

Usando o Client Credentials

Usando o Client Credentials

- [Usando o Client Credentials](#)

Usando o Client Credentials

Neste tutorial vamos aprender o seguinte:

1. Configuração no Cerberus (Authorization Server);
2. Obtendo um token para acessar a API protegida (Client);
3. Criando uma API protegida (Resource Server);

Configuração no Cerberus (Authorization Server)

1. No Cerberus, logado como Admin, vá em Aplicações;
2. Cadastre uma nova aplicação, por exemplo, nome "NovaApp001", descrição "Aplicação para testar o Client Credentials Grant Type";
3. Ainda no cadastro da nova aplicação, clique no botão Editar, em seguida vá na aba Clientes OAuth, e clique no botão azul com o símbolo de soma;
4. Preencha o dialog Adicionar Cliente OAuth da maneira abaixo. Perceba que em Tipos de garantia de autorização Client Credentials foi selecionado, e que em Autoridades definimos duas: NovaApp003:acessoA e NovaApp003:acessoB

Atenção: utilizamos as Autoridades para gerenciar o acesso aos recursos do Resource Sever (API protegida).

Cliente OAuth

Nome
NovaApp002

ID do Cliente - Opcional
NovaApp002Id

Escopo de acesso - Opcional
Leitura, Escrita, Informação

Escopos auto-aprovados - Opcional
Leitura, Escrita, Informação

Tipos de garantia de autorização
Client Credentials

Autoridades
NovaApp003:acessoB
NovaApp003:acessoA

URI de redirecionamento - Opcional

Validade do Access Token (em segundos)
3600

Validade do Refresh Token (em segundos)
43200

Secret obrigatório?

Secret pura
tYAPhZofjCGDhRb3nsRfGHKvT0qhoRTGh3bvSIT3bTqM0RnHMxNcmvT
vHkU9V0CinnMzzxUucc5UaaQsxhoKtQ34OIVMkMIF9iVCsaWgFPQgJX
oi9oCp22rkp4WVxHRnip7N3A89awFZPy22stnuLaywLpFRxa8H9sHsdHJJ

Esta chave só será mostrada neste momento. Guarde-a em um lugar seguro. Caso a perca, será necessário criar uma nova.

Editar Fechar

5. Ao clicar no botão Salvar, anote o Secret pura. O ID do Cliente e a Secret pura são o user e o password que usaremos para obter um token.

