**自定义Realm**

package com.czdp.daping.shiro.realm;

import com.baomidou.mybatisplus.core.toolkit.Wrappers;

import com.czdp.daping.entity.GUser;

import com.czdp.daping.service.IGUserService;

import lombok.extern.slf4j.Slf4j;

import org.apache.shiro.authc.\*;

import org.apache.shiro.realm.AuthenticatingRealm;

import org.springframework.beans.factory.annotation.Autowired;

import java.util.Optional;

/\*\*

\* @program: czdp

\* @description:

\* @author: wangguanzhong

\* @create: 2019/09/15 18:45

\*/

@Slf4j

public class CustomRealm extends AuthenticatingRealm {

@Autowired

IGUserService userService;

@Override

protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token) throws AuthenticationException {

String username = (String) token.getPrincipal();

GUser user= userService.getOne(Wrappers.<GUser>lambdaQuery().eq(GUser::getUsername,username));

Optional.ofNullable(user).orElseThrow(()->new UnknownAccountException());

if(user.getLocked()==true){

throw new LockedAccountException();

}

SimpleAuthenticationInfo authenticationInfo = new SimpleAuthenticationInfo(username,user.getPassword(),getName());

return authenticationInfo;

}

}

========================================

**shrioConfig**

package com.czdp.daping.shiro.config;

import com.czdp.daping.shiro.filter.FrontAuthenticationFilter;

import com.czdp.daping.shiro.filter.KickoutSessionControlFilter;

import com.czdp.daping.shiro.filter.WsIdCheckFilter;

import com.czdp.daping.shiro.realm.CustomRealm;

import org.apache.shiro.authc.credential.CredentialsMatcher;

import org.apache.shiro.cache.ehcache.EhCacheManager;

import org.apache.shiro.crypto.hash.SimpleHash;

import org.apache.shiro.mgt.SecurityManager;

import org.apache.shiro.session.mgt.DefaultSessionManager;

import org.apache.shiro.session.mgt.SessionManager;

import org.apache.shiro.spring.web.ShiroFilterFactoryBean;

import org.apache.shiro.web.mgt.DefaultWebSecurityManager;

import org.apache.shiro.web.session.mgt.DefaultWebSessionManager;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.boot.web.servlet.FilterRegistrationBean;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import javax.servlet.Filter;

import java.util.LinkedHashMap;

import java.util.Map;

/\*\*

\* @program: czdp

\* @description:

\* @author: wangguanzhong

\* @create: 2019/09/15 18:09

\*/

@Configuration

public class ShiroConfig {

@Bean

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

ShiroFilterFactoryBean shiroFilterFactoryBean = new ShiroFilterFactoryBean();

shiroFilterFactoryBean.setSecurityManager(securityManager);

shiroFilterFactoryBean.setLoginUrl("/login");

shiroFilterFactoryBean.setSuccessUrl("/loginOk");

Map<String, Filter> filterChain = new LinkedHashMap<>();

filterChain.put("frontauth",frontAuthenticationFilter());

filterChain.put("kickout",kickoutSessionControlFilter());

filterChain.put("wscheck",wsIdCheckFilter());

//filterChain.put("frontauth",new FrontAuthenticationFilter());

//filterChain.put("kickout",new KickoutSessionControlFilter());

shiroFilterFactoryBean.setFilters(filterChain);

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

filterChainDefinitionMap.put("/ws/dapingServer", "wscheck");

filterChainDefinitionMap.put("/swagger-ui.html", "anon");

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

filterChainDefinitionMap.put("/v2/\*\*", "anon");

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

filterChainDefinitionMap.put("/favicon.ico", "anon");

filterChainDefinitionMap.put("/csrf", "anon");

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

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

//2021-2-23新添加开放登录接口

filterChainDefinitionMap.put("/login","anon");

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

filterChainDefinitionMap.put("/\*\*","frontauth,kickout");

shiroFilterFactoryBean.setFilterChainDefinitionMap(filterChainDefinitionMap);

return shiroFilterFactoryBean;

}

// @Bean

// public SecurityManager securityManager(SessionManager sessionManager){

// DefaultWebSecurityManager defaultWebSecurityManager = new DefaultWebSecurityManager();

