2-15 创建ShiroConfig类

我们要创建的 `ShiroConfig` 类，是用来把 `OAuth2Filter` 和 `OAuth2Realm` 配置到Shiro框架，这样我们辛苦搭建的Shiro+JWT才算生效。

在com.example.emos.wx.config.shiro中创建ShiroConfig类。

```
package com.example.emos.wx.config.shiro;
import org.apache.shiro.mgt.SecurityManager;
import org.apache.shiro.spring.LifecycleBeanPostProcessor;
import org.apache.shiro.spring.security.interceptor.AuthorizationAttributeSourceAdvisor;
import org.apache.shiro.spring.web.ShiroFilterFactoryBean;
import org.apache.shiro.web.mgt.DefaultWebSecurityManager;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import javax.servlet.Filter;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.Map;
@Configuration
public class ShiroConfig {
    @Bean("securityManager")
    public SecurityManager securityManager(OAuth2Realm oAuth2Realm) {
        DefaultWebSecurityManager securityManager = new DefaultWebSecurityManager();
        securityManager.setRealm(oAuth2Realm);
        securityManager.setRememberMeManager(null);
        return securityManager;
    }
    @Bean("shiroFilter")
    public ShiroFilterFactoryBean shiroFilter(SecurityManager securityManager,OAuth2Filter
oAuth2Filter) {
        ShiroFilterFactoryBean shiroFilter = new ShiroFilterFactoryBean();
        shiroFilter.setSecurityManager(securityManager);
        //oauth过滤
        Map<String, Filter> filters = new HashMap<>();
        filters.put("oauth2", oAuth2Filter);
        shiroFilter.setFilters(filters);
        Map<String, String> filterMap = new LinkedHashMap<>();
        filterMap.put("/webjars/**", "anon");
        filterMap.put("/druid/**", "anon");
        filterMap.put("/app/**", "anon");
        filterMap.put("/sys/login", "anon");
        filterMap.put("/swagger/**", "anon");
        filterMap.put("/v2/api-docs", "anon");
        filterMap.put("/swagger-ui.html", "anon");
        filterMap.put("/swagger-resources/**", "anon");
        filterMap.put("/captcha.jpg", "anon");
        filterMap.put("/user/register", "anon");
        filterMap.put("/user/login", "anon");
        filterMap.put("/test/**", "anon");
        filterMap.put("/**", "oauth2");
        shiroFilter.setFilterChainDefinitionMap(filterMap);
        return shiroFilter;
    }
    @Bean("lifecycleBeanPostProcessor")
    public LifecycleBeanPostProcessor lifecycleBeanPostProcessor() {
        return new LifecycleBeanPostProcessor();
    }
    @Bean
    public AuthorizationAttributeSourceAdvisor
authorizationAttributeSourceAdvisor(SecurityManager securityManager) {
```

```
53.         AuthorizationAttributeSourceAdvisor advisor = new
    AuthorizationAttributeSourceAdvisor();
54.         advisor.setSecurityManager(securityManager);
55.         return advisor;
56.     }
57. }
```