

# springboot整合shiro-session管理(六)

原文地址，转载请注明出处：[https://blog.csdn.net/qq\\_34021712/article/details/80418112](https://blog.csdn.net/qq_34021712/article/details/80418112) ©王赛超

Shiro 提供了完整的企业级会话管理功能，不依赖于底层容器（如Tomcat），不管是J2SE还是J2EE环境都可以使用，提供了会话管理，会话事件监听，会话存储/持久化，容器无关的集群，失效/过期支持，对Web的透明支持，SSO单点登录的支持等特性。即直接使用 Shiro 的会话管理可以直接替换如 Web 容器的会话管理。

## shiro中的session 特性

- 基于POJO/J2SE：shiro中session相关的类都是基于接口实现的简单的java对象（POJO），兼容所有java对象的配置方式，扩展更方便，完全可以定制自己的会话管理功能。
- 简单灵活的会话存储/持久化：因为shiro中的session对象是基于简单的java对象的，所以你可以将session存储在任何地方，例如，文件，各种数据库，内存中等。
- 容器无关的集群功能：shiro中的session可以很容易的集成第三方的缓存产品完成集群的功能。例如，Ehcache + Terracotta, Coherence, GigaSpaces等。你可以很容易的实现会话集群而无需关注底层的容器实现。
- 异构客户端的访问：可以实现web中的session和非web项目中的session共享。
- 会话事件监听：提供对对session整个生命周期的监听。
- 保存主机地址：在会话开始session会存用户的ip地址和主机名，以此可以判断用户的位置。
- 会话失效/过期的支持：用户长时间处于不活跃状态可以使会话过期，调用touch()方法，可以主动更新最后访问时间，让会话处于活跃状态。
- 透明的Web支持：shiro全面支持Servlet 2.5中的session规范。这意味着你可以将你现有的web程序改为shiro会话，而无需修改代码。
- 单点登录的支持：shiro session基于普通java对象，使得它更容易存储和共享，可以实现跨应用程序共享。可以根据共享的会话，来保证认证状态到另一个程序。从而实现单点登录。

## 会话相关API

```
1 Subject subject = SecurityUtils.getSubject();
2 Session session = subject.getSession();
```

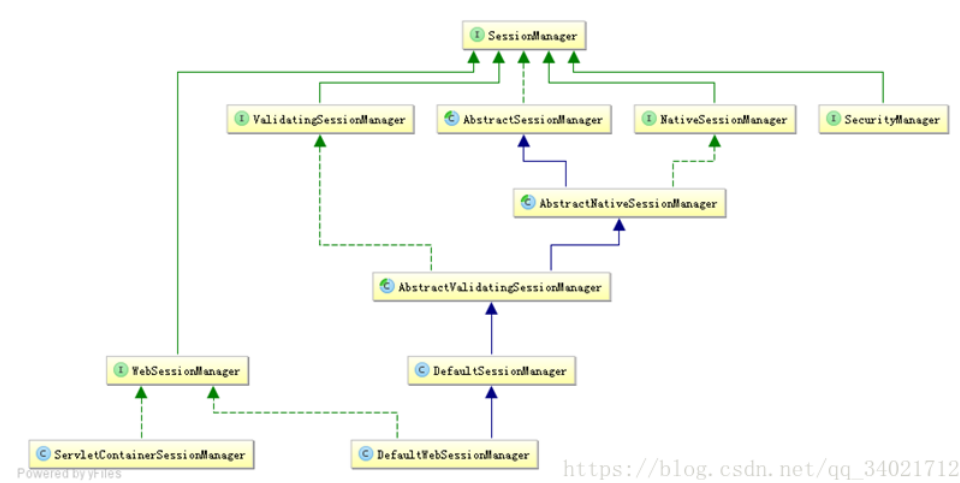
与web中的 HttpServletRequest.getSession(boolean create) 类似!  
Subject.getSession(true)。即如果当前没有创建session对象会创建一个;  
Subject.getSession(false)，如果当前没有创建session对象则返回null。

