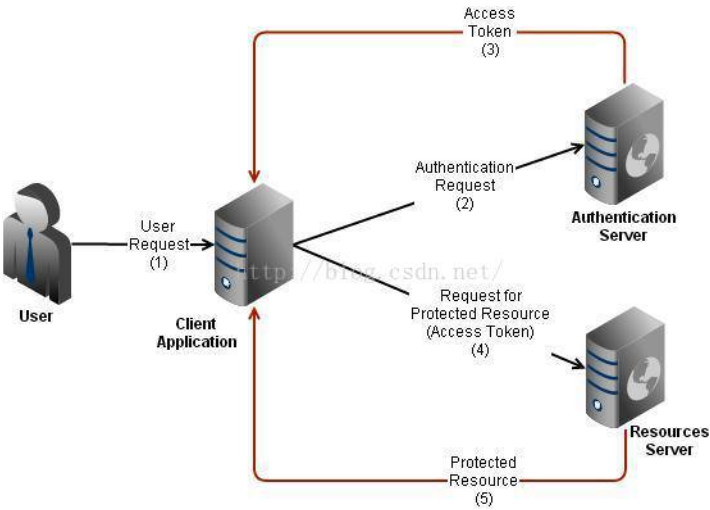


基于spring security 保护的restful框架：Spring Cloud oauth

OAuth2分为四个角色：资源拥有者、资源服务器、授权服务器、客户端

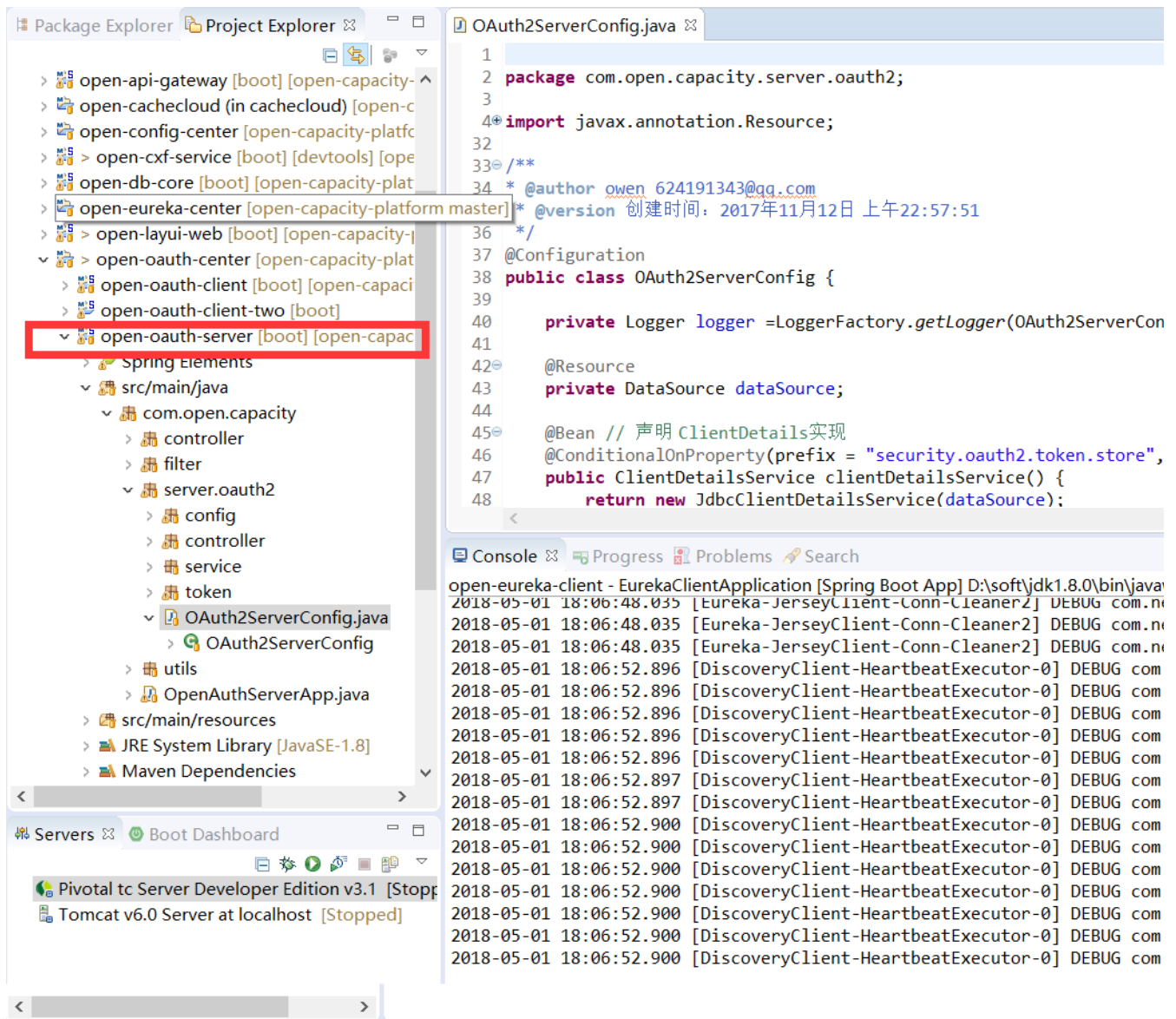


open-oauth-center oauth2 认证服务器 资源服务器分离架构中心

The screenshot displays an IDE interface with the following components:

- Package Explorer / Project Explorer:** Shows a project structure under 'open-capacity-platform'. The 'open-oauth-center' project is highlighted with a red box. Below it are 'open-oauth-client', 'open-oauth-client-two', 'open-oauth-server', 'open-oauth-ssso', and 'db'. The 'pom.xml' file is also visible in the list.
- open-oauth-center/pom.xml:** The XML content is shown, including dependencies for Spring Framework, Spring Social, Spring Cloud, and Spring Security. A comment in Chinese is present: `<!-- 加入spring security spring security oauth2的处理 -->`.
- Console:** Displays logs from the 'open-eureka-client' application, showing heartbeat messages from 'DiscoveryClient-HeartbeatExecutor-0'.
- Servers / Boot Dashboard:** Shows the status of 'Pivotal tc Server Developer Edition v3.1' and 'Tomcat v6.0 Server at localhost'.
- Bottom Bar:** Indicates 'Building workspace (Finished)' and 'OK'.

open-oauth-server 认证中心



## 客户端模式

## Request 2

METHOD

POST

SCHEME :// HOST [ ":" PORT ] [ PATH [ "?" QUERY ]]

http://localhost:8000/oauth/token

▼ QUERY PARAMETERS

+ Add query parameter

HEADERS ⓘ

Form ▼

☒ Content-Type : application/x-www-form-urlencoded ×

+ Add header

Add authorization

BODY ⓘ

☒ grant\_type [ Text ▼]☒ client\_id [ Text ▼]☒ client\_secret [ Text ▼]

+ Add form parameter

☒ application/x-www-form-urlencoded

## Response

200

HEADERS ⓘ

pretty ▼

pragma: no-cache, no-cache  
date: Mon, 09 Apr 2018 01:58:12 GMT  
x-content-type-options: nosniff  
