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web容器tomcat），不管JavaSE还是JavaEE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。

**会话**

所谓会话，即用户访问应用时保持的连接关系，在多次交互中应用能够识别出当前访问的用户是谁，且可以在多次交互中保存一些数据。如访问一些网站时登录成功后，网站可以记住用户，且在退出之前都可以识别当前用户是谁。

Shiro的会话支持不仅可以在普通的JavaSE应用中使用，也可以在JavaEE应用中使用，如web应用。且使用方式是一致的。

**Java代码  [收藏代码](javascript:void())**

1. login("classpath:shiro.ini", "zhang", "123");
2. Subject subject = SecurityUtils.getSubject();
3. Session session = subject.getSession();

登录成功后使用Subject.getSession()即可获取会话；其等价于Subject.getSession(true)，即如果当前没有创建Session对象会创建一个；另外Subject.getSession(false)，如果当前没有创建Session则返回null（不过默认情况下如果启用会话存储功能的话在创建Subject时会主动创建一个Session）。

**Java代码  [收藏代码](javascript:void())**

1. session.getId();

获取当前会话的唯一标识。

**Java代码  [收藏代码](javascript:void())**

1. session.getHost();

获取当前Subject的主机地址，该地址是通过HostAuthenticationToken.getHost()提供的。

**Java代码  [收藏代码](javascript:void())**

1. session.getTimeout();
2. session.setTimeout(毫秒);

获取/设置当前Session的过期时间；如果不设置默认是会话管理器的全局过期时间。

**Java代码  [收藏代码](javascript:void())**

1. session.getStartTimestamp();
2. session.getLastAccessTime();

获取会话的启动时间及最后访问时间；如果是JavaSE应用需要自己定期调用session.touch()去更新最后访问时间；如果是Web应用，每次进入ShiroFilter都会自动调用session.touch()来更新最后访问时间。

**Java代码  [收藏代码](javascript:void())**

1. session.touch();
2. session.stop();

更新会话最后访问时间及销毁会话；当Subject.logout()时会自动调用stop方法来销毁会话。如果在web中，调用javax.servlet.http.HttpSession. invalidate()也会自动调用Shiro Session.stop方法进行销毁Shiro的会话。

**Java代码  [收藏代码](javascript:void())**

1. session.setAttribute("key", "123");
2. Assert.assertEquals("123", session.getAttribute("key"));
3. session.removeAttribute("key");

设置/获取/删除会话属性；在整个会话范围内都可以对这些属性进行操作。

Shiro提供的会话可以用于JavaSE/JavaEE环境，不依赖于任何底层容器，可以独立使用，是完整的会话模块。

**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：

**Java代码  [收藏代码](javascript:void())**

1. Session start(SessionContext context); //启动会话
2. 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：

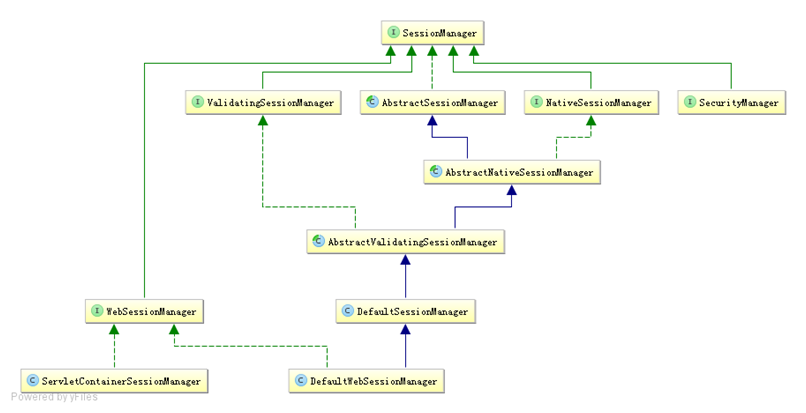
**Java代码  [收藏代码](javascript:void())**

1. **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：

**Java代码  [收藏代码](javascript:void())**

1. **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。

替换SecurityManager默认的SessionManager可以在ini中配置（shiro.ini）：

**Java代码  [收藏代码](javascript:void())**

1. [main]
2. sessionManager=org.apache.shiro.session.mgt.DefaultSessionManager
3. securityManager.sessionManager=$sessionManager

