Shiro提供了类似于Spring的Cache抽象，即Shiro本身不实现Cache，但是对Cache进行了又抽象，方便更换不同的底层Cache实现。对于Cache的一些概念可以参考我的《Spring Cache抽象详解》：<http://jinnianshilongnian.iteye.com/blog/2001040>。

**Shiro提供的Cache接口：**

**Java代码  [收藏代码](javascript:void())**

1. **public** **interface** Cache<K, V> {
2. //根据Key获取缓存中的值
3. **public** V get(K key) **throws** CacheException;
4. //往缓存中放入key-value，返回缓存中之前的值
5. **public** V put(K key, V value) **throws** CacheException;
6. //移除缓存中key对应的值，返回该值
7. **public** V remove(K key) **throws** CacheException;
8. //清空整个缓存
9. **public** **void** clear() **throws** CacheException;
10. //返回缓存大小
11. **public** **int** size();
12. //获取缓存中所有的key
13. **public** Set<K> keys();
14. //获取缓存中所有的value
15. **public** Collection<V> values();
16. }

**Shiro提供的CacheManager接口：**

**Java代码  [收藏代码](javascript:void())**

1. **public** **interface** CacheManager {
2. //根据缓存名字获取一个Cache
3. **public** <K, V> Cache<K, V> getCache(String name) **throws** CacheException;
4. }

**Shiro还提供了CacheManagerAware用于注入CacheManager：**

**Java代码  [收藏代码](javascript:void())**

1. **public** **interface** CacheManagerAware {
2. //注入CacheManager
3. **void** setCacheManager(CacheManager cacheManager);
4. }

Shiro内部相应的组件（DefaultSecurityManager）会自动检测相应的对象（如Realm）是否实现了CacheManagerAware并自动注入相应的CacheManager。

本章用例使用了与第六章的代码。

## Realm缓存

Shiro提供了CachingRealm，其实现了CacheManagerAware接口，提供了缓存的一些基础实现；另外AuthenticatingRealm及AuthorizingRealm分别提供了对AuthenticationInfo 和AuthorizationInfo信息的缓存。

**ini配置**

**Java代码  [收藏代码](javascript:void())**

1. userRealm=com.github.zhangkaitao.shiro.chapter11.realm.UserRealm
2. userRealm.credentialsMatcher=$credentialsMatcher
3. userRealm.cachingEnabled=**true**
4. userRealm.authenticationCachingEnabled=**true**
5. userRealm.authenticationCacheName=authenticationCache
6. userRealm.authorizationCachingEnabled=**true**
7. userRealm.authorizationCacheName=authorizationCache
8. securityManager.realms=$userRealm
10. cacheManager=org.apache.shiro.cache.ehcache.EhCacheManager
11. cacheManager.cacheManagerConfigFile=classpath:shiro-ehcache.xml
12. securityManager.cacheManager=$cacheManager

userRealm.cachingEnabled：启用缓存，默认false；

userRealm.authenticationCachingEnabled：启用身份验证缓存，即缓存AuthenticationInfo信息，默认false；

userRealm.authenticationCacheName：缓存AuthenticationInfo信息的缓存名称；

userRealm. authorizationCachingEnabled：启用授权缓存，即缓存AuthorizationInfo信息，默认false；

userRealm. authorizationCacheName：缓存AuthorizationInfo信息的缓存名称；

cacheManager：缓存管理器，此处使用EhCacheManager，即Ehcache实现，需要导入相应的Ehcache依赖，请参考pom.xml；

因为测试用例的关系，需要将Ehcache的CacheManager改为使用VM单例模式：

this.manager = new net.sf.ehcache.CacheManager(getCacheManagerConfigFileInputStream());

改为

this.manager = net.sf.ehcache.CacheManager.create(getCacheManagerConfigFileInputStream());

**测试用例**

**Java代码  [收藏代码](javascript:void())**

1. @Test
2. **public** **void** testClearCachedAuthenticationInfo() {
3. login(u1.getUsername(), password);
4. userService.changePassword(u1.getId(), password + "1");
6. RealmSecurityManager securityManager =
7. (RealmSecurityManager) SecurityUtils.getSecurityManager();
8. UserRealm userRealm = (UserRealm) securityManager.getRealms().iterator().next();
9. userRealm.clearCachedAuthenticationInfo(subject().getPrincipals());
11. login(u1.getUsername(), password + "1");
12. }

首先登录成功（此时会缓存相应的AuthenticationInfo），然后修改密码；此时密码就变了；接着需要调用Realm的clearCachedAuthenticationInfo方法清空之前缓存的AuthenticationInfo；否则下次登录时还会获取到修改密码之前的那个AuthenticationInfo；

**Java代码  [收藏代码](javascript:void())**

1. @Test
2. **public** **void** testClearCachedAuthorizationInfo() {
3. login(u1.getUsername(), password);
4. subject().checkRole(r1.getRole());
5. userService.correlationRoles(u1.getId(), r2.getId());
7. RealmSecurityManager securityManager =
8. (RealmSecurityManager) SecurityUtils.getSecurityManager();
9. UserRealm userRealm = (UserRealm)securityManager.getRealms().iterator().next();
10. userRealm.clearCachedAuthorizationInfo(subject().getPrincipals());
12. subject().checkRole(r2.getRole());
13. }

和之前的用例差不多；此处调用Realm的clearCachedAuthorizationInfo清空之前缓存的AuthorizationInfo；

另外还有clearCache，其同时调用clearCachedAuthenticationInfo和clearCachedAuthorizationInfo，清空AuthenticationInfo和AuthorizationInfo。

UserRealm还提供了clearAllCachedAuthorizationInfo、clearAllCachedAuthenticationInfo、clearAllCache，用于清空整个缓存。

在某些清空下这种方式可能不是最好的选择，可以考虑直接废弃Shiro的缓存，然后自己通过如AOP机制实现自己的缓存；可以参考：

<https://github.com/zhangkaitao/es/tree/master/web/src/main/java/com/sishuok/es/extra/aop>

另外如果和Spring集成时可以考虑直接使用Spring的Cache抽象，可以考虑使用SpringCacheManagerWrapper，其对Spring Cache进行了包装，转换为Shiro的CacheManager实现：

<https://github.com/zhangkaitao/es/blob/master/web/src/main/java/org/apache/shiro/cache/spring/SpringCacheManagerWrapper.java>

## Session缓存

当我们设置了SecurityManager的CacheManager时，如：

**Java代码  [收藏代码](javascript:void())**

1. securityManager.cacheManager=$cacheManager

当我们设置SessionManager时：

**Java代码  [收藏代码](javascript:void())**

1. sessionManager=org.apache.shiro.session.mgt.DefaultSessionManager
2. securityManager.sessionManager=$sessionManager

如securityManager实现了SessionsSecurityManager，其会自动判断SessionManager是否实现了CacheManagerAware接口，如果实现了会把CacheManager设置给它。然后sessionManager会判断相应的sessionDAO（如继承自CachingSessionDAO）是否实现了CacheManagerAware，如果实现了会把CacheManager设置给它；如第九章的MySessionDAO就是带缓存的SessionDAO；其会先查缓存，如果找不到才查数据库。

对于CachingSessionDAO，可以通过如下配置设置缓存的名称：

**Java代码  [收藏代码](javascript:void())**

1. sessionDAO=com.github.zhangkaitao.shiro.chapter11.session.dao.MySessionDAO
2. sessionDAO.activeSessionsCacheName=shiro-activeSessionCache

activeSessionsCacheName默认就是shiro-activeSessionCache。