x-frame-options: DENY  
content-type: application/json; charset=UTF-8  
cache-control: no-cache, no-store, max-age=0, must-revalidate, no-store  
transfer-encoding: chunked  
x-xss-protection: 1; mode=block  
x-application-context: open-auth-server:8000  
expires: 0

BODY ⓘ

```
{  
  access_token : "50a01e43-ae1c-48bd-b842"  
  token_type : "bearer",  
  expires_in : 43198,  
  scope : "app"  
}
```

lines nums

密码模式

## Request 13

METHOD

POST

SCHEME :// HOST [ ":" PORT ] [ PATH [ "?" QUERY ] ]

http://127.0.0.1:8000/oauth/token

http://127.0.0.1:8000/oauth/token

HEADERS ⓘ

Form

☒ Content-Type : application/x-www-form-urlencoded ×

☒ Authorization : Basic d2ViQXBwOndiYkFwcA== ×

+ Add header Add authorization

BODY ⓘ

☒ grant\_type [ Text ] =

☒ username [ Text ] =

☒ password [ Text ] =

☒ scope [ Text ] =

+ Add form parameter ☒ application/x-www-form-urlencoded

## Response

200

HEADERS ⓘ

pretty

pragma: no-cache, no-cache

date: Sat, 28 Apr 2018 06:42:28 GMT -3d 3h

x-content-type-options: nosniff

x-frame-options: DENY

content-type: application/json; charset=UTF-8

cache-control: no-cache, no-store, max-age=0, must-revalidate, no-store

transfer-encoding: chunked

x-xss-protection: 1; mode=block

x-application-context: open-auth-server:8000

expires: 0

BODY ⓘ

```
{
  access_token : "33042a20-97ac-4385-831d-e",
  token_type : "bearer",
  refresh_token : "3ee132f2-0228-4b9d-bec9-",
  expires_in : 43160,
  scope : "app"
}
```

lines nums



授权码模式 1.获取授权码 [http://localhost:8000/oauth/authorize?client\\_id=webApp&redirect\\_uri=http://www.baidu.com&state=abc&scope=app&response\\_type=code](http://localhost:8000/oauth/authorize?client_id=webApp&redirect_uri=http://www.baidu.com&state=abc&scope=app&response_type=code)