Web环境下的ini配置(shiro-web.ini)：

<!--EndFragment-->

**Java代码  [收藏代码](javascript:void())**

1. [main]
2. sessionManager=org.apache.shiro.web.session.mgt.ServletContainerSessionManager
3. securityManager.sessionManager=$sessionManager

另外可以设置会话的全局过期时间（毫秒为单位），默认30分钟：

**Java代码  [收藏代码](javascript:void())**

1. sessionManager. globalSessionTimeout=1800000

默认情况下globalSessionTimeout将应用给所有Session。可以单独设置每个Session的timeout属性来为每个Session设置其超时时间。

另外如果使用ServletContainerSessionManager进行会话管理，Session的超时依赖于底层Servlet容器的超时时间，可以在web.xml中配置其会话的超时时间（分钟为单位）：

**Java代码  [收藏代码](javascript:void())**

1. <session-config>
2. <session-timeout>30</session-timeout>
3. </session-config>

在Servlet容器中，默认使用JSESSIONID Cookie维护会话，且会话默认是跟容器绑定的；在某些情况下可能需要使用自己的会话机制，此时我们可以使用DefaultWebSessionManager来维护会话：

**Java代码  [收藏代码](javascript:void())**

1. sessionIdCookie=org.apache.shiro.web.servlet.SimpleCookie
2. sessionManager=org.apache.shiro.web.session.mgt.DefaultWebSessionManager
3. sessionIdCookie.name=sid
4. #sessionIdCookie.domain=sishuok.com
5. #sessionIdCookie.path=
6. sessionIdCookie.maxAge=1800
7. sessionIdCookie.httpOnly=**true**
8. sessionManager.sessionIdCookie=$sessionIdCookie
9. sessionManager.sessionIdCookieEnabled=**true**
10. securityManager.sessionManager=$sessionManager

sessionIdCookie是sessionManager创建会话Cookie的模板：

sessionIdCookie.name：设置Cookie名字，默认为JSESSIONID；

sessionIdCookie.domain：设置Cookie的域名，默认空，即当前访问的域名；

sessionIdCookie.path：设置Cookie的路径，默认空，即存储在域名根下；

sessionIdCookie.maxAge：设置Cookie的过期时间，秒为单位，默认-1表示关闭浏览器时过期Cookie；

sessionIdCookie.httpOnly：如果设置为true，则客户端不会暴露给客户端脚本代码，使用HttpOnly cookie有助于减少某些类型的跨站点脚本攻击；此特性需要实现了Servlet 2.5 MR6及以上版本的规范的Servlet容器支持；

sessionManager.sessionIdCookieEnabled：是否启用/禁用Session Id Cookie，默认是启用的；如果禁用后将不会设置Session Id Cookie，即默认使用了Servlet容器的JSESSIONID，且通过URL重写（URL中的“;JSESSIONID=id”部分）保存Session Id。

另外我们可以如“sessionManager. sessionIdCookie.name=sid”这种方式操作Cookie模板。

**会话监听器**

会话监听器用于监听会话创建、过期及停止事件：

**Java代码  [收藏代码](javascript:void())**

1. **public** **class** MySessionListener1 **implements** SessionListener {
2. @Override
3. **public** **void** onStart(Session session) {//会话创建时触发
4. System.out.println("会话创建：" + session.getId());
5. }
6. @Override
7. **public** **void** onExpiration(Session session) {//会话过期时触发
8. System.out.println("会话过期：" + session.getId());
9. }
10. @Override
11. **public** **void** onStop(Session session) {//退出/会话过期时触发
12. System.out.println("会话停止：" + session.getId());
13. }
14. }

如果只想监听某一个事件，可以继承SessionListenerAdapter实现：

**Java代码  [收藏代码](javascript:void())**

1. **public** **class** MySessionListener2 **extends** SessionListenerAdapter {
2. @Override
3. **public** **void** onStart(Session session) {
4. System.out.println("会话创建：" + session.getId());
5. }
6. }

在shiro-web.ini配置文件中可以进行如下配置设置会话监听器：

**Java代码  [收藏代码](javascript:void())**

1. sessionListener1=com.github.zhangkaitao.shiro.chapter10.web.listener.MySessionListener1
2. sessionListener2=com.github.zhangkaitao.shiro.chapter10.web.listener.MySessionListener2
3. sessionManager.sessionListeners=$sessionListener1,$sessionListener2

