仅需四步，整合SpringSecurity+JWT实现登录认证！

原创 梦想de星空 macrozheng 2019-12-11 08:32

收录于合集
#mall学习教程（技术要点篇）
17个

> 学习过我的mall项目的应该知道， `mall-admin` 模块是使用SpringSecurity+JWT来实现登录认证的，而 `mall-portal` 模块是使用的SpringSecurity基于Session的默认机制来实现登陆认证的。很多小伙伴都找不到 `mall-portal` 的登录接口，最近我把这两个模块的登录认证给统一了，都使用SpringSecurity+JWT的形式实现。主要是通过把登录认证的通用逻辑抽取到了 `mall-security` 模块来实现的，下面我们讲讲如何使用 `mall-security` 模块来实现登录认证，仅需四步即可。

整合步骤

> 这里我们以 `mall-portal` 改造为例来说说如何实现。

- 第一步，给需要登录认证的模块添加 `mall-security` 依赖：

```
<dependency>
    <groupId>com.macro.mall</groupId>
    <artifactId>mall-security</artifactId>
</dependency>
```

- 第二步，添加MallSecurityConfig配置类，继承 `mall-security` 中的SecurityConfig配置，并且配置一个UserDetailsService接口的实现类，用于获取登录用户详情：

```
/**
 * mall-security模块相关配置
 * Created by macro on 2019/11/5.
 */
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled=true)
```

```
@EnableGlobalMethodSecurity(prePostEnabled=true)
publicclass MallSecurityConfig extends SecurityConfig {

    @Autowired
    private UmsMemberService memberService;

    @Bean
    public UserDetailsService userDetailsService() {
        //获取登录用户信息
        return username -> memberService.loadUserByUsername(username);
    }
}
```

- 第三步，在application.yml中配置下不需要安全保护的资源路径：

```
secure:
  ignored:
    urls:#安全路径白名单
      -/swagger-ui.html
      -/swagger-resources/**
      -/swagger/**
      -/**/v2/api-docs
      -/**/*.js
      -/**/*.css
      -/**/*.png
      -/**/*.ico
      -/webjars/springfox-swagger-ui/**
      -/druid/**
      -/actuator/**
      -/sso/**
      -/home/**
```

- 第四步，在UmsMemberController中实现登录和刷新token的接口：

```
/**
 * 会员登录注册管理Controller
 * Created by macro on 2018/8/3.
 */
@Controller
@Api(tags = "UmsMemberController", description = "会员登录注册管理")
@RequestMapping("/sso")
publicclass UmsMemberController {
```

```
    @Value("${jwt.tokenHeader}")
    private String tokenHeader;
    @Value("${jwt.tokenHead}")
    private String tokenHead;
    @Autowired
    private UmsMemberService memberService;

    @ApiOperation("会员登录")
    @RequestMapping(value = "/login", method = RequestMethod.POST)
    @ResponseBody
    public CommonResult login(@RequestParam String username,
                              @RequestParam String password) {
        String token = memberService.login(username, password);
        if (token == null) {
            return CommonResult.validateFailed("用户名或密码错误");
        }
        Map<String, String> tokenMap = new HashMap<>();
        tokenMap.put("token", token);
        tokenMap.put("tokenHead", tokenHead);
        return CommonResult.success(tokenMap);
    }

    @ApiOperation(value = "刷新token")
    @RequestMapping(value = "/refreshToken", method = RequestMethod.GET)
    @ResponseBody
    public CommonResult refreshToken(HttpServletRequest request) {
        String token = request.getHeader(tokenHeader);
        String refreshToken = memberService.refreshToken(token);
        if (refreshToken == null) {
            return CommonResult.failed("token已经过期！");
        }
        Map<String, String> tokenMap = new HashMap<>();
        tokenMap.put("token", refreshToken);
        tokenMap.put("tokenHead", tokenHead);
        return CommonResult.success(tokenMap);
    }
}
```

实现原理

> 将SpringSecurity+JWT的代码封装成通用模块后，就可以方便其他需要登录认证的模块来使用，下面我们来看看它是如何实现的，首先我们看下 `mall-security` 的目录结构。