// defaultWebSecurityManager.setRealm(customRealm());

// defaultWebSecurityManager.setSessionManager(sessionManager);

// return defaultWebSecurityManager;

// }

@Bean

public SessionManager sessionManager(){

DefaultWebSessionManager sessionManager = new DefaultWebSessionManager();

//设置session永不过期

sessionManager.setGlobalSessionTimeout(-1);

return sessionManager;

}

@Bean

public CustomRealm customRealm(){

return new CustomRealm();

}

@Bean

public KickoutSessionControlFilter kickoutSessionControlFilter(){

return new KickoutSessionControlFilter();

}

@Bean

public FrontAuthenticationFilter frontAuthenticationFilter(){

return new FrontAuthenticationFilter();

}

@Bean

public FilterRegistrationBean regKickoutSessionControlFilter(KickoutSessionControlFilter filter) {

FilterRegistrationBean registration = new FilterRegistrationBean(filter);

registration.setEnabled(false);

return registration;

}

@Bean

public FilterRegistrationBean regFrontAuthenticationFilter(FrontAuthenticationFilter filter) {

FilterRegistrationBean registration = new FilterRegistrationBean(filter);

registration.setEnabled(false);

return registration;

}

@Bean

public WsIdCheckFilter wsIdCheckFilter(){

return new WsIdCheckFilter();

}

@Bean

public FilterRegistrationBean regWsIdCheckFilter(WsIdCheckFilter filter){

FilterRegistrationBean registration = new FilterRegistrationBean(filter);

registration.setEnabled(false);

return registration;

}

/\*\*

\* 创建defaultwebsecuritymanager

\*

\*/

@Bean(name="defaultWebSecurityManager")

public DefaultWebSecurityManager defaultWebSecurityManager(@Qualifier("userRealm")CustomRealm userRealm,SessionManager sessionManager) {

DefaultWebSecurityManager securityManager = new DefaultWebSecurityManager();

securityManager.setRealm(userRealm);

securityManager.setSessionManager(sessionManager);

return securityManager;

}

/\*\*

\* 创建realm

\*

\*/

@Bean(name="userRealm")

public CustomRealm userRealm() {

CustomRealm userRealm=new CustomRealm();

userRealm.setCredentialsMatcher(this.retryLimitCredentialsMatcher());

return userRealm;

}

//设置密码重试次数

@Bean

public EhCacheManager ehCacheManager() {

EhCacheManager ehCacheManager = new EhCacheManager();

ehCacheManager.setCacheManagerConfigFile("classpath:ehcache-shiro.xml");

return ehCacheManager;

}

//设置密码重试次数

@Bean

public CredentialsMatcher retryLimitCredentialsMatcher() {

RetryLimitCredentialsMatcher retryLimitCredentialsMatcher = new RetryLimitCredentialsMatcher(this.ehCacheManager());

//retryLimitCredentialsMatcher.setHashAlgorithmName("MD5");

//retryLimitCredentialsMatcher.setHashIterations(MD5Util.HASH\_ITERATIONS);

//retryLimitCredentialsMatcher.setStoredCredentialsHexEncoded(true);

return retryLimitCredentialsMatcher;

}

}

=====================

**RetryLimitCredentialsMatcher**

package com.czdp.daping.shiro.config;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;

import com.baomidou.mybatisplus.core.toolkit.Wrappers;

import com.czdp.daping.entity.GUser;

import com.czdp.daping.mapper.GUserMapper;

import com.czdp.daping.service.IGUserService;

import org.apache.shiro.authc.AuthenticationInfo;

import org.apache.shiro.authc.AuthenticationToken;

import org.apache.shiro.authc.ExcessiveAttemptsException;

import org.apache.shiro.authc.UsernamePasswordToken;

import org.apache.shiro.authc.credential.HashedCredentialsMatcher;

import org.apache.shiro.cache.Cache;

import org.apache.shiro.cache.ehcache.EhCacheManager;

import org.apache.shiro.subject.PrincipalCollection;

import org.slf4j.Logger;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Component;

import javax.annotation.Resource;

import java.util.concurrent.atomic.AtomicInteger;

import static org.slf4j.LoggerFactory.getLogger;

public class RetryLimitCredentialsMatcher extends HashedCredentialsMatcher {

/\*\*

\* 集群中可能会导致出现验证多过5次的现象，因为AtomicInteger只能保证单节点并发

\*/

private static final int MAX\_LOGIN\_RETRY\_TIMES = 5;

private static final Logger LOGGER = getLogger(RetryLimitCredentialsMatcher.class);

private Cache<String, AtomicInteger> passwordRetryCache;

public Cache<String, AtomicInteger> getPasswordRetryCache() {

return passwordRetryCache;

