启动测试

将 redis 清空, 并将浏览器缓存清除,启动项目测试,问题已解决,至于另外两个key都是我们自定义的功能,如果想要删除的话,直接删除redis的key就行了 都是使用username 拼接的key。

管理员清理其他用户的缓存

还有这样一种情况,有 test 和 admin 两名用户,admin 是管理员,在给 test 用户分配新的权限之后,需要清除该用户的 权限缓存信息,这里有一种笨方法,如下,添加一个 删除用户缓存的方法,只有 userInfo:clearCache 权限才可以执行此操作,并将 该权限给 admin 用户,在 admin 用户 给 test 用户 分配新的权限之后,可以立即清除 test 用户的权限缓存。

```
1
2
    * 将该权限赋给 admin用户 使用admin用户清理其他用户的 权限缓存
3
    * @param username
    * @return
 4
5
    */
    @RequiresPermissions("userInfo:clearCache")
6
    @RequestMapping(value = "/clearCache",method = RequestMethod.GET)
7
8
    @ResponseBody
9
    public String clearCache(String username) {
10
11
        String[] keys = new String[3];
12
        keys[0] = "shiro:cache:authenticationCache:"+username;
13
        keys[1] = "shiro:cache:authorizationCache:"+username;
        keys[2] = "shiro:cache:retrylimit:"+username;
14
15
```

我们将在下一篇讨论: SimpleAuthenticationInfo 应该 使用username 还是User实体。