mall整合SpringSecurity和JWT实现认证和授权（一）

原创 梦想de星空 macrozheng 2019-05-13 12:00

收录于合集
#mall学习教程（架构篇）
10个

> 本文主要讲解mall通过整合SpringSecurity和JWT实现后台用户的登录和授权功能，同时改造Swagger-UI的配置使其可以自动记住登录令牌进行发送。

项目使用框架介绍

SpringSecurity

> SpringSecurity是一个强大的可高度定制的认证和授权框架，对于Spring应用来说它是一套Web安全标准。SpringSecurity注重于为Java应用提供认证和授权功能，像所有的Spring项目一样，它对自定义需求具有强大的扩展性。

JWT

> JWT是JSON WEB TOKEN的缩写，它是基于 RFC 7519 标准定义的一种可以安全传输的的JSON对象，由于使用了数字签名，所以是可信任和安全的。

JWT的组成

- JWT token的格式：header.payload.signature
- header中用于存放签名的生成算法

```
{"alg": "HS512"}
```

- payload中用于存放用户名、token的生成时间和过期时间

```
{"sub":"admin","created":1489079981393,"exp":1489684781}
```

- signature为以header和payload生成的签名，一旦header和payload被篡改，验证将失败

```
//secret为加密算法的密钥
String signature = HMACSHA512(base64UrlEncode(header) + "." +base64UrlEncode(payload),se
```

JWT实例

这是一个JWT的字符串

```
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbiIsImNyZWF0ZWQiOjE1NTY3NzkxMjUzMDksImV4cCI6MTU1NzM
```

可以在该网站上获得解析结果：https://jwt.io/

JWT实现认证和授权的原理

- 用户调用登录接口，登录成功后获取到JWT的token；
- 之后用户每次调用接口都在http的header中添加一个叫Authorization的头，值为JWT的token；
- 后台程序通过对Authorization头中信息的解码及数字签名校验来获取其中的用户信息，从而实现认证和授权。

Hutool

> Hutool是一个丰富的Java开源工具包,它帮助我们简化每一行代码，减少每一个方法，mall项目采用了此工具包。

项目使用表说明

- `ums_admin`：后台用户表
- `ums_role`：后台用户角色表

- ums_permission：后台用户权限表
- ums_admin_role_relation：后台用户和角色关系表，用户与角色是多对多关系
- ums_role_permission_relation：后台用户角色和权限关系表，角色与权限是多对多关系
- ums_admin_permission_relation：后台用户和权限关系表(除角色中定义的权限以外的加减权限)，加权限是指用户比角色多出的权限，减权限是指用户比角色少的权限

整合SpringSecurity及JWT

在pom.xml中添加项目依赖

```
<!--SpringSecurity依赖配置-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<!--Hutool Java工具包-->
<dependency>
    <groupId>cn.hutool</groupId>
    <artifactId>hutool-all</artifactId>
    <version>4.5.7</version>
</dependency>
<!--JWT(Json Web Token)登录支持-->
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.0</version>
</dependency>
```