## 拿到session之后,就可以调用以下API

返回值	方法名	描述
Object	getAttribute(Object key)	根据key标识返回绑定到session的对象
Collection	getAttributeKeys()	获取在session中存储的所有的key
String	getHost()	获取当前主机ip地址，如果未知，返回null
Serializable	getId()	获取session的唯一id
Date	getLastAccessTime()	获取最后的访问时间
Date	getStartTimestamp()	获取session的启动时间
long	getTimeout()	获取session失效时间，单位毫秒
void	setTimeout(long maxIdleTimeInMillis)	设置session的失效时间
Object	removeAttribute(Object key)	通过key移除session中绑定的对象
void	setAttribute(Object key, Object value)	设置session会话属性
void	stop()	销毁会话
void	touch()	更新会话最后访问时间

## 会话管理器

会话管理器管理着应用中所有Subject的会话的创建、维护、删除、失效、验证等工作。是Shiro的核心组件，顶层组件SecurityManager直接继承了SessionManager，且提供了SessionsSecurityManager实现直接把会话管理委托给相应的SessionManager，DefaultSecurityManager及DefaultWebSecurityManager默认SecurityManager都继承了SessionsSecurityManager。  
SecurityManager提供了如下接口：  
`Session start(SessionContext context);` //启动会话  
`Session getSession(SessionKey key) throws SessionException;` //根据会话Key获取会话  
另外用于Web环境的WebSessionManager又提供了如下接口：  
`boolean isServletContainerSessions();` //是否使用Servlet容器的会话  
Shiro还提供了ValidatingSessionManager用于验资并过期会话：  
`void validateSessions();` //验证所有会话是否过期



Shiro提供了三个默认实现：

- DefaultSessionManager**： DefaultSecurityManager使用的默认实现，用于JavaSE环境；
- ServletContainerSessionManager**： DefaultWebSecurityManager使用的默认实现，用于Web环境，其直接使用Servlet容器的会话；
- DefaultWebSessionManager**： 用于Web环境的实现，可以替代ServletContainerSessionManager，自己维护着会话，直接废弃了Servlet容器的会话管理。

### shiro配置会话管理配置

创建session监听类，实现SessionListener接口

```
1 package com.springboot.test.shiro.config.shiro;
2
3 import org.apache.shiro.session.Session;
4 import org.apache.shiro.session.SessionListener;
5
6 import java.util.concurrent.atomic.AtomicInteger;
7
8 /**
9  * @author: wangsaihao
10  * @date: 2018/5/15
11  * @description: 配置session监听器
12  */
13 public class ShiroSessionListener implements SessionListener{
14
15     /**
16      * 统计在线人数
17      * juc包下线程安全自增
18      */
19     private final AtomicInteger sessionCount = new AtomicInteger(0);
20
21     /**
22      * 会话创建时触发
23      * @param session
24      */
25     @Override
26     public void onStart(Session session) {
27         //会话创建，在线人数加一
28         sessionCount.incrementAndGet();
29     }
30
31     /**
32      * 退出会话时触发
33      * @param session
34      */
35     @Override
36     public void onStop(Session session) {
37         //会话退出，在线人数减一
38         sessionCount.decrementAndGet();
39     }
40
41     /**
42      * 会话过期时触发
43      * @param session
44      */
45     @Override
46     public void onExpiration(Session session) {
47         //会话过期，在线人数减一
48         sessionCount.decrementAndGet();
49     }
50 }
```

```
49     }
50     /**
51      * 获取在线人数使用
52      * @return
53      */
54     public AtomicInteger getSessionCount() {
55         return sessionCount;
56     }
57 }
```

在ShiroConfig类中添加以下Bean

配置session监听

```
1  /**
2   * 配置session监听
3   * @return
4   */
5  @Bean("sessionListener")
6  public ShiroSessionListener sessionListener(){
7      ShiroSessionListener sessionListener = new ShiroSessionListener();
8      return sessionListener;
9  }
```

配置会话ID生成器

```
1  /**
2   * 配置会话ID生成器
3   * @return
4   */
5  @Bean
6  public SessionIdGenerator sessionIdGenerator() {
7      return new JavaUuidSessionIdGenerator();
8  }
```