}

public RetryLimitCredentialsMatcher(EhCacheManager ehCacheManager) {

passwordRetryCache = ehCacheManager.getCache("passwordRetryCache");

}

@Resource

GUserMapper gUserMapper;

@Override

public boolean doCredentialsMatch(AuthenticationToken token, AuthenticationInfo info) throws ExcessiveAttemptsException {

String userName = (String) token.getPrincipal();

//测试

UsernamePasswordToken upt = (UsernamePasswordToken) token;

char[] password = upt.getPassword();

System.out.print("缓存中的密码:");

for (char c : password) {

System.out.print(c);

}

System.out.println();

AtomicInteger retryCount = passwordRetryCache.get(userName);

System.out.println("密码重试次数:" + retryCount);

if (retryCount == null) {

// 高并发下使用的线程安全的int类

retryCount = new AtomicInteger(0);

passwordRetryCache.put(userName, retryCount);

}

if (retryCount.incrementAndGet() > MAX\_LOGIN\_RETRY\_TIMES) {

// throw new ExcessiveAttemptsException();

LOGGER.error("登录次数超过限制");

throw new ExcessiveAttemptsException("用户:" + userName + "登录次数已超过限制，账户已进入自动锁定，请您一小时解锁后再尝试登录！");

}

//并将其保存到缓存中

passwordRetryCache.put(userName, retryCount);

//debug

if (LOGGER.isDebugEnabled()) {

LOGGER.debug("用户:{},尝试登录次数:{}", userName, retryCount.get());

}

String password1 = String.valueOf(password);//传入密码

String password2 = info.getCredentials().toString();//数据库密码

//String password2 = gUserMapper.selectOne(Wrappers.<GUser>lambdaQuery().eq(GUser::getUsername, username2)).getPassword();//数据库密码

boolean match = false;

if (password1.equals(password2)) {

match = true;

}

if (match) {

passwordRetryCache.remove(userName);

} else {

throw new ExcessiveAttemptsException("用户名或密码错误,剩余" + (MAX\_LOGIN\_RETRY\_TIMES - retryCount.get()) + "次输入机会");

}

return match;

}

}

=====================================

**ehcache-shiro.xml**

<?xml version="1.0" encoding="UTF-8"?>

<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="http://ehcache.org/ehcache.xsd"

updateCheck="false"

dynamicConfig="true"

name="ehcache\_retry">

<!-- 磁盘缓存位置 -->

<diskStore path="java.io.tmpdir/ehcache" />

<!-- 默认缓存 -->

<defaultCache maxEntriesLocalHeap="10000"

eternal="false"

timeToIdleSeconds="120"

timeToLiveSeconds="120"

maxEntriesLocalDisk="10000000"

diskExpiryThreadIntervalSeconds="120"

memoryStoreEvictionPolicy="LRU">

<persistence strategy="localTempSwap" />

</defaultCache>

<!-- helloworld缓存 -->

<cache name="passwordRetryCache"

maxElementsInMemory="2000"

eternal="false"

timeToIdleSeconds="600"

timeToLiveSeconds="600"

overflowToDisk="false"

statistics="true" />

</ehcache>

====================================

**登录主入口**

package com.czdp.daping.shiro.controller;

import com.czdp.daping.constant.BizConstant;

import com.czdp.daping.shiro.realm.CustomRealm;

import com.czdp.daping.shiro.util.ShiroUtil;

import com.czdp.daping.websocket.extension.CityClientCo;

import com.czdp.daping.websocket.handler.ServiceHandler;

import com.gykj.core.common.Result;

import com.gykj.core.enums.ErrCodeEnum;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import lombok.extern.slf4j.Slf4j;

