Shiro 实现 Tomcat 集群的 Session 共享

一、背景

Session 共享有多种方案，之前写过《Spring Session 实现 Tomcat 集群的 Session 共享》文章，功能实现起来非常简单和方便。

最近在学习 Shiro 框架，Shiro 也提供了会话管理的功能。如果项目中选用 Shiro 作为权限控制的方案，同时项目又需要集群，那么可以自定义 sessionDAO 来实现 Session 共享。

二、实现

> JDK：1.8
>
> JDK：1.8
> 容器：Tomcat 8
>
> JDK：1.8
> 容器：Tomcat 8
> Session 存储容器：Redis 3.2.0

测试环境与测试 Spring Session 时的一样，将项目部署到同一台虚拟机上的 2 个 tomcat 中，使用 8080 和 8081 端口启动。

下边列出主要配置，Shiro 所依赖的 jar 配置和运行配置忽略，具体代码可以下载由下文提供的源码进行查看。

2.1 applicationContext-shiro.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                    http://www.springframework.org/schema/beans/spring-
beans-4.0.xsd
                    http://www.springframework.org/schema/context
                    http://www.springframework.org/schema/context/spring-
context-4.0.xsd
                    http://www.springframework.org/schema/tx
                    http://www.springframework.org/schema/tx/spring-tx-
4.0.xsd
                    http://www.springframework.org/schema/aop
                    http://www.springframework.org/schema/aop/spring-aop-
4.0.xsd">

    <context:component-scan base-package="com.light.dao.*"/>

    <!-- redis 连接池 -->
      <bean id="jedisPoolConfig"
class="redis.clients.jedis.JedisPoolConfig">
        <property name="maxTotal" value="20"></property>
        <property name="maxIdle" value="1"></property>
      </bean>

      <!-- redis 连接工厂 -->
      <bean id="jedisConnectionFactory"
class="org.springframework.data.redis.connection.jedis.JedisConnectionFact
ory"
      destroy-method="destroy">
        <property name="hostName" value="192.168.2.11"/>
        <property name="port" value="6379"/>
        <property name="timeout" value="5000"/>
        <property name="password" value=""/>
        <property name="usePool" value="true"/>
        <property name="poolConfig" ref="jedisPoolConfig"/>
      </bean>
```

```
      <!-- redis 模板 -->
      <bean id="redisTemplate"
class="org.springframework.data.redis.core.RedisTemplate" >
        <property name="connectionFactory" ref="jedisConnectionFactory" />
    </bean >

      <!-- Shiro 的Web过滤器 -->
    <bean id="shiroFilter"
class="org.apache.shiro.spring.web.ShiroFilterFactoryBean">
        <property name="securityManager" ref="securityManager" />
        <property name="loginUrl" value="/index.jsp" />
        <!-- 过虑器链定义，从上向下顺序执行，一般将/**放在最下边 -->
        <property name="filterChainDefinitions">
            <value>
                /resources/**=anon
                /login=anon
            </value>
        </property>
    </bean>

    <!-- 安全管理器 -->
    <bean id="securityManager"
class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">
        <property name="sessionManager" ref="sessionManager" />
    </bean>

    <!-- 会话管理器 -->
    <bean id="sessionManager"
class="org.apache.shiro.web.session.mgt.DefaultWebSessionManager">
        <property name="sessionDAO" ref="sessionDAO"></property>
    </bean>

    <!-- 自定义 sessionDAO -->
    <bean id="sessionDAO" class="com.light.dao.CustomSessionDAO"></bean>
</beans>
```

2.2 自定义 sessionDAO

自定义 sessionDAO 需要继承 AbstractSessionDAO 类来重写 session 的 CRUD。

```
public class CustomSessionDAO extends AbstractSessionDAO {

    private static final int EXPIRE_TIME = 600;

    @Resource(name="redisTemplate")
    private RedisTemplate<String,Object> redisTemplate;


    public void update(Session session) throws UnknownSessionException {
        this.redisTemplate.opsForValue().set(
                session.getId().toString(),
                session,
                EXPIRE_TIME,
                TimeUnit.SECONDS);
    }

    public void delete(Session session) {
        this.redisTemplate.delete(session.getId().toString());
    }

    public Collection<Session> getActiveSessions() {
        // TODO
        return null;
    }

    @Override
    protected Serializable doCreate(Session session) {
        // 生成 sessionId
        Serializable sessionId = this.generateSessionId(session);
        // session 绑定 sessionId
        this.assignSessionId(session, sessionId);

        this.redisTemplate.opsForValue().set(
                session.getId().toString(),
                session,
                EXPIRE_TIME,
                TimeUnit.SECONDS);

        return sessionId;
    }
```

```
    @Override
    protected Session doReadSession(Serializable sessionId) {
        Session session = (Session)
this.redisTemplate.opsForValue().get(sessionId.toString());

        if (session != null) {
            this.redisTemplate.opsForValue().set(
                    session.getId().toString(),
                    session,
                    EXPIRE_TIME,
                    TimeUnit.SECONDS);
        }

        return session;
    }

}
```

CustomSessionDAO 类是实现 **session** 共享的核心。

3.3 后端代码

```
@Controller
public class LoginController {

    @Autowired
    private SecurityManager sm;

