# **1，pom中加入依赖**

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

<version>1.5.6.RELEASE</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.thymeleaf/thymeleaf -->

<dependency>

<groupId>org.thymeleaf</groupId>

<artifactId>thymeleaf</artifactId>

<version>${thymeleaf.version}</version>

</dependency>

<!-- shiro安全框架 -->

<dependency>

<groupId>org.apache.shiro</groupId>

<artifactId>shiro-spring</artifactId>

<version>1.4.0</version>

</dependency>

<!--thymeleaf-shiro-extras-->

<dependency>

<groupId>com.github.theborakompanioni</groupId>

<artifactId>thymeleaf-extras-shiro</artifactId>

<version>1.2.1</version>

</dependency>

# **2，用户-角色-权限的表关系**

//用户表

public class User {

private Integer userId;

private String userName;

private Set<Role> roles = new HashSet<>();

}

//角色表

public class User {

private Integer id;

private String role;

private Set<Module> modules = new HashSet<>();

private Set<User> users = new HashSet<>();

}

//权限表

public class Module {

private Integer mid;

private String mname;

private Set<Role> roles = new HashSet<>();

}

//用户查询

<resultMap id="BaseResultMap" type="com.lanyu.common.model.User" >

<id column="user\_id" property="userId" jdbcType="INTEGER" />

<result column="user\_name" property="userName" jdbcType="VARCHAR" />

<!-- 多对多关联映射：collection -->

<collection property="roles" ofType="Role">

<id property="id" column="c\_id" />

<result property="role" column="role" />

<collection property="modules" ofType="Module">

<id property="mid" column="mid"/>

<result property="mname" column="mname"/>

</collection>

</collection>

</resultMap>

//查询用户信息，返回结果会自动分组，得到用户信息

<select id="selectByPhone" resultMap="BaseResultMap" parameterType="java.lang.String" >

SELECT

u.\*, r.\*, m.\*

FROM

sys\_user u

INNER JOIN sys\_user\_role ur ON ur.userId = u.user\_id

INNER JOIN sys\_role r ON r.rid = ur.roleid

INNER JOIN sys\_role\_module mr ON mr.rid = r.rid

INNER JOIN sys\_module m ON mr.mid = m.mid

WHERE

u.user\_name=#{username} or u.phone=#{username};

</select>

# **3，编写shiro核心类**

@Configuration

public class ShiroConfiguration {

//用于thymeleaf模板使用shiro标签

@Bean

public ShiroDialect shiroDialect() {

return new ShiroDialect();

}

@Bean(name="shiroFilter")

public ShiroFilterFactoryBean shiroFilter(@Qualifier("securityManager") SecurityManager manager) {

ShiroFilterFactoryBean bean=new ShiroFilterFactoryBean();

bean.setSecurityManager(manager);

//配置登录的url和登录成功的url

bean.setLoginUrl("/loginpage");

bean.setSuccessUrl("/indexpage");

//配置访问权限

LinkedHashMap<String, String> filterChainDefinitionMap=new LinkedHashMap<>();

// filterChainDefinitionMap.put("/loginpage\*", "anon"); //表示可以匿名访问

filterChainDefinitionMap.put("/admin/\*", "authc");//表示需要认证才可以访问

filterChainDefinitionMap.put("/logout\*","anon");

filterChainDefinitionMap.put("/img/\*\*","anon");

filterChainDefinitionMap.put("/js/\*\*","anon");

filterChainDefinitionMap.put("/css/\*\*","anon");

filterChainDefinitionMap.put("/fomts/\*\*","anon");

filterChainDefinitionMap.put("/\*\*", "anon");

bean.setFilterChainDefinitionMap(filterChainDefinitionMap);

return bean;

}

//配置核心安全事务管理器

@Bean(name="securityManager")

public SecurityManager securityManager(@Qualifier("authRealm") AuthRealm authRealm) {

System.err.println("--------------shiro已经加载----------------");

DefaultWebSecurityManager manager=new DefaultWebSecurityManager();

manager.setRealm(authRealm);

return manager;

