# Shiro学习文档

## 1.功能介绍

系统存在三个用户

账号：admin 密码：admin 角色：admin

账号：lk 密码：lk 角色：visitor

账号：test 密码：123456 角色：test

用户进入系统界面可以看到用户列表，

不同的是：

admin 可以看到添加用户链接，和跳转到权限操作链接，可以看到删改查按钮。

Visitor只能看到查询按钮

Test 只能看到查询和修改按钮。

## 2. shiro配置文件介绍

<!-- 自定义realm -->

<**bean** id=*"customRealm"* class=*"com.lk.play.common.security.MyRealm"*></**bean**>

<!-- web.xml中shiro的filter对应的bean -->

<**bean** id=*"shiroFilter"* class=*"org.apache.shiro.spring.web.ShiroFilterFactoryBean"*>

<!-- 管理器，必须设置 -->

<**property** name=*"securityManager"* ref=*"securityManager"* />

<!-- 拦截到，跳转到的地址,通过此地址去认证 -->

<**property** name=*"loginUrl"* value=*"/login.jsp"* />

<!-- 通过unauthorizedUrl指定没有权限操作时跳转页面 -->

<**property** name=*"unauthorizedUrl"* value=*"/error.jsp"* />

<**property** name=*"filterChainDefinitions"*>

<**value**>

/test/login/loginOn=anon

/test/user/\* = authc

/test/role/\* = authc

</**value**>

</**property**>

</**bean**>

<!-- securityManager安全管理器 -->

<**bean** id=*"securityManager"* class=*"org.apache.shiro.web.mgt.DefaultWebSecurityManager"*>

<**property** name=*"realm"* ref=*"customRealm"* />

<!-- 注入缓存管理器 -->

<**property** name=*"cacheManager"* ref=*"cacheManager"* />

<!-- 记住我 -->

<**property** name=*"rememberMeManager"* ref=*"rememberMeManager"* />

</**bean**>

<!-- 缓存管理器 -->

<**bean** id=*"cacheManager"* class=*"org.apache.shiro.cache.ehcache.EhCacheManager"*>

<**property** name=*"cacheManagerConfigFile"* value=*"classpath:ehcache.xml"* />

</**bean**>

<!-- 记住我cookie -->

<**bean** id=*"rememberMeCookie"* class=*"org.apache.shiro.web.servlet.SimpleCookie"*>

<!-- rememberMe是cookie的名字 -->

<**constructor-arg** value=*"rememberMe"* />

<!-- 记住我cookie生效时间30天 -->

<**property** name=*"maxAge"* value=*"2592000"* />

</**bean**>

<!-- rememberMeManager管理器，写cookie，取出cookie生成用户信息 -->

<**bean** id=*"rememberMeManager"* class=*"org.apache.shiro.web.mgt.CookieRememberMeManager"*>

<**property** name=*"cookie"* ref=*"rememberMeCookie"* />

</**bean**>

**Caching**：缓存，比如用户登录后，其用户信息、拥有的角色/权限不必每次去查，这样可以提高效率；

**Remember Me：**记住我，这个是非常常见的功能，即一次登录后，下次再来的话不用登录了。

anon: 表示可匿名使用,可以理解为匿名用户或游客

authc    : 表示需认证才能使用

其他详细见：<https://blog.csdn.net/userrefister/article/details/47807075>

## 3.shiro后台相关代码

### 登陆验证

***@RequestMapping***(value = "/loginOn")

**public** **String** **loginOn**(***@ModelAttribute***("user") **User** user,Model model) {

//登陆令牌验证

Subject **subject** = **SecurityUtils**.*getSubject*();

**UsernamePasswordToken** **token** = **new** UsernamePasswordToken(user.getAccount(), user.getPassword());

//获取用户列表

List<User> **userList** = userService.findAllList();

model.addAttribute("userList",userList);

**try** {

subject.login(token);

} **catch** (**AuthenticationException** **e**) {

// **TODO** Auto-generated catch block

e.printStackTrace();

**return** "error";

}

**return** "userList";

}

### 2.自定义realm

public class MyRealm extends AuthorizingRealm {

@Resource

private UserService userService;

@Resource

private RoleService rolerService;

/\*\*

\* 权限验证时调用

\*/

@Override

protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principals) {

//获取用户名

String userName=(String)principals.getPrimaryPrincipal();

SimpleAuthorizationInfo authorizationInfo=new SimpleAuthorizationInfo();