配置sessionDAO

```
1  /**
2   * SessionDAO的作用是为Session提供CRUD并进行持久化的一个shiro组件
3   * MemorySessionDAO 直接在内存中进行会话维护
4   * EnterpriseCacheSessionDAO 提供了缓存功能的会话维护，默认情况下使用MapCache实现，内部使用ConcurrentHashMap保存缓存的会话。
5   * @return
6   */
7  @Bean
8  public SessionDAO sessionDAO() {
9      EnterpriseCacheSessionDAO enterpriseCacheSessionDAO = new EnterpriseCacheSessionDAO();
10     //使用ehCacheManager
11     enterpriseCacheSessionDAO.setCacheManager(ehCacheManager());
12     //设置session缓存的名字 默认为 shiro-activeSessionCache
13     enterpriseCacheSessionDAO.setActiveSessionsCacheName("shiro-activeSessionCache");
14     //sessionId生成器
15     enterpriseCacheSessionDAO.setSessionIdGenerator(sessionIdGenerator());
16     return enterpriseCacheSessionDAO;
17 }
```

配置sessionId的Cookie

```
1  /**
2   * 配置保存sessionId的cookie
3   * 注意：这里的cookie 不是上面的记住我 cookie 记住我需要一个cookie session管理 也需要自己的cookie
4   * @return
5   */
6  @Bean("sessionIdCookie")
7  public SimpleCookie sessionIdCookie(){
8      //这个参数是cookie的名称
9      SimpleCookie simpleCookie = new SimpleCookie("sid");
10     //setcookie的httpOnly属性如果设为true的话，会增加对xss防护的安全系数。它有以下特点：
11
12     //setcookie()的第七个参数
13     //设为true后，只能通过http访问，javascript无法访问
14     //防止xss读取cookie
15     simpleCookie.setHttpOnly(true);
16     simpleCookie.setPath("/");
17     //maxAge=-1表示浏览器关闭时失效此Cookie
18     simpleCookie.setMaxAge(-1);
19     return simpleCookie;
20 }
```

## 配置session管理器

```
1  /**
2   * 配置会话管理器, 设定会话超时及保存
3   * @return
4   */
5  @Bean("sessionManager")
6  public SessionManager sessionManager() {
7
8      DefaultWebSessionManager sessionManager = new DefaultWebSessionManager();
9      Collection<SessionListener> listeners = new ArrayList<SessionListener>();
10     //配置监听
11     listeners.add(sessionListener());
12     sessionManager.setSessionListeners(listeners);
13     sessionManager.setSessionIdCookie(sessionIdCookie());
14     sessionManager.setSessionDAO(sessionDAO());
15     sessionManager.setCacheManager(ehCacheManager());
16
17     return sessionManager;
18 }
19 }
```

注意: 这里的SessionIdCookie 是新建的一个SimpleCookie对象,不是之前整合记住我的那个rememberMeCookie 如果配错了, 就会出现session经典问题: 每次请求都是一个新的session 并且后台报以下异常, 解析的时候报错.因为记住我cookie是加密的用户信息,所以报解密错误

