**JWT组件应用说明**

2021-03-08

# JWT组件应用说明

## 编写目的

微服务架构，前后端分离目前已成为互联网项目开发的业界标准，其核心思想就是前端（APP、小程序、H5页面等）通过调用后端的API接口，提交及返回JSON数据进行交互。

在前后端分离项目中，首先要解决的就是登录及授权的问题。微服务架构下，传统的session认证限制了应用的扩展能力，无状态的JWT认证方法应运而生，该认证机制特别适用于分布式站点的单点登录（SSO）场景。

需要技术储备 SpringSecurity+jwt权限认证相关基本知识。

## 组件应用

### 引入组件

可引用jar包，也可以源码作为子模块

 

### 过滤器注入

启动类注入过滤器bean

@Bean

**public** FilterRegistrationBean registration(JwtAuthenticationTokenFilter filter) {

FilterRegistrationBean registration = **new** FilterRegistrationBean(filter);

registration.setEnabled(**false**);

**return** registration;

}

### yml配置jwt信息

#jwt

jwt:

# 使用Base64对该令牌进行编码 (制定您的密钥) 256

base64-secret: yuantiaokj62672000yuantiaokj62672000yuantiaokj62672000cnzz

# 令牌过期时间 此处单位/毫秒

token-validity-in-seconds: 14400000

#放行url（静态资源已放行）

permitMatchers:

- /JwtTokenController/getToken

### 认证，权限校验，用户信息获取demo

/\*\*  
 \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \* Copyright © 2020 远眺科技 Inc.All rights reserved. \* \*\*  
 \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 \*  
 \* @program: module-lib  
 \* @description: 测试  
 \* @author: cnzz  
 \* @create: 2020-06-04 16:39  
 \*\*/  
@RestController  
@Slf4j  
@RequestMapping("/JwtTokenController")  
public class JwtTokenController {  
 @Resource  
 JwtTokenUtils jwtTokenUtils;  
 @Resource  
 JwtSecurityProperties jwtSecurityProperties;  
  
 /\*\*  
 \* 登录认证生成token  
 \*  
 \* @return  
 \*/  
 @GetMapping("/getToken")  
 public String getToken() {  
 List<String> permissions = new ArrayList<>();  
 permissions.add("admin");  
 permissions.add("afds");  
 String token = jwtTokenUtils.createToken("1223", "abcc", permissions);  
 log.info("获取的|token={}", token);  
 return token;  
 }  
  
 /\*\*  
 \* 从token中获取用户信息  
 \*  
 \* @param httpServletRequest  
 \* @return  
 \*/  
 @GetMapping("/noToken")  
 public String noToken(HttpServletRequest httpServletRequest) {  
 //获取token  
 String bearerToken = httpServletRequest.getHeader(jwtSecurityProperties.getHeader());  
 log.info("获取token|bearerToken={}", bearerToken);  
 //获取token用户信息  
 Claims claims = jwtTokenUtils.getClaimsFromToken(bearerToken);  
 String id = claims.getId();  
 log.info("获取token用户信息|id={}", id);  
 return id;  
 }  
  
  
 /\*\*  
 \* 权限认证注解  
 \*  
 \* @return  
 \*/  
 @RequestMapping("/validPermission")  
 @PreAuthorize("hasAnyAuthority('admin')")  
 public String validPermission() {  
 log.info("进入|validPermission");  
 return "OK";  
 }  
}

## 单点SSO实现时序图