}

//配置自定义的权限登录器

@Bean(name="authRealm")

public AuthRealm authRealm(@Qualifier("credentialsMatcher") CredentialsMatcher matcher) {

AuthRealm authRealm=new AuthRealm();

authRealm.setCredentialsMatcher(matcher);

return authRealm;

}

//配置自定义的密码比较器

@Bean(name="credentialsMatcher")

public CredentialsMatcher credentialsMatcher() {

return new CredentialsMatcher();

}

@Bean

public LifecycleBeanPostProcessor lifecycleBeanPostProcessor(){

return new LifecycleBeanPostProcessor();

}

@Bean

public DefaultAdvisorAutoProxyCreator defaultAdvisorAutoProxyCreator(){

DefaultAdvisorAutoProxyCreator creator=new DefaultAdvisorAutoProxyCreator();

creator.setProxyTargetClass(true);

return creator;

}

@Bean

public AuthorizationAttributeSourceAdvisor authorizationAttributeSourceAdvisor(@Qualifier("securityManager") SecurityManager manager) {

AuthorizationAttributeSourceAdvisor advisor=new AuthorizationAttributeSourceAdvisor();

advisor.setSecurityManager(manager);

return advisor;

}

}

— - - -- - -- - -- - -- - - -- - - - --

public class AuthRealm extends AuthorizingRealm {

@Autowired

private UserService userService;

//认证.登录

@Override

protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token) throws AuthenticationException {

UsernamePasswordToken utoken=(UsernamePasswordToken) token;//获取用户输入的token

String username = utoken.getUsername();

User user = userService.selectByPhone(username);

return new SimpleAuthenticationInfo(user, user.getPassword(),this.getClass().getName());//放入shiro.调用CredentialsMatcher检验密码

}

//授权

@Override

protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principal) {

User user=(User) principal.fromRealm(this.getClass().getName()).iterator().next();//获取session中的用户

List<String> permissions=new ArrayList<>();

Set<Role> roles = user.getRoleList();

SimpleAuthorizationInfo info=new SimpleAuthorizationInfo();

List<String> listrole = new ArrayList<>();

if(roles.size()>0) {

for(Role role : roles) {

if(!listrole.contains(role.getRole())){

listrole.add(role.getRole());

}

Set<Module> modules = role.getModules();

if(modules.size()>0) {

for(Module module : modules) {

permissions.add(module.getMname());

}

}

}

}

info.addRoles(listrole); //将角色放入shiro中.

info.addStringPermissions(permissions); //将权限放入shiro中.

return info;

}

}

//自定义密码比较器

public class CredentialsMatcher extends SimpleCredentialsMatcher {

private Logger logger = Logger.getLogger(CredentialsMatcher.class);

@Override

public boolean doCredentialsMatch(AuthenticationToken token, AuthenticationInfo info) {

UsernamePasswordToken utoken=(UsernamePasswordToken) token;

//所需加密的参数 即 用户输入的密码

String source = String.valueOf(utoken.getPassword());

//[盐] 一般为用户名 或 随机数

String salt = utoken.getUsername();

//加密次数

int hashIterations = 50;

SimpleHash sh = new SimpleHash("md5", source, salt, hashIterations);

String Strsh =sh.toHex();

//打印最终结果

logger.info("正确密码为："+Strsh);

//获得数据库中的密码

String dbPassword= (String) getCredentials(info);

logger.info("数据库密码为："+dbPassword);

//进行密码的比对

return this.equals(Strsh, dbPassword);

}

}

# **4，登录控制器**

@RequestMapping("/loginUser")

public String loginUser(String username,String password,HttpSession session) {

UsernamePasswordToken usernamePasswordToken=new UsernamePasswordToken(username,password);

Subject subject = SecurityUtils.getSubject();

Map map=new HashMap();

try {

subject.login(usernamePasswordToken); //完成登录

User user=(User) subject.getPrincipal();

session.setAttribute("user", user);

return "index";

} catch (IncorrectCredentialsException e) {

map.put("msg", "密码错误");

} catch (LockedAccountException e) {

map.put("msg", "登录失败，该用户已被冻结");

} catch (AuthenticationException e) {

map.put("msg", "该用户不存在");