添加JWT token的工具类

> 用于生成和解析JWT token的工具类

相关方法说明：

- generateToken(UserDetails userDetails) :用于根据登录用户信息生成token
- getUserNameFromToken(String token)：从token中获取登录用户的信息
- validateToken(String token, UserDetails userDetails)：判断token是否还有效

```
package com.macro.mall.tiny.common.utils;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Component;

import java.util.Date;
import java.util.HashMap;
import java.util.Map;

/**
 * JwtToken生成的工具类
 * Created by macro on 2018/4/26.
 */
@Component
public class JwtTokenUtil {
    private static final Logger LOGGER = LoggerFactory.getLogger(JwtTokenUtil.class);
    private static final String CLAIM_KEY_USERNAME = "sub";
    private static final String CLAIM_KEY_CREATED = "created";
    @Value("${jwt.secret}")
    private String secret;
    @Value("${jwt.expiration}")
    private Long expiration;

    /**
     * 根据负责生成JWT的token
     */
    private String generateToken(Map<String, Object> claims) {
        return Jwts.builder()
                .setClaims(claims)
                .setExpiration(generateExpirationDate())
                .signWith(SignatureAlgorithm.HS512, secret)
                .compact();
    }

    /**
     * 从token中获取JWT中的负载
     */
    private Claims getClaimsFromToken(String token) {
        Claims claims = null;
        try {
            claims = Jwts.parser()
                    .setSigningKey(secret)
                    .parseClaimsJws(token)
                    .getBody();
        } catch (Exception e) {
            LOGGER.info("JWT格式验证失败:{}",token);
        }
        return claims;
    }

    /**
     * 生成token的过期时间
     */
    private Date generateExpirationDate() {
        return new Date(System.currentTimeMillis() + expiration * 1000);
```

```
    }

    /**
     * 从token中获取登录用户名
     */
    public String getUserNameFromToken(String token) {
        String username;
        try {
            Claims claims = getClaimsFromToken(token);
            username =  claims.getSubject();
        } catch (Exception e) {
            username = null;
        }
        return username;
    }

    /**
     * 验证token是否还有效
     *
     * @param token       客户端传入的token
     * @param userDetails 从数据库中查询出来的用户信息
     */
    public boolean validateToken(String token, UserDetails userDetails) {
        String username = getUserNameFromToken(token);
        return username.equals(userDetails.getUsername()) && !isTokenExpired(token);
    }

    /**
     * 判断token是否已经失效
     */
    private boolean isTokenExpired(String token) {
        Date expiredDate = getExpiredDateFromToken(token);
        return expiredDate.before(new Date());
    }

    /**
     * 从token中获取过期时间
     */
    private Date getExpiredDateFromToken(String token) {
        Claims claims = getClaimsFromToken(token);
        return claims.getExpiration();
    }

    /**
     * 根据用户信息生成token
     */
    public String generateToken(UserDetails userDetails) {
        Map<String, Object> claims = new HashMap<>();
        claims.put(CLAIM_KEY_USERNAME, userDetails.getUsername());
        claims.put(CLAIM_KEY_CREATED, new Date());
        return generateToken(claims);
    }

    /**
```

```
     * 判断token是否可以被刷新
     */
    public boolean canRefresh(String token) {
        return !isTokenExpired(token);
    }

    /**
     * 刷新token
     */
    public String refreshToken(String token) {
        Claims claims = getClaimsFromToken(token);
        claims.put(CLAIM_KEY_CREATED, new Date());
        return generateToken(claims);
    }
}
```