目录结构

```
mall-security
├── component
|    ├── JwtAuthenticationTokenFilter -- JWT登录授权过滤器
|    ├── RestAuthenticationEntryPoint -- 自定义返回结果：未登录或登录过期
|    └── RestfulAccessDeniedHandler -- 自定义返回结果：没有权限访问时
├── config
|    ├── IgnoreUrlsConfig -- 用于配置不需要安全保护的资源路径
|    └── SecurityConfig -- SpringSecurity通用配置
└── util
     └── JwtTokenUtil -- JWT的token处理工具类
```

做了哪些变化

> 其实我也就添加了两个类，一个IgnoreUrlsConfig，用于从application.yml中获取不需要安全保护的资源路径。一个SecurityConfig提取了一些SpringSecurity的通用配置。

- IgnoreUrlsConfig中的代码：

```
/**
 * 用于配置不需要保护的资源路径
 * Created by macro on 2018/11/5.
 */
@Getter
@Setter
@ConfigurationProperties(prefix = "secure.ignored")
publicclass IgnoreUrlsConfig {

    private List<String> urls = new ArrayList<>();

}
```

- SecurityConfig中的代码：

```
/**
 * 对SpringSecurity的配置的扩展，支持自定义白名单资源路径和查询用户逻辑
 * Created by macro on 2019/11/5.
 */
publicclass SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity httpSecurity) throws Exception {
        ExpressionUrlAuthorizationConfigurer<HttpSecurity>.ExpressionInterceptUrlRegistry registry
                .authorizeRequests();
        //不需要保护的资源路径允许访问
        for (String url : ignoreUrlsConfig().getUrls()) {
            registry.antMatchers(url).permitAll();
        }
        //允许跨域请求的OPTIONS请求
        registry.antMatchers(HttpMethod.OPTIONS)
                .permitAll();
        // 任何请求需要身份认证
        registry.and()
                .authorizeRequests()
                .anyRequest()
                .authenticated()
                // 关闭跨站请求防护及不使用session
                .and()
                .csrf()
                .disable()
                .sessionManagement()
                .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                // 自定义权限拒绝处理类
                .and()
                .exceptionHandling()
                .accessDeniedHandler(restfulAccessDeniedHandler())
                .authenticationEntryPoint(restAuthenticationEntryPoint())
                // 自定义权限拦截器JWT过滤器
                .and()
                .addFilterBefore(jwtAuthenticationTokenFilter(), UsernamePasswordAuthenticationFi
    }

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

    @Bean
```

```
    public PasswordEncoder passwordEncoder() {
        returnnew BCryptPasswordEncoder();
    }

    @Bean
    public JwtAuthenticationTokenFilter jwtAuthenticationTokenFilter() {
        returnnew JwtAuthenticationTokenFilter();
    }

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

    @Bean
    public RestfulAccessDeniedHandler restfulAccessDeniedHandler() {
        returnnew RestfulAccessDeniedHandler();
    }

    @Bean
    public RestAuthenticationEntryPoint restAuthenticationEntryPoint() {
        returnnew RestAuthenticationEntryPoint();
    }

    @Bean
    public IgnoreUrlsConfig ignoreUrlsConfig() {
        returnnew IgnoreUrlsConfig();
    }

    @Bean
    public JwtTokenUtil jwtTokenUtil() {
        returnnew JwtTokenUtil();
    }

}
```

项目源码地址

https://github.com/macrozheng/mall

推荐阅读

- 看完这篇还不了解**Nginx**，那我就哭了！
- **Nginx**的这些妙用，你肯定有不知道的！
- **Github**标星**25K+Star**，**SpringBoot**实战电商项目**mall**出**SpringCloud**版本啦！
- 在**Docker**容器中部署整套基于**Spring Cloud**的微服务架构，看这篇就对了！
- “失败”的北漂十年，我真的尽力了。。。
- 如何判断一家互联网公司要倒闭了？
- 虚拟机安装及使用**Linux**，看这一篇就够了！
- 涵盖大部分核心组件使用的 **Spring Cloud** 教程，一定要收藏哦！
- 我的**Github**开源项目，从**0**到**20000 Star**！

欢迎关注，点个在看

收录于合集 #mall学习教程（技术要点篇） 17

上一篇
Java 8都出那么久了，Stream API了解下?

下一篇
前后端分离项目，如何优雅实现文件存储!

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