User user = userService.getByUserName(userName);

//进行授权角色

authorizationInfo.addRole(rolerService.getRole(user.getId()));

//进行授权权限

return authorizationInfo;

}

/\*\*

\* 登陆时验证时调用

\*/

@Override

protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token) throws AuthenticationException {

UsernamePasswordToken ustoken=(UsernamePasswordToken)token;

String userName = ustoken.getUsername();

//根据用户名查找用户信息

User user=userService.getByUserName(userName);

if(user!=null){

AuthenticationInfo authcInfo=new SimpleAuthenticationInfo(user.getAccount(),user.getPassword(),getName());

return authcInfo;

}else{

return null;

}

}

}

## 4.shiro前台相关代码

<body>

<div>

<shiro:hasRole name="admin">

<a href="${ctx }/user/form">添加用户</a><br>

<a href="${ctx }/role/list">跳转到权限页面</a>

</shiro:hasRole>

</div>

<div>

<table>

<thead>

<tr>

<th>姓名</th>

<th>账号</th>

<th>操作</th>

</tr>

</thead>

<tbody>

<c:forEach items="${userList }" var="user">

<tr>

<td>${user.name }</td>

<td>${user.account }</td>

<td>

<button onclick="viewUser('${user.id}')">查看</button>

<shiro:hasAnyRoles name="admin,test">

<button onclick="updateUser('${user.id}')">修改</button>

</shiro:hasAnyRoles >

<shiro:hasRole name="admin">

<button onclick="deleteUser('${user.id}')">删除</button>

</shiro:hasRole>

</td>

</tr>

</c:forEach>

</tbody>

</table>

</div>

</body>

## 5.项目shiro执行流程

用户登录时信息保存到UsernamePasswordToken 对象中，subject的login方法到myRealm中进行验证，验证调用doGetAuthenticationInfo进行用户验证，验证成功后将用户信息存入，AuthenticationInfo对象中。

当进行权限验证时，会调用myRealm中的doGetAuthorizationInfo方法进行授权（角色，权限）

## 6.密码加密/解密（添加时间：2018/4/16 ）

### 1.userService

/\*\*

\* 添加用户

\*/

@Override

public void insert(User user) {

if(user.getId() == "") {

// uuid自动生成主键

String id = UUID.randomUUID().toString().replaceAll("-", "");

if(user.getId() == "") {

user.setId(id);

}

// 密码加密存储

if (user.getPassword() != null) {

user.setPassword(SystemService.encodePassword(user.getPassword(),user.getId()));

}

userDao.insert(user);

//将用户id与角色id存入到中间库

insertUserRole(user);

}else {

// 密码加密存储

if (user.getPassword() != null) {

user.setPassword(SystemService.encodePassword(user.getPassword(),user.getId()));

}

update(user);

//更新中间库

updateUserRole(user);

}

}

### 2.systemService

/\*\*

\* sha-1加密1024次,使用主键id值为盐

\* @param password 密码

\* @param salt 盐

\* @return

\*/

public static String encodePassword(String password,String salt) {

String hash = new SimpleHash(MD5, password, salt, HASH\_INTERATIONS).toString();

return hash;

}

### 3.myRealm

/\*\*

\* 登陆时验证时调用

\*/

@Override

protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token) throws AuthenticationException {

UsernamePasswordToken ustoken=(UsernamePasswordToken)token;

String userName = ustoken.getUsername();

//根据用户名查找用户信息

User user=userService.getByUserName(userName);

if(user!=null){

String salt = user.getId();

AuthenticationInfo authcInfo=new SimpleAuthenticationInfo(user.getAccount(),

user.getPassword(), ByteSource.Util.bytes(salt), getName());

return authcInfo;

}else{

throw new UnknownAccountException();//没有找到账号

}

}

### 4.spring-context-shiro

<!-- 自定义realm -->

<bean id="customRealm" class="com.lk.play.common.security.MyRealm">

<!-- <property name="credentialsMatcher" ref="credentialsMatcher"></property> -->

<!-- 凭证匹配器 -->

<property name="credentialsMatcher">

<bean class="org.apache.shiro.authc.credential.HashedCredentialsMatcher">

<property name="hashAlgorithmName" value="MD5"></property><!-- 加密算法的名称 -->

<property name="hashIterations" value="1024"></property> <!-- 配置加密的次数 -->

</bean>

</property>

</bean>