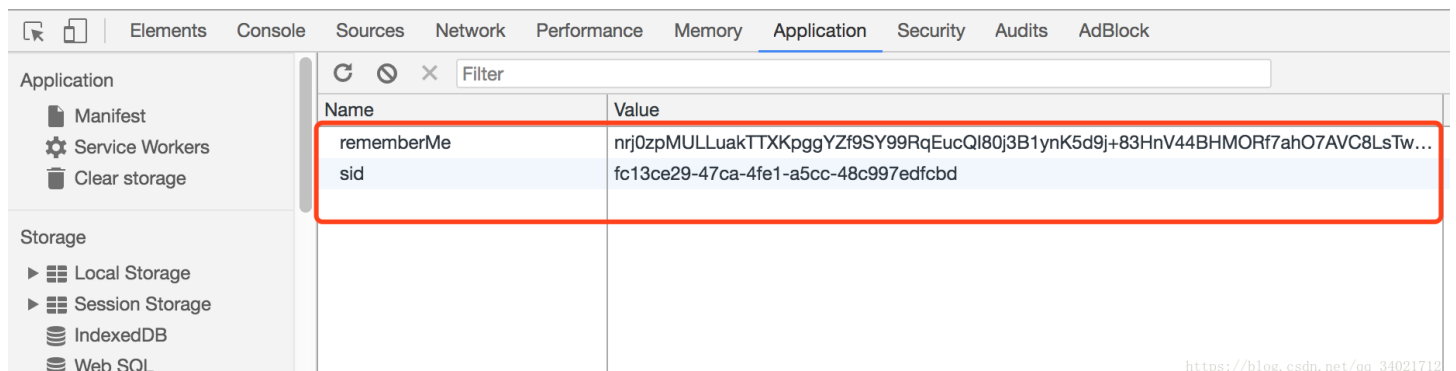
```
1 org.apache.shiro.crypto.CryptoException: Unable to execute 'doFinal' with cipher instance [javax.crypto.Cipher@461df537].
```

## 将session管理器交给SecurityManager

```
1  /**
2   * 配置核心安全事务管理器
3   * @return
4   */
5  @Bean(name="securityManager")
6  public SecurityManager securityManager() {
7      DefaultWebSecurityManager securityManager = new DefaultWebSecurityManager();
8      //设置自定义realm.
9      securityManager.setRealm(shiroRealm());
10     //配置记住我 参考博客:
11     securityManager.setRememberMeManager(rememberMeManager());
12
13     //配置 ehcache缓存管理器 参考博客:
14     securityManager.setCacheManager(ehCacheManager());
15
16     //配置自定义session管理, 使用ehcache 或redis
17     securityManager.setSessionManager(sessionManager());
18
19     return securityManager;
20 }
```

## 启动测试

配置完成之后启动测试,登陆的时候点击 rememberMe 查看cookie 可以看到一个sessionId 和 一个记住我cookie



Name	Value
rememberMe	nrj0zpMULLuakTTXKpggYZf9SY99RqEucQl80j3B1ynK5d9j+83HnV44BHMORf7ahO7AVC8LsTw...
sid	fc13ce29-47ca-4fe1-a5cc-48c997edfcdb

## 配置清理孤立session

以上整合会话管理,还有一个问题: 如果用户如果不点注销, 直接关闭浏览器, 不能够进行session的清空处理, 所以为了防止这样的问题, 还需要增加有一个会话的验证调度。

修改sessionManager如下:

```

1  /**
2   * 配置会话管理器, 设定会话超时及保存
3   * @return
4   */
5  @Bean("sessionManager")
6  public SessionManager sessionManager() {
7
8      DefaultWebSessionManager sessionManager = new DefaultWebSessionManager();
9      Collection<SessionListener> listeners = new ArrayList<SessionListener>();
10     //配置监听
11     listeners.add(sessionListener());
12     sessionManager.setSessionListeners(listeners);
13     sessionManager.setSessionIdCookie(sessionIdCookie());
14     sessionManager.setSessionDAO(sessionDAO());
15     sessionManager.setCacheManager(ehCacheManager());
16
17     //全局会话超时时间(单位毫秒), 默认30分钟 暂时设置为10秒钟 用来测试
18     sessionManager.setGlobalSessionTimeout(1800000);
19     //是否开启删除无效的session对象 默认为true
20     sessionManager.setDeleteInvalidSessions(true);
21     //是否开启定时调度器进行检测过期session 默认为true
22     sessionManager.setSessionValidationSchedulerEnabled(true);
23     //设置session失效的扫描时间, 清理用户直接关闭浏览器造成的孤立会话 默认为 1个小时
24     //设置该属性 就不需要设置 ExecutorServiceSessionValidationScheduler 底层也是默认自动调用ExecutorServiceSessionValidationScheduler
25     //暂时设置为 5秒 用来测试
26     sessionManager.setSessionValidationInterval(3600000);
27
28     return sessionManager;
29 }
30 }