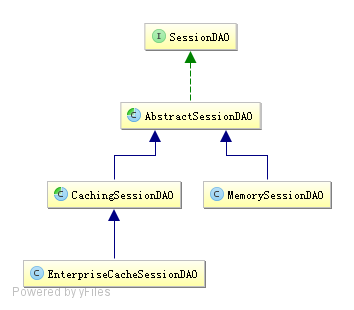
**会话存储/持久化**

Shiro提供SessionDAO用于会话的CRUD，即DAO（Data Access Object）模式实现：

**Java代码  [收藏代码](javascript:void())**

1. //如DefaultSessionManager在创建完session后会调用该方法；如保存到关系数据库/文件系统/NoSQL数据库；即可以实现会话的持久化；返回会话ID；主要此处返回的ID.equals(session.getId())；
2. Serializable create(Session session);
3. //根据会话ID获取会话
4. Session readSession(Serializable sessionId) **throws** UnknownSessionException;
5. //更新会话；如更新会话最后访问时间/停止会话/设置超时时间/设置移除属性等会调用
6. **void** update(Session session) **throws** UnknownSessionException;
7. //删除会话；当会话过期/会话停止（如用户退出时）会调用
8. **void** delete(Session session);
9. //获取当前所有活跃用户，如果用户量多此方法影响性能
10. Collection<Session> getActiveSessions();

Shiro内嵌了如下SessionDAO实现：



AbstractSessionDAO提供了SessionDAO的基础实现，如生成会话ID等；CachingSessionDAO提供了对开发者透明的会话缓存的功能，只需要设置相应的CacheManager即可；MemorySessionDAO直接在内存中进行会话维护；而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。

可以通过如下配置设置SessionDAO：

**Java代码  [收藏代码](javascript:void())**

1. sessionDAO=org.apache.shiro.session.mgt.eis.EnterpriseCacheSessionDAO
2. sessionManager.sessionDAO=$sessionDAO

Shiro提供了使用Ehcache进行会话存储，Ehcache可以配合TerraCotta实现容器无关的分布式集群。

首先在pom.xml里添加如下依赖：

**Java代码  [收藏代码](javascript:void())**

1. <dependency>
2. <groupId>org.apache.shiro</groupId>
3. <artifactId>shiro-ehcache</artifactId>
4. <version>1.2.2</version>
5. </dependency>

接着配置shiro-web.ini文件：

**Java代码  [收藏代码](javascript:void())**

1. sessionDAO=org.apache.shiro.session.mgt.eis.EnterpriseCacheSessionDAO
2. sessionDAO. activeSessionsCacheName=shiro-activeSessionCache
3. sessionManager.sessionDAO=$sessionDAO
4. cacheManager = org.apache.shiro.cache.ehcache.EhCacheManager
5. cacheManager.cacheManagerConfigFile=classpath:ehcache.xml
6. securityManager.cacheManager = $cacheManager

sessionDAO. activeSessionsCacheName：设置Session缓存名字，默认就是shiro-activeSessionCache；

cacheManager：缓存管理器，用于管理缓存的，此处使用Ehcache实现；

cacheManager.cacheManagerConfigFile：设置ehcache缓存的配置文件；

securityManager.cacheManager：设置SecurityManager的cacheManager，会自动设置实现了CacheManagerAware接口的相应对象，如SessionDAO的cacheManager；

然后配置ehcache.xml：

**Java代码  [收藏代码](javascript:void())**

1. <cache name="shiro-activeSessionCache"
2. maxEntriesLocalHeap="10000"
3. overflowToDisk="false"
4. eternal="false"
5. diskPersistent="false"
6. timeToLiveSeconds="0"
7. timeToIdleSeconds="0"
8. statistics="true"/>

Cache的名字为shiro-activeSessionCache，即设置的sessionDAO的activeSessionsCacheName属性值。

另外可以通过如下ini配置设置会话ID生成器：

**Java代码  [收藏代码](javascript:void())**

1. sessionIdGenerator=org.apache.shiro.session.mgt.eis.JavaUuidSessionIdGenerator
2. sessionDAO.sessionIdGenerator=$sessionIdGenerator

