链滴

求助！！！SpringSecurity 配置单点登录问题

作者：ronger

原文链接：https://ld246.com/article/1498358324601

来源网站：链滴

问题：不添加权限控制，页面可以正常访问，添加后页面无法正常访问（未进控制层），权限认证正常

代码如下：

1. WebSecurityConfig.java

```
package com.player.config;

import com.player.handler.AuthenticationProviderCustom;
import com.player.handler.UserDetailsServiceCustom;
import com.player.repository.AuthorRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationProvider;
import org.springframework.security.config.annotation.authentication.builders.AuthenticatioManagerBuilder;
import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigrerAdapter;
import org.springframework.security.core.userdetails.UserDetailsService;

/**
 * Created by ronger on 2017/6/24. */
@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter{

    @Autowired
  private AuthorRepository authorRepository;

  @Bean
  public UserDetailsService userDetailsService(){
        UserDetailsService userDetailsService=new UserDetailsServiceCustom(authorRepository);
 return userDetailsService;
  }

    @Bean
  public AuthenticationProvider authenticationProvider(){
        AuthenticationProvider authenticationProvider = new AuthenticationProviderCustom(usrDetailsService());
 return authenticationProvider;
  }

    /**
 * 匹配 "/" 路径，不需要权限即可访问
  * 匹配 "/user" 及其以下所有路径，都需要 "USER" 权限
```

```
 * 登录地址为 "/login"，登录成功默认跳转到页面 "/user"
 * 退出登录的地址为 "/logout"，退出成功后跳转到页面 "/login"
 * 默认启用 CSRF
 */  @Override
  protected void configure(HttpSecurity http) throws Exception {
        http
                .authorizeRequests()
                .antMatchers("/").permitAll()
                .antMatchers("/user/**").hasRole("USER")
                .antMatchers("/upload/**").hasRole("USER")
                .antMatchers("/admin/**").hasRole("ADMIN")
                .and()
                .formLogin().loginPage("/login").defaultSuccessUrl("/index").failureUrl("/login?error")
usernameParameter("username").passwordParameter("password").permitAll()
                .and()
                .logout().logoutUrl("/logout").logoutSuccessUrl("/login");
  }

    @Override
  public void configure(WebSecurity web) throws Exception {
        web.ignoring().antMatchers("/static/**");
  }

    /**
 * 添加 UserDetailsService， 实现自定义登录校验
 */
  @Override
  protected void configure(AuthenticationManagerBuilder builder) throws Exception{
        //暂时使用基于内存的AuthenticationProvider
 //builder.inMemoryAuthentication().withUser("username").password("password").roles("USER
); //自定义AuthenticationProvider
  builder.authenticationProvider(authenticationProvider());
  }

}
```

2. UserDetailsServiceCustom.java

```
package com.player.handler;

import com.player.repository.AuthorRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;

/**
 * Created by ronger on 2017/6/24. */
 public class UserDetailsServiceCustom implements UserDetailsService {

    @Autowired
  private AuthorRepository authorRepository;
```

```
 public UserDetailsServiceCustom(AuthorRepository authorRepository) {
        this.authorRepository = authorRepository;
  }

    @Override
  public UserDetails loadUserByUsername(String username) throws UsernameNotFoundExcept
on {
        return authorRepository.findByUsernameWithAuthorities(username);
  }
}
```

3. AuthenticationProviderCustom.java

```
package com.player.handler;

import com.player.util.MD5Tools;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.*;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;

/**
 * Created by ronger on 2017/6/24. */
 public class AuthenticationProviderCustom implements AuthenticationProvider {

    @Autowired
  private UserDetailsService userDetailsService;

 public AuthenticationProviderCustom(UserDetailsService userDetailsService) {
        this.userDetailsService = userDetailsService;
  }

    @Override
  public Authentication authenticate(Authentication authentication) throws AuthenticationExc
ption {
        try{
            UsernamePasswordAuthenticationToken token = (UsernamePasswordAuthenticationT
ken) authentication;
  String account = token.getName();
  //从数据库找到的用户
  UserDetails userDetails = null;
 if(account != null) {
              userDetails = userDetailsService.loadUserByUsername(account);
  }
          //
  if(userDetails == null) {
              throw new UsernameNotFoundException("用户名/密码无效");
  }else if (!userDetails.isEnabled()){
              throw new DisabledException("用户已被禁用");
  }else if (!userDetails.isAccountNonExpired()) {
```

```
                throw new AccountExpiredException("账号已过期");
  }else if (!userDetails.isAccountNonLocked()) {
                throw new LockedException("账号已被锁定");
  }else if (!userDetails.isCredentialsNonExpired()) {
                throw new LockedException("凭证已过期");
  }
          //数据库用户的密码
  String password = userDetails.getPassword();
  //与authentication里面的credentials相比较
  if(!password.equals(MD5Tools.MD5(token.getCredentials().toString()))) {
                throw new BadCredentialsException("Invalid username/password");
  }
          //授权
  return new UsernamePasswordAuthenticationToken(userDetails, password,userDetails.getAu
thorities());
  }catch (Exception e){
          e.printStackTrace();
  }
       return null;
  }

    @Override
  public boolean supports(Class authentication) {
       //返回true后才会执行上面的authenticate方法,这步能确保authentication能正确转换类型
  return UsernamePasswordAuthenticationToken.class.equals(authentication);
  }

}
```