添加SpringSecurity的配置类

```
package com.macro.mall.tiny.config;

import com.macro.mall.tiny.component.JwtAuthenticationTokenFilter;
import com.macro.mall.tiny.component.RestAuthenticationEntryPoint;
import com.macro.mall.tiny.component.RestfulAccessDeniedHandler;
import com.macro.mall.tiny.dto.AdminUserDetails;
import com.macro.mall.tiny.mbg.model.UmsAdmin;
import com.macro.mall.tiny.mbg.model.UmsPermission;
import com.macro.mall.tiny.service.UmsAdminService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.config.annotation.authentication.builders.Authenti
import org.springframework.security.config.annotation.method.configuration.EnableGloba
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecur
import org.springframework.security.config.annotation.web.configuration.WebSecurityCon
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationF

import java.util.List;

/**
 * SpringSecurity的配置
 * Created by macro on 2018/4/26.
 */
@Configuration
@EnableWebSecurity
```

```
@EnableGlobalMethodSecurity(prePostEnabled=true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    private UmsAdminService adminService;
    @Autowired
    private RestfulAccessDeniedHandler restfulAccessDeniedHandler;
    @Autowired
    private RestAuthenticationEntryPoint restAuthenticationEntryPoint;

    @Override
    protected void configure(HttpSecurity httpSecurity) throws Exception {
        httpSecurity.csrf()// 由于使用的是JWT，我们这里不需要csrf
                .disable()
                .sessionManagement()// 基于token，所以不需要session
                .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                .and()
                .authorizeRequests()
                .antMatchers(HttpMethod.GET, // 允许对于网站静态资源的无授权访问
                        "/",
                        "/*.html",
                        "/favicon.ico",
                        "/**/*.html",
                        "/**/*.css",
                        "/**/*.js",
                        "/swagger-resources/**",
                        "/v2/api-docs/**"
                )
                .permitAll()
                .antMatchers("/admin/login", "/admin/register")// 对登录注册要允许匿名访
                .permitAll()
                .antMatchers(HttpMethod.OPTIONS)//跨域请求会先进行一次options请求
                .permitAll()
//                .antMatchers("/**")//测试时全部运行访问
//                .permitAll()
                .anyRequest()// 除上面外的所有请求全部需要鉴权认证
                .authenticated();
        // 禁用缓存
        httpSecurity.headers().cacheControl();
        // 添加JWT filter
        httpSecurity.addFilterBefore(jwtAuthenticationTokenFilter(), UsernamePasswordAu
        //添加自定义未授权和未登录结果返回
        httpSecurity.exceptionHandling()
                .accessDeniedHandler(restfulAccessDeniedHandler)
                .authenticationEntryPoint(restAuthenticationEntryPoint);

    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService())
                .passwordEncoder(passwordEncoder());
    }

    @Bean
```

```
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public UserDetailsService userDetailsService() {
        //获取登录用户信息
        return username -> {
            UmsAdmin admin = adminService.getAdminByUsername(username);
            if (admin != null) {
                List<UmsPermission> permissionList = adminService.getPermissionList(adı
                return new AdminUserDetails(admin,permissionList);
            }
            throw new UsernameNotFoundException("用户名或密码错误");
        };
    }

    @Bean
    public JwtAuthenticationTokenFilter jwtAuthenticationTokenFilter(){
        return new JwtAuthenticationTokenFilter();
    }

    @Bean
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }

}
```

相关依赖及方法说明

- configure(HttpSecurity httpSecurity)：用于配置需要拦截的url路径、jwt过滤器及出异常后的处理器；
- configure(AuthenticationManagerBuilder auth)：用于配置UserDetailsService及PasswordEncoder；
- RestfulAccessDeniedHandler：当用户没有访问权限时的处理器，用于返回JSON格式的处理结果；
- RestAuthenticationEntryPoint：当未登录或token失效时，返回JSON格式的结果；
- UserDetailsService:SpringSecurity定义的核心接口，用于根据用户名获取用户信息，需要自行实现；
- UserDetails：SpringSecurity定义用于封装用户信息的类（主要是用户信息和权限），需要自行实现；
- PasswordEncoder：SpringSecurity定义的用于对密码进行编码及比对的接口，目前使用的是BCryptPasswordEncoder；

- JwtAuthenticationTokenFilter：在用户名和密码校验前添加的过滤器，如果有jwt的token，会自行根据token信息进行登录。

添加RestfulAccessDeniedHandler

```
package com.macro.mall.tiny.component;

import cn.hutool.json.JSONUtil;
import com.macro.mall.tiny.common.api.CommonResult;
import org.springframework.security.access.AccessDeniedException;
import org.springframework.security.web.access.AccessDeniedHandler;
import org.springframework.stereotype.Component;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * 当访问接口没有权限时，自定义的返回结果
 * Created by macro on 2018/4/26.
 */
@Component
public class RestfulAccessDeniedHandler implements AccessDeniedHandler{
    @Override
    public void handle(HttpServletRequest request,
                       HttpServletResponse response,
                       AccessDeniedException e) throws IOException, ServletException {
        response.setCharacterEncoding("UTF-8");
        response.setContentType("application/json");
        response.getWriter().println(JSONUtil.parse(CommonResult.forbidden(e.getMessage(
        response.getWriter().flush();
    }
}
```

添加RestAuthenticationEntryPoint

```
package com.macro.mall.tiny.component;

import cn.hutool.json.JSONUtil;
import com.macro.mall.tiny.common.api.CommonResult;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.AuthenticationEntryPoint;
import org.springframework.stereotype.Component;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
```

```
 * 当未登录或者token失效访问接口时，自定义的返回结果
 * Created by macro on 2018/5/14.
 */
@Component
public class RestAuthenticationEntryPoint implements AuthenticationEntryPoint {
    @Override
    public void commence(HttpServletRequest request, HttpServletResponse response, Authe
        response.setCharacterEncoding("UTF-8");
        response.setContentType("application/json");
        response.getWriter().println(JSONUtil.parse(CommonResult.unauthorized(authExcept
        response.getWriter().flush();
    }
}
```