    @RequestMapping("login")
    public String login(String userName, String
password,HttpServletRequest request) {
        // 首次登录
        if ("admin".equals(userName) && "admin".equals(password)) {


            SecurityUtils.setSecurityManager(sm);
            Subject subject = SecurityUtils.getSubject();
            // 使用 shiro 的 session 保存数据
            Session session = subject.getSession();
            session.setAttribute("userName", userName);

            return "manageUI";
        }

        // 如果已经登录过，从另一个 tomcat 访问该方法，跳转到 manageUI 页面可以
查看 session 信息
        if ("".equals(userName) && "".equals(password)) {
            return "manageUI";
        }

        return "redirect:/index.jsp";
    }

    @RequestMapping("logout")
    public String logout(HttpSession session) {

        session.removeAttribute("userName");
        session.removeAttribute("url");

        return "redirect:/index.jsp";
    }
```

```
}
```

注意：后端代码使用的是 **Shiro** 提供的 **session API** 进行保存数据。

3.4 前端代码

index.jsp 页面：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">
    <title>登陆界面</title>
    <link href="/resources/css/bootstrap.min.css" rel="stylesheet">
    <style>
        html {
            background: url("/resources/images/bg.png") no-repeat center
center;
        }
        label {
            color: #fff;
        }
        .container {
            position:absolute;
            top:50%;
            left:50%;
            margin-top: -115px;
            margin-left: -250px;
            width: 500px;
            height:230px;
            padding:50px;
            border: 2px solid #eee;
            border-radius: 5px;
            box-shadow:5px 5px 16px #000;
        }
    </style>
</head>

<body>
    <div class="container">
        <form class="form-horizontal" role="form" action="/login"
method="post">
```

```
            <div class="form-group">
                <label for="inputEmail3" class="col-sm-2 control-label">用
户名</label>
                <div class="col-sm-10">
                    <input type="text" class="form-control"
name="userName" placeholder="用户名">
                </div>
            </div>
            <div class="form-group">
                <label for="inputPassword3" class="col-sm-2 control-
label">密码</label>
                <div class="col-sm-10">
                    <input type="password" class="form-control"
name="password" placeholder="密码">
                </div>
            </div>
            <div class="form-group">
                <div class="col-sm-offset-2 col-sm-10">
                    <button type="submit" class="btn btn-primary"
style="width: 100%">登陆</button>
                </div>
            </div>
        </form>
    </div>
</body>
</html>
```

manageUI.jsp 页面：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">
    <title>管理界面</title>
    <link href="/resources/css/bootstrap.min.css" rel="stylesheet">
</head>

<body>
    <div class="container">
      <div class="jumbotron">
        <h3>测试 Shiro 实现 session 共享</h3>
        <h3>端口为 8080 的页面</h3>
        <h3>用户名：${sessionScope.userName}（session 域数据）</h3>
        <p><a class="btn btn-lg btn-success" href="/logout" role="button">
注销</a></p>
      </div>
    </div>
</body>
</html>
```

注意：**8081** 项目的页面需要改成 **“**端口为 **8081** 的页面**”**。

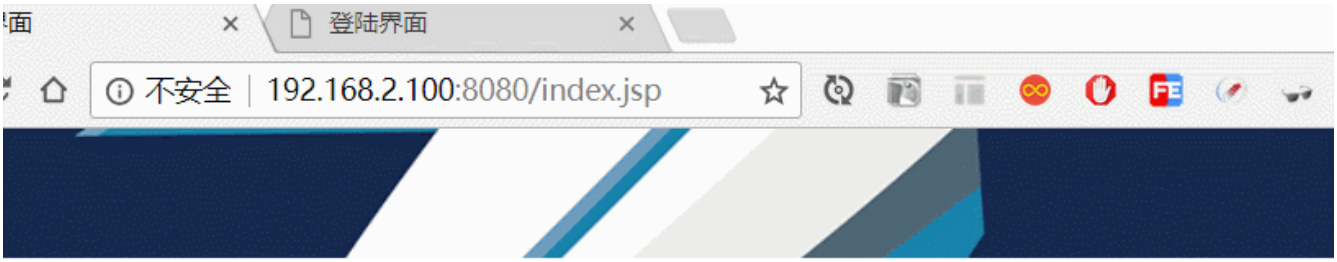
三、演示

测试步骤同样与测试 Spring Session 时的一致。

预期效果：

1. 首先访问 8080 端口的项目并进行登陆操作，跳转到管理界面并显示保存的信息。
2. 在同个浏览器中访问 8081 端口项目的页面，不需要输入账号密码直接点击登陆按钮，会直接跳转到管理界面。如果 session 实现了共享，那么在管理界面就可以查看由 8080 端口项目保存在 session 的信息。否则反之。

演示图如下：



总体来说，功能实现不算困难，但是比使用 Spring session 方案要麻烦一些，因为需要开发者自己实现 session 的 CRUD。正因为需要手动实现，从另方面考虑使用 Shiro 方案管理 session 会比较灵活。

四、源码下载

[session-share]