```

测试方法:

设置 全局session超时时间setGlobalSessionTimeout为 10000 ,设置session的失效扫描时间setSessionValidationInterval为5000,然后测试

结果如下:

后台每5秒会打印日志 显示 如下:

```

1  2018/05/23 11:20:11.516 o.a.s.s.m.ExecutorServiceSessionValidationScheduler [] DEBUG Executing session validation...
2  2018/05/23 11:20:11.516 o.a.s.s.m.AbstractValidatingSessionManager [] INFO Validating all active sessions...
3  2018/05/23 11:20:11.516 o.a.s.w.s.m.DefaultWebSessionManager [] DEBUG SessionKey argument is not HTTP compatible or does not have an H
4  2018/05/23 11:20:11.516 o.a.s.s.m.AbstractValidatingSessionManager [] DEBUG Invalidated session with id [2e9e317f-7575-4bb0-98c4-3e6e5
5  2018/05/23 11:20:11.516 o.a.s.s.m.AbstractValidatingSessionManager [] INFO Finished session validation. [1] sessions were stopped.
6  2018/05/23 11:20:11.516 o.a.s.s.m.ExecutorServiceSessionValidationScheduler [] DEBUG Session validation completed successfully in 0 mi

```

session过期之后 再点击连接跳转到首页,并且后台报错 提示 找不到session

```

1  org.apache.shiro.session.UnknownSessionException: There is no session with id [2e9e317f-7575-4bb0-98c4-3e6e5d2578f5]

```

ehcache-shiro.xml添加session缓存属性

```

1  <!-- session缓存 -->
2  <cache name="shiro-activeSessionCache"
3      maxEntriesLocalHeap="2000"
4      eternal="false"
5      timeToIdleSeconds="0"
6      timeToLiveSeconds="0"
7      overflowToDisk="false"
8      statistics="true">
9  </cache>

```

注意: 这里一定要注意缓存的设置过期时间,还有 setGlobalSessionTimeout 的值,任一个时间设置的比较短,session就会从ehcache中清除,到时候就会报  
There is no session with id [2e9e317f-7575-4bb0-98c4-3e6e5d2578f5]

shiro取消url上面的JSESSIONID

```

1  //取消url 后面的 JSESSIONID
2  sessionManager.setSessionIdUrlRewritingEnabled(false);

```

完整的sessionManager如下:

```

1  /**
2   * 配置会话管理器, 设定会话超时及保存
3   * @return
4   */
5  @Bean("sessionManager")
6  public SessionManager sessionManager() {

```

```
7
8     DefaultWebSessionManager sessionManager = new DefaultWebSessionManager();
9     Collection<SessionListener> listeners = new ArrayList<SessionListener>();
10    //配置监听
11    listeners.add(sessionListener());
12    sessionManager.setSessionListeners(listeners);
13    sessionManager.setSessionIdCookie(sessionIdCookie());
14    sessionManager.setSessionDAO(sessionDAO());
15    sessionManager.setCacheManager(ehCacheManager());
16
17    //全局会话超时时间（单位毫秒），默认30分钟 暂时设置为10秒钟 用来测试
18    sessionManager.setGlobalSessionTimeout(10000);
19    //是否开启删除无效的session对象 默认为true
20    sessionManager.setDeleteInvalidSessions(true);
21    //是否开启定时调度器进行检测过期session 默认为true
22    sessionManager.setSessionValidationSchedulerEnabled(true);
23    //设置session失效的扫描时间，清理用户直接关闭浏览器造成的孤立会话 默认为 1个小时
24    //设置该属性 就不需要设置 ExecutorServiceSessionValidationScheduler 底层也是默认自动调用ExecutorServiceSessionValidationScheduler
25    //暂时设置为 5秒 用来测试
26    sessionManager.setSessionValidationInterval(5000);
27
28    //取消url 后面的 JSESSIONID
29    sessionManager.setSessionIdUrlRewritingEnabled(false);
30
31    return sessionManager;
32
33 }
```