用于生成会话ID，默认就是JavaUuidSessionIdGenerator，使用java.util.UUID生成。

如果自定义实现SessionDAO，继承CachingSessionDAO即可：

**Java代码  [收藏代码](javascript:void())**

1. **public** **class** MySessionDAO **extends** CachingSessionDAO {
2. **private** JdbcTemplate jdbcTemplate = JdbcTemplateUtils.jdbcTemplate();
3. **protected** Serializable doCreate(Session session) {
4. Serializable sessionId = generateSessionId(session);
5. assignSessionId(session, sessionId);
6. String sql = "insert into sessions(id, session) values(?,?)";
7. jdbcTemplate.update(sql, sessionId, SerializableUtils.serialize(session));
8. **return** session.getId();
9. }
10. **protected** **void** doUpdate(Session session) {
11. **if**(session **instanceof** ValidatingSession && !((ValidatingSession)session).isValid()) {
12. **return**; //如果会话过期/停止 没必要再更新了
13. }
14. String sql = "update sessions set session=? where id=?";
15. jdbcTemplate.update(sql, SerializableUtils.serialize(session), session.getId());
16. }
17. **protected** **void** doDelete(Session session) {
18. String sql = "delete from sessions where id=?";
19. jdbcTemplate.update(sql, session.getId());
20. }
21. **protected** Session doReadSession(Serializable sessionId) {
22. String sql = "select session from sessions where id=?";
23. List<String> sessionStrList = jdbcTemplate.queryForList(sql, String.**class**, sessionId);
24. **if**(sessionStrList.size() == 0) **return** **null**;
25. **return** SerializableUtils.deserialize(sessionStrList.get(0));
26. }
27. }

doCreate/doUpdate/doDelete/doReadSession分别代表创建/修改/删除/读取会话；此处通过把会话序列化后存储到数据库实现；接着在shiro-web.ini中配置：

**Java代码  [收藏代码](javascript:void())**

1. sessionDAO=com.github.zhangkaitao.shiro.chapter10.session.dao.MySessionDAO

其他设置和之前一样，因为继承了CachingSessionDAO；所有在读取时会先查缓存中是否存在，如果找不到才到数据库中查找。

**会话验证**

Shiro提供了会话验证调度器，用于定期的验证会话是否已过期，如果过期将停止会话；出于性能考虑，一般情况下都是获取会话时来验证会话是否过期并停止会话的；但是如在web环境中，如果用户不主动退出是不知道会话是否过期的，因此需要定期的检测会话是否过期，Shiro提供了会话验证调度器SessionValidationScheduler来做这件事情。

可以通过如下ini配置开启会话验证：

**Java代码  [收藏代码](javascript:void())**

1. sessionValidationScheduler=org.apache.shiro.session.mgt.ExecutorServiceSessionValidationScheduler
2. sessionValidationScheduler.interval = 3600000
3. sessionValidationScheduler.sessionManager=$sessionManager
4. sessionManager.globalSessionTimeout=1800000
5. sessionManager.sessionValidationSchedulerEnabled=**true**
6. sessionManager.sessionValidationScheduler=$sessionValidationScheduler

sessionValidationScheduler：会话验证调度器，sessionManager默认就是使用ExecutorServiceSessionValidationScheduler，其使用JDK的ScheduledExecutorService进行定期调度并验证会话是否过期；

sessionValidationScheduler.interval：设置调度时间间隔，单位毫秒，默认就是1小时；

sessionValidationScheduler.sessionManager：设置会话验证调度器进行会话验证时的会话管理器；

sessionManager.globalSessionTimeout：设置全局会话超时时间，默认30分钟，即如果30分钟内没有访问会话将过期；

sessionManager.sessionValidationSchedulerEnabled：是否开启会话验证器，默认是开启的；

sessionManager.sessionValidationScheduler：设置会话验证调度器，默认就是使用ExecutorServiceSessionValidationScheduler。

Shiro也提供了使用Quartz会话验证调度器：

**Java代码  [收藏代码](javascript:void())**

1. sessionValidationScheduler=org.apache.shiro.session.mgt.quartz.QuartzSessionValidationScheduler
2. sessionValidationScheduler.sessionValidationInterval = 3600000
3. sessionValidationScheduler.sessionManager=$sessionManager

使用时需要导入shiro-quartz依赖：

**Java代码  [收藏代码](javascript:void())**

1. <dependency>
2. <groupId>org.apache.shiro</groupId>
3. <artifactId>shiro-quartz</artifactId>
4. <version>1.2.2</version>
5. </dependency>