} catch (Exception e) {

return "login";//返回登录页面

}

return map.toString();

}

# 5，thymeleaf页面权限控制

<html lang="zh\_CN" xmlns:th="http://www.thymeleaf.org"

xmlns:shiro="http://www.pollix.at/thymeleaf/shiro">

//作为属性控制

<button type="button" shiro:authenticated="true" class="btn btn-outline btn-default">

<i class="glyphicon glyphicon-plus" aria-hidden="true"></i>

</button>

//作为标签

<shiro:hasRole name="admin">

<button type="button" class="btn btn-outline btn-default">

<i class="glyphicon glyphicon-heart" aria-hidden="true"></i>

</button>

</shiro:hasRole>

# 6,标签说明

guest标签

　　<shiro:guest>

　　</shiro:guest>

***用户***没有身份验证时显示相应信息，即游客访问信息。

user标签

　　<shiro:user>

　　</shiro:user>

　　用户已经身份验证/记住我登录后显示相应的信息。

authenticated标签

　　<shiro:authenticated>

　　</shiro:authenticated>

　　用户已经身份验证通过，即Subject.login登录成功，不是记住我登录的。

notAuthenticated标签

　　<shiro:notAuthenticated>

　　</shiro:notAuthenticated>

　　用户已经身份验证通过，即没有调用Subject.login进行登录，包括记住我自动登录的也属于未进行身份验证。

principal标签

　　<shiro: principal/>

　　<shiro:principal property="username"/>

　　相当于((User)Subject.getPrincipals()).getUsername()。

lacksPermission标签

　　<shiro:lacksPermission name="org:create">

　　</shiro:lacksPermission>

　　如果当前Subject没有权限将显示body体内容。

hasRole标签

　　<shiro:hasRole name="admin">

　　</shiro:hasRole>

　　如果当前Subject有角色将显示body体内容。

hasAnyRoles标签

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

　　</shiro:hasAnyRoles>

　　如果当前Subject有任意一个角色（或的关系）将显示body体内容。

lacksRole标签

　　<shiro:lacksRole name="abc">

　　</shiro:lacksRole>

　　如果当前Subject没有角色将显示body体内容。

hasPermission标签

　　<shiro:hasPermission name="user:create">

　　</shiro:hasPermission>

如果当前Subject有权限将显示body体内容

参考地址

<https://blog.csdn.net/u012796085/article/details/79900936>

7,引入thymeleaf-extras-shiro

在pom中引入：

<dependency>

<groupId>com.github.theborakompanioni</groupId>

<artifactId>thymeleaf-extras-shiro</artifactId>

<version>2.0.0</version>

</dependency>

ShiroConfig配置

引入依赖后，需要在ShiroConfig中配置该方言标签：

@Bean

public ShiroDialect shiroDialect() {

return new ShiroDialect();

}

首页改造

更改index.html，用于测试Shiro标签的使用：

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org"

xmlns:shiro="http://www.pollix.at/thymeleaf/shiro" >

<head>

<meta charset="UTF-8">

<title>首页</title>

</head>

<body>

<p>你好！[[${user.userName}]]</p>

<p shiro:hasRole="admin">你的角色为超级管理员</p>

<p shiro:hasRole="test">你的角色为测试账户</p>

<div>

<a shiro:hasPermission="user:user" th:href="@{/user/list}">获取用户信息</a>

<a shiro:hasPermission="user:add" th:href="@{/user/add}">新增用户</a>

<a shiro:hasPermission="user:delete" th:href="@{/user/delete}">删除用户</a>

</div>

<a th:href="@{/logout}">注销</a>

</body>

</html>

值得注意的是，在html页面中使用Shiro标签需要给html标签添加xmlns:shiro="http://www.pollix.at/thymeleaf/shiro"。

测试

启动项目，使用mrbird（角色为admin，具有user:user，user:add，user:delete权限）账户登录：

使用tester（角色为tester，仅有user:user权限）账户登录：

————————————————

版权声明：本文为CSDN博主「笙箫123」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：<https://blog.csdn.net/q15102780705/article/details/107445247>

8，shiro权限注解