添加AdminUserDetails

```
package com.macro.mall.tiny.dto;

import com.macro.mall.tiny.mbg.model.UmsAdmin;
import com.macro.mall.tiny.mbg.model.UmsPermission;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.util.Collection;
import java.util.List;
import java.util.stream.Collectors;

/**
 * SpringSecurity需要的用户详情
 * Created by macro on 2018/4/26.
 */
public class AdminUserDetails implements UserDetails {
    private UmsAdmin umsAdmin;
    private List<UmsPermission> permissionList;
    public AdminUserDetails(UmsAdmin umsAdmin, List<UmsPermission> permissionList) {
        this.umsAdmin = umsAdmin;
        this.permissionList = permissionList;
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        //返回当前用户的权限
        return permissionList.stream()
                .filter(permission -> permission.getValue()!=null)
                .map(permission ->new SimpleGrantedAuthority(permission.getValue()))
                .collect(Collectors.toList());
    }

    @Override
    public String getPassword() {
```

```
        return umsAdmin.getPassword();
    }

    @Override
    public String getUsername() {
        return umsAdmin.getUsername();
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return umsAdmin.getStatus().equals(1);
    }
}
```

添加JwtAuthenticationTokenFilter

> 在用户名和密码校验前添加的过滤器，如果请求中有jwt的token且有效，会取出token中的用户名，然后调用SpringSecurity的API进行登录操作。

```
package com.macro.mall.tiny.component;

import com.macro.mall.tiny.common.utils.JwtTokenUtil;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * JWT登录授权过滤器
 * Created by macro on 2018/4/26.
 */
public class JwtAuthenticationTokenFilter extends OncePerRequestFilter {
    private static final Logger LOGGER = LoggerFactory.getLogger(JwtAuthenticationTokenF
    @Autowired
    private UserDetailsService userDetailsService;
    @Autowired
    private JwtTokenUtil jwtTokenUtil;
    @Value("${jwt.tokenHeader}")
    private String tokenHeader;
    @Value("${jwt.tokenHead}")
    private String tokenHead;

    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response,
                                    FilterChain chain) throws ServletException, IOExcept
        String authHeader = request.getHeader(this.tokenHeader);
        if (authHeader != null && authHeader.startsWith(this.tokenHead)) {
            String authToken = authHeader.substring(this.tokenHead.length());// The part
            String username = jwtTokenUtil.getUserNameFromToken(authToken);
            LOGGER.info("checking username:{}", username);
            if (username != null && SecurityContextHolder.getContext().getAuthentication
                UserDetails userDetails = this.userDetailsService.loadUserByUsername(use
                if (jwtTokenUtil.validateToken(authToken, userDetails)) {
                    UsernamePasswordAuthenticationToken authentication = new UsernamePas
                    authentication.setDetails(new WebAuthenticationDetailsSource().build
                    LOGGER.info("authenticated user:{}", username);
                    SecurityContextHolder.getContext().setAuthentication(authentication)
                }
            }
        }
        chain.doFilter(request, response);
    }
}
```

项目源码地址

https://github.com/macrozheng/mall-learning/tree/master/mall-tiny-04

推荐阅读

- mall架构及功能概览
- mall学习所需知识点（推荐资料）

- mall整合SpringBoot+MyBatis搭建基本骨架
- mall整合Swagger-UI实现在线API文档
- mall整合Redis实现缓存功能

欢迎关注，点个在看

收录于合集 #mall学习教程（架构篇） 10

上一篇
mall整合Redis实现缓存功能

下一篇
mall整合SpringSecurity和JWT实现认证和授权（二）

阅读原文

喜欢此内容的人还喜欢

项目中到底该不该用Lombok?
macrozheng