如上会话验证调度器实现都是直接调用AbstractValidatingSessionManager 的validateSessions方法进行验证，其直接调用SessionDAO的getActiveSessions方法获取所有会话进行验证，如果会话比较多，会影响性能；可以考虑如分页获取会话并进行验证，如com.github.zhangkaitao.shiro.chapter10.session.scheduler.MySessionValidationScheduler：

**Java代码  [收藏代码](javascript:void())**

1. //分页获取会话并验证
2. String sql = "select session from sessions limit ?,?";
3. **int** start = 0; //起始记录
4. **int** size = 20; //每页大小
5. List<String> sessionList = jdbcTemplate.queryForList(sql, String.**class**, start, size);
6. **while**(sessionList.size() > 0) {
7. **for**(String sessionStr : sessionList) {
8. **try** {
9. Session session = SerializableUtils.deserialize(sessionStr);
10. Method validateMethod =
11. ReflectionUtils.findMethod(AbstractValidatingSessionManager.**class**,
12. "validate", Session.**class**, SessionKey.**class**);
13. validateMethod.setAccessible(**true**);
14. ReflectionUtils.invokeMethod(validateMethod,
15. sessionManager, session, **new** DefaultSessionKey(session.getId()));
16. } **catch** (Exception e) {
17. //ignore
18. }
19. }
20. start = start + size;
21. sessionList = jdbcTemplate.queryForList(sql, String.**class**, start, size);
22. }

其直接改造自ExecutorServiceSessionValidationScheduler，如上代码是验证的核心代码，可以根据自己的需求改造此验证调度器器；ini的配置和之前的类似。

如果在会话过期时不想删除过期的会话，可以通过如下ini配置进行设置：

**Java代码  [收藏代码](javascript:void())**

1. sessionManager.deleteInvalidSessions=**false**

默认是开启的，在会话过期后会调用SessionDAO的delete方法删除会话：如会话时持久化存储的，可以调用此方法进行删除。

如果是在获取会话时验证了会话已过期，将抛出InvalidSessionException；因此需要捕获这个异常并跳转到相应的页面告诉用户会话已过期，让其重新登录，如可以在web.xml配置相应的错误页面：

**Java代码  [收藏代码](javascript:void())**

1. <error-page>
2. <exception-type>org.apache.shiro.session.InvalidSessionException</exception-type>
3. <location>/invalidSession.jsp</location>
4. </error-page>

**sessionFactory**

sessionFactory是创建会话的工厂，根据相应的Subject上下文信息来创建会话；默认提供了SimpleSessionFactory用来创建SimpleSession会话。

首先自定义一个Session：

**Java代码  [收藏代码](javascript:void())**

1. **public** **class** OnlineSession **extends** SimpleSession {
2. **public** **static** **enum** OnlineStatus {
3. on\_line("在线"), hidden("隐身"), force\_logout("强制退出");
4. **private** **final** String info;
5. **private** OnlineStatus(String info) {
6. **this**.info = info;
7. }
8. **public** String getInfo() {
9. **return** info;
10. }
11. }
12. **private** String userAgent; //用户浏览器类型
13. **private** OnlineStatus status = OnlineStatus.on\_line; //在线状态
14. **private** String systemHost; //用户登录时系统IP
15. //省略其他
16. }

OnlineSession用于保存当前登录用户的在线状态，支持如离线等状态的控制。

接着自定义SessionFactory：

**Java代码  [收藏代码](javascript:void())**

1. **public** **class** OnlineSessionFactory **implements** SessionFactory {
3. @Override
4. **public** Session createSession(SessionContext initData) {
5. OnlineSession session = **new** OnlineSession();
6. **if** (initData != **null** && initData **instanceof** WebSessionContext) {
7. WebSessionContext sessionContext = (WebSessionContext) initData;
8. HttpServletRequest request = (HttpServletRequest) sessionContext.getServletRequest();
9. **if** (request != **null**) {
10. session.setHost(IpUtils.getIpAddr(request));
11. session.setUserAgent(request.getHeader("User-Agent"));
12. session.setSystemHost(request.getLocalAddr() + ":" + request.getLocalPort());
13. }
14. }
15. **return** session;
16. }
17. }

根据会话上下文创建相应的OnlineSession。

最后在shiro-web.ini配置文件中配置：

**Java代码  [收藏代码](javascript:void())**

1. sessionFactory=org.apache.shiro.session.mgt.OnlineSessionFactory
2. sessionManager.sessionFactory=$sessionFactory