2.授权码换token


## Request 12



METHOD **POST** SCHEME :// HOST [ ":" PORT ] [ PATH [ "?" QUERY ] ]  
http://localhost:8000/oauth/token



QUERY PARAMETERS

HEADERS <sup>?</sup>  Form 

☒ Content-Type : application/x-www-form-urlencoded ×

☒ Authorization : Basic d2ViQXBwOndlYkFwcA== × 

+ Add header  Add authorization 

BODY <sup>?</sup>  Form 

☒ grant\_type [ Text ▼ ] =

☒ code [ Text ▼ ] =

☒ scope [ Text ▼ ] =

☒ redirect\_uri [ Text ▼ ] =

+ Add form parameter ☒ application/x-www-form-urlencoded

## Response

200

HEADERS <sup>?</sup> pretty 

pragma: no-cache, no-cache

date: Fri, 13 Apr 2018 06:22:02 GMT -6s

x-content-type-options: nosniff

x-frame-options: DENY

content-type: application/json; charset=UTF-8

cache-control: no-cache, no-store, max-age=0, must-revalidate, no-store

transfer-encoding: chunked

x-xss-protection: 1; mode=block

x-application-context: open-auth-server:8000

expires: 0

BODY <sup>?</sup> 

```
{
  access_token : "548dda2c-3de0-44ad-bf74-a4",
  token_type : "bearer",
  refresh_token : "295b824a-e40b-4c92-8abd-d",
  expires_in : 43199,
  scope : "app"
}
```

[lines](#) [nums](#)

open-oauth-client open-oauth-client-two open-api-gateway 均为资源服务器

Package Explorer

Project Explorer

auth [boot] [devtools]

CicadasCms [boot] [devtools]

Goku.Framework.CoreUI [boot] [devtools]

micro-unieap-platform

open-capacity-platform [open-capacity-pl

apollo [open-capacity-platform master]

apollo-gateway [boot] [open-capacity-pla

open-api-gateway [boot] [open-capacity-pla

open-cachecloud (in cachecloud) [open-c

open-config-center [open-capacity-platf

open-cxf-service [boot] [devtools] [ope

open-db-core [boot] [open-capacity-plat

open-eureka-center [open-capacity-platf

open-layui-web [boot] [open-capacity-pla

open-oauth-center [open-capacity-plat

open-oauth-client [boot] [open-capaci

Spring Elements

src/main/java

com.open.capacity

client.oauth2

authorize

config

controller

service

token

OAuth2ClientConfig.java

OAuth2ClientConfig

controller

OAuth2ServerConfig.java

OAuth2ClientConfig.java

```

1
2 package com.open.capacity.client.oauth2;
3
4 import java.util.HashMap;
29
30 /**
31  * @author 作者 owen E-mail: 624191343@qq.com
32  * @version 创建时间: 2018年4月5日 下午19:52:21
33  */
34 @Component
35 @Configuration
36 @EnableResourceServer
37 public class OAuth2ClientConfig extends ResourceServerConfigurerAda
38
39
40 //对应oauth_client_details的 resource_ids字段 如果表中有数据 c
41 private static final String DEMO_RESOURCE_ID = "test";
42
43 @Resource
44 private ObjectMapper objectMapper ; //springmvc启动时自动装
45

```

Console

Progress

Problems

Search

open-eureka-client - EurekaClientApplication [Spring Boot App] D:\soft\jdk1.8.0\bin\java
2018-05-01 18:09:03.108 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:03.108 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:03.118 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:03.119 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:03.119 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:03.119 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:03.119 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:03.120 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:08.120 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:08.122 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:08.122 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:08.122 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:08.126 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:08.127 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:08.127 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:08.127 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:08.127 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.
2018-05-01 18:09:08.127 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.

