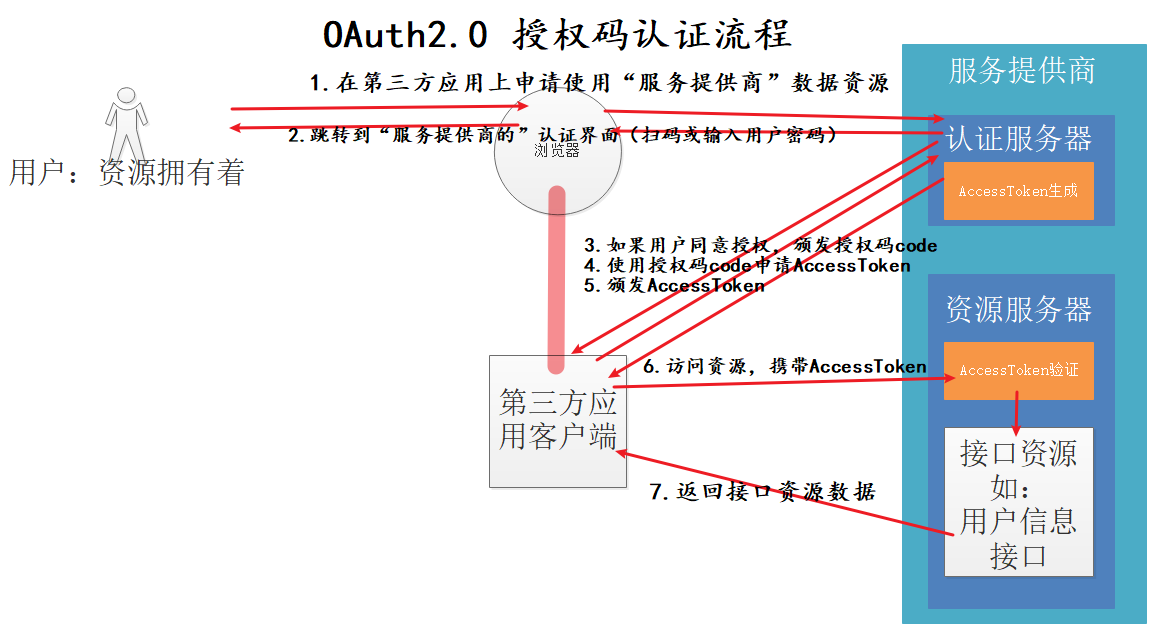
Spring Security OAuth2之resource\_id配置与验证

字母哥博客 2020-07-28 [原文](https://www.bbsmax.com/link/bzc1TjZRMTk1Vw==)

一、resource\_id的作用



Spring Security OAuth2 架构上分为Authorization Server认证服务器和Resource Server资源服务器。我们可以为每一个Resource Server（一个微服务实例）设置一个resourceid。Authorization Server给client第三方客户端授权的时候，可以设置这个client可以访问哪一些Resource Server资源服务，如果没设置，就是对所有的Resource Server都有访问权限。

二、ResourceServer如何设置ResourceID

在每个ResourceServer实例上设置resourceId，该resourceId作为该服务资源的唯一标识。（假如同一个微服务资源部署多份，resourceId相同）

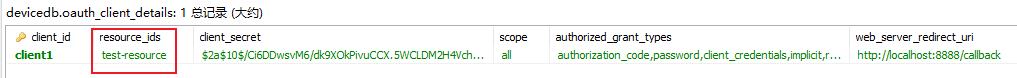
1. @Configuration
2. @EnableResourceServer
3. public class OAuth2ResourceServer extends ResourceServerConfigurerAdapter {
4. private static final String DEMO\_RESOURCE\_ID = "test-resource";
5. @Override
6. public void configure(ResourceServerSecurityConfigurer resources) {
7. resources.resourceId(DEMO\_RESOURCE\_ID)
8. //...... 还可以有有其他的配置
9. }
10. }

三、AuthorizationServer如何设置ResourceIDs

在AuthorizationServer为客户端client配置ResourceID的目的是：限制某个client可以访问的资源服务。

1. @Override
2. public void configure(ClientDetailsServiceConfigurer clients) throws Exception {
3. //配置客户端存储到db 代替原来得内存模式
4. JdbcClientDetailsService clientDetailsService = new JdbcClientDetailsService(dataSource);
5. clientDetailsService.setPasswordEncoder(passwordEncoder);
6. clients.withClientDetails(clientDetailsService);
7. }

这里需要使用JdbcClientDetailsService类和数据库表oauth\_client\_details进行配置的持久化存储，以及动态配置。



三、ResourceID在哪验证

ResourceID当然是在Resource Server资源服务器进行验证（你能不能访问我的资源，当然由我自己来验证）。当资源请求发送到Resource Server的时候会携带access\_token，Resource Server会根据access\_token找到client\_id，进而找到该client可以访问的resource\_ids。如果resource\_ids包含ResourceServer自己设置ResourceID，这关就过去了，就可以继续进行其他的权限验证。

* @EnableResourceServer会给Spring Security的FilterChan添加一个OAuth2AuthenticationProcessingFilter过滤器，过滤所有的资源请求。
* OAuth2AuthenticationProcessingFilter会使用OAuth2AuthenticationManager来验证token。验证Token的时候会去oauth\_client\_details表加载client配置信息。

如果AuthorizationServer认证client1可以访问test-resource，但client1去访问了oauth-rs，会响应如下信息：

1. {"error":"access\_denied","error\_description":"Invalid token does not contain resource id (oauth-rs)"}

具体实现resource\_id验证的源码：OAuth2AuthenticationManager#authenticate(Authentication authentication)

1. public Authentication authenticate(Authentication authentication) throws AuthenticationException {
2. if (authentication == null) {
3. throw new InvalidTokenException("Invalid token (token not found)");
4. }
5. String token = (String) authentication.getPrincipal();
6. OAuth2Authentication auth = tokenServices.loadAuthentication(token);
7. if (auth == null) {
8. throw new InvalidTokenException("Invalid token: " + token);
9. }
10. Collection<String> resourceIds = auth.getOAuth2Request().getResourceIds();
11. if (resourceId != null && resourceIds != null && !resourceIds.isEmpty() && !resourceIds.contains(resourceId)) {
12. throw new OAuth2AccessDeniedException("Invalid token does not contain resource id (" + resourceId + ")");
13. }
14. checkClientDetails(auth);
15. if (authentication.getDetails() instanceof OAuth2AuthenticationDetails) {
16. OAuth2AuthenticationDetails details = (OAuth2AuthenticationDetails) authentication.getDetails();
17. // Guard against a cached copy of the same details
18. if (!details.equals(auth.getDetails())) {
19. // Preserve the authentication details from the one loaded by token services
20. details.setDecodedDetails(auth.getDetails());
21. }
22. }
23. auth.setDetails(authentication.getDetails());
24. auth.setAuthenticated(true);
25. return auth;
26. }

下面这段便是验证resourceid的地方

1. Collection<String> resourceIds = auth.getOAuth2Request().getResourceIds();
2. if (resourceId != null && resourceIds != null && !resourceIds.isEmpty() && !resourceIds.contains(resourceId)) {
3. throw new OAuth2AccessDeniedException("Invalid token does not contain resource id (" + resourceId + ")");
4. }

在Spring Security的FilterChain中，OAuth2AuthenticationProcessingFilter在FilterSecurityInterceptor的前面，所以会先验证client有没有此resource的权限，只有在有此resource的权限的情况下，才会再去做进一步的进行其他验证的判断。