import org.apache.shiro.SecurityUtils;

import org.apache.shiro.authc.\*;

import org.apache.shiro.cache.Cache;

import org.apache.shiro.cache.ehcache.EhCacheManager;

import org.apache.shiro.realm.AuthenticatingRealm;

import org.apache.shiro.session.Session;

import org.apache.shiro.subject.Subject;

import org.apache.shiro.web.filter.authc.FormAuthenticationFilter;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import javax.servlet.http.HttpServletRequest;

import java.io.IOException;

import java.io.Serializable;

import java.util.concurrent.atomic.AtomicInteger;

/\*\*

\* @program: czdp

\* @description:

\* @author: wangguanzhong

\* @create: 2019/09/15 19:03

\*/

@RestController

@Api(value = "登录操作")

@Slf4j

public class LoginController {

@Autowired

ServiceHandler serviceHandler;

@Autowired

CustomRealm customRealm;

@PostMapping(value = "/login")

@ApiOperation(value = "登录操作")

public Result login(HttpServletRequest request,

@RequestParam("username") String username,

@RequestParam("password") String password) {

Subject subject = SecurityUtils.getSubject();

UsernamePasswordToken token = new UsernamePasswordToken(username, password);

ErrCodeEnum err;

try {

subject.login(token);

err = ErrCodeEnum.GL90000400;

//给前端返回sessionid

String sessionid = ShiroUtil.getSessionId();

log.error("===login success。sessionId: {} ", sessionid);

//登录进来默认城市设置为江苏

CityClientCo.add(sessionid, BizConstant.JIANGSU\_CODE);

return Result.ok(ErrCodeEnum.GL90000400, sessionid);

} catch (UnknownAccountException e) {

// 用户名不存在

err = ErrCodeEnum.GL90000401;

} catch (IncorrectCredentialsException e) {

// TODO: handle exception

err = ErrCodeEnum.GL90000401;

} catch (ExcessiveAttemptsException e) {

err = ErrCodeEnum.RESTRICT;

err.setMessage(e.getMessage());

}

// String exceptionClassName =

// (String) request.getAttribute(FormAuthenticationFilter.DEFAULT\_ERROR\_KEY\_ATTRIBUTE\_NAME);

// ErrCodeEnum err ;

// if (UnknownAccountException.class.getName().equals(exceptionClassName)) {

// err = ErrCodeEnum.GL90000401;

// } else if (IncorrectCredentialsException.class.getName().equals(exceptionClassName)) {

// err = ErrCodeEnum.GL90000401;

// }else if (ExcessiveAttemptsException.class.getName().equals(exceptionClassName)){

// err =ErrCodeEnum.RESTRICT;

// }

// else if (exceptionClassName != null) {

// err = ErrCodeEnum.GL90000402;

// } else {

// //err = ErrCodeEnum.GL90000400;

// //给前端返回sessionid

//

// String sessionid = ShiroUtil.getSessionId();

// log.error("===login success。sessionId: {} ",sessionid);

// //登录进来默认城市设置为江苏

// CityClientCo.add(sessionid,BizConstant.JIANGSU\_CODE);

//

// return Result.ok(ErrCodeEnum.GL90000400,sessionid);

// }

return Result.ok(err);

}

@GetMapping(value = "/login")

//@ApiOperation(value = "登录页面")

public Result loginShow() {

return Result.ok();

}

/\*@GetMapping(value = "/loginOk")

@ApiOperation(value = "登出成功")

public Result loginOk() {

return Result.ok();

}\*/

@PostMapping(value = "/logout")

@ApiOperation(value = "登出操作")

public Result logOut() {

Subject subject = SecurityUtils.getSubject();

subject.logout();

Session session = subject.getSession();

Serializable sessionId = session.getId();

log.error("===login out。sessionId: {} ", sessionId);

try {

serviceHandler.closeWsSocket((String) sessionId);

} catch (IOException e) {

e.printStackTrace();

return Result.ok(ErrCodeEnum.GL90000416);

}

return Result.ok();

}

@GetMapping(value = "/")

//@ApiOperation(value = "默认页面")

public Result logDefault() {

return Result.ok();

}

}