源码分析

## Spring Security OAuth简介

The diagram illustrates the Spring Security OAuth architecture, divided into three main components:

- 服务提供者 (Service Provider):** This is the client application that interacts with the other two components.
- 认证服务器 (Authorization Server):** This component handles authentication and token generation. It includes:
  - 4种授权模式 (4 authorization modes):** These are the different ways a client can obtain a token.
  - 自定义的认证 (Custom authentication):** This is a custom implementation of the authentication process.
  - Token的生成存储 (Token generation and storage):** This is where the tokens are generated and stored.
- 资源服务器 (Resource Server):** This component provides the resources (REST services) that are protected by OAuth. It includes:
  - SpringSecurity过滤器链 (Spring Security filter chain):** This is the chain of filters that protect the resources.
  - OAuth2AuthenticationProcessingFilter (OAuth2 authentication processing filter):** This filter is responsible for processing the OAuth2 authentication request.
  - 资源 (rest 服务) (Resources (REST services)):** These are the actual REST services that the client can access.

Arrows indicate the flow of data and control between these components, showing how the client interacts with the authorization server to get a token and then uses that token to access the resource server.

资源服务器校验核心逻辑



```
OAuth2AuthenticationProcessingFilter.class OAuth2AuthenticationManager.class
67 * authorization header). Loads an authentication from the {@link ResourceServerTokenServices} and checks
68 * resource id is contained in the {@link AuthorizationRequest} (if one is specified). Also copies authen
69 * details over from the input to the output (e.g. typically so that the access token value and request d
70 * be reported later).
71 *
72 * @param authentication an authentication request containing an access token value as the principal
73 * @return an {@link OAuth2Authentication}
74 *
75 * @see org.springframework.security.authentication.AuthenticationManager#authenticate(org.springframework
76 */
77 public Authentication authenticate(Authentication authentication) throws AuthenticationException {
78
79     if (authentication == null) {
80         throw new InvalidTokenException("Invalid token (token not found)");
81     }
82     String token = (String) authentication.getPrincipal();
83     OAuth2Authentication auth = tokenServices.loadAuthentication(token);
84     if (auth == null) {
85         throw new InvalidTokenException("Invalid token: " + token);
86     }
87
88 Console Progress Problems Search
open-eureka-client - EurekaClientApplication [Spring Boot App] D:\soft\jdk1.8.0\bin\javaw.exe (2018年5月1日 下午5:25:59)
2018-05-01 18:11:17.955 [Eureka-JerseyClient-Conn-Cleaner2] DEBUG com.netflix.discovery.shared.MonitoredConnectionP
2018-05-01 18:11:17.955 [Eureka-JerseyClient-Conn-Cleaner2] DEBUG com.netflix.discovery.shared.NamedConnectionPool
2018-05-01 18:11:18.049 [Eureka-JerseyClient-Conn-Cleaner2] DEBUG com.netflix.discovery.shared.MonitoredConnectionP
2018-05-01 18:11:18.049 [Eureka-JerseyClient-Conn-Cleaner2] DEBUG com.netflix.discovery.shared.NamedConnectionPool
2018-05-01 18:11:18.049 [Eureka-JerseyClient-Conn-Cleaner2] DEBUG com.netflix.discovery.shared.NamedConnectionPool
2018-05-01 18:11:18.049 [Eureka-JerseyClient-Conn-Cleaner2] DEBUG com.netflix.discovery.shared.NamedConnectionPool
2018-05-01 18:11:18.331 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.netflix.discovery.shared.MonitoredConnectio
2018-05-01 18:11:18.332 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.netflix.discovery.shared.NamedConnectionPoc
2018-05-01 18:11:18.332 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.netflix.discovery.shared.NamedConnectionPoc
2018-05-01 18:11:18.332 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.netflix.discovery.shared.NamedConnectionPoc
2018-05-01 18:11:18.332 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.netflix.discovery.shared.NamedConnectionPoc
2018-05-01 18:11:18.344 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.netflix.discovery.shared.MonitoredConnectio
2018-05-01 18:11:18.344 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.netflix.discovery.shared.NamedConnectionPoc
2018-05-01 18:11:18.344 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.netflix.discovery.shared.NamedConnectionPoc
2018-05-01 18:11:18.344 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.netflix.discovery.shared.NamedConnectionPoc
2018-05-01 18:11:18.344 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.netflix.discovery.shared.transport.jersey.A
2018-05-01 18:11:18.344 [DiscoveryClient-HeartbeatExecutor-0] DEBUG com.netflix.discovery.DiscoveryClient - Discove
```

认证服务器 资源服务器统一实现配置tokenstore，同时直接连接redis校验token是否有效

DefaultTokenServices - org.springframework.security.oauth2

RemoteTokenServices

一般为单点登录sso时使用，此方式需要通过http方式连接认证中心，同时还需要访问redis，多一层网络连接，建议不采用此方式