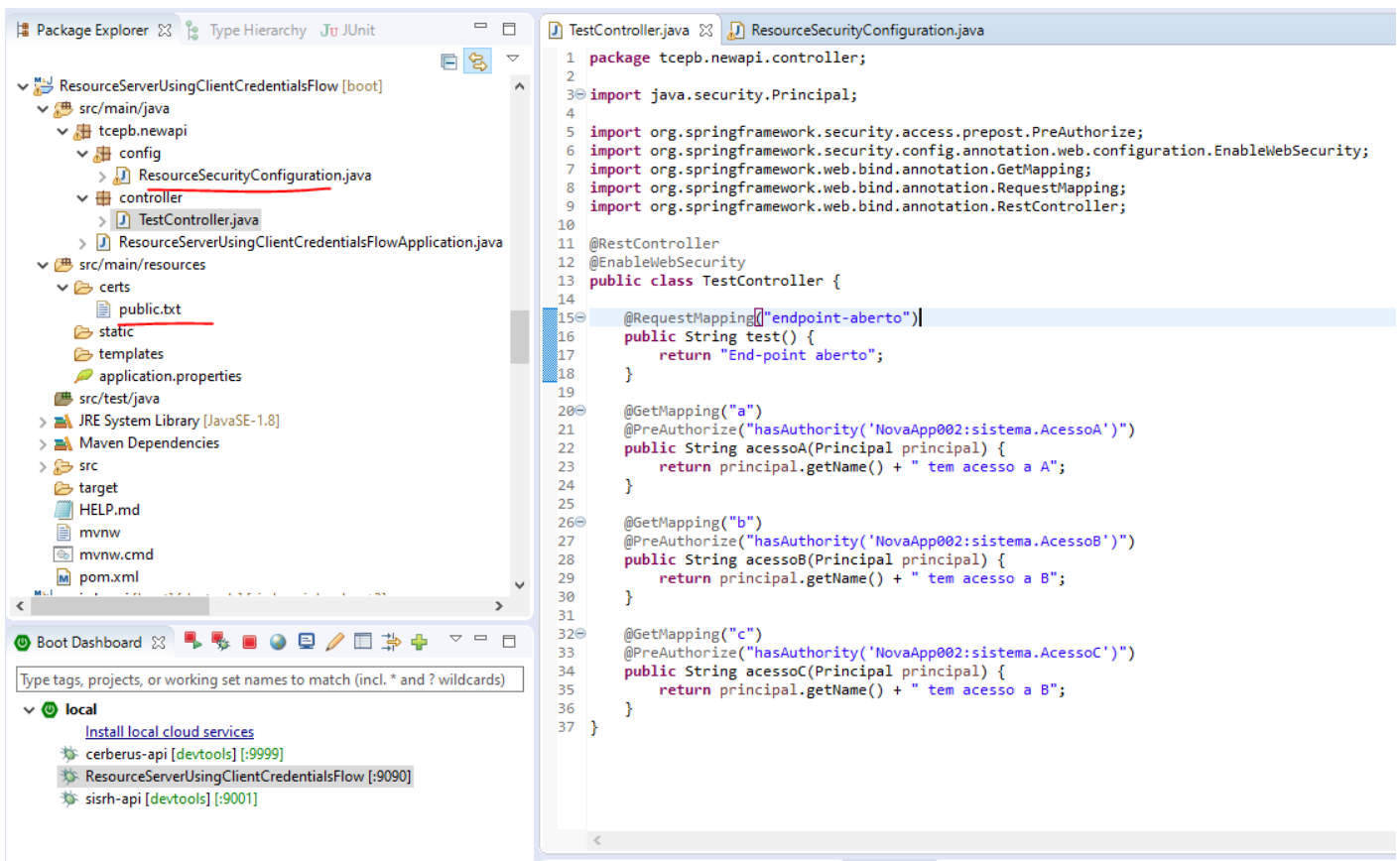
Usando o site JWT para decodificar o token obtido, temos o seguinte:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzY29wZSI6WyJyZWZkIiwid3JpdGUuIiwiaWF0Ij0iOiJ0b3ZhbWMDAyIn0.TmhbCeWJ8K39P_R4B29pLQ1PBEPtVkbzbEnooo6xHKF633swB4qjEuR17Z1veWMM-NN3AazYdsVj421AaQvPszqwZfQvyTE5PECmQU5-b38pAcpvTmdwkIADf_j2JkoLGTQgr9EQszFKO9qgi1UFIa34gMts3h1M1su08LKXd_-C81ECYJGkSMbPMhQkFv2mCGof9J-kd6j0U_DwEHvJbbEV38eq4rRsuJPKggHZoS02w9oEuzkRceLIm-Aji9DoNHXhkQC13LN6ixt_eA6MOKFqsGu_v91KbeYG4PBPMT0QJ85GTcrwDTQdveBpZm2j0tb9fYLY2yRW25xiXt4Yw
```

HEADER: ALGORITHM & TOKEN TYPE
<pre>{ "alg": "RS256", "typ": "JWT" }</pre>
PAYLOAD: DATA
<pre>{ "scope": ["read", "write", "info"], "exp": 1568320851, "authorities": ["NovaApp002:sistema.AcessoA", "NovaApp002:sistema.AcessoB"], "jti": "abf7deb0-1c4b-4ae5-8f31-54170e85980f", "client_id": "NovaApp002" }</pre>
VERIFY SIGNATURE
<pre>RSASHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), Public Key or Certificate. Ente</pre>

Criando uma API protegida (Resource Server)

O resource server normalmente é uma API, em [anexo](#) temos o boilerplate de uma que pode ser reutilizada como base. Ela basicamente é uma adaptação do sistrh-api, onde a classe de configuração principal é a `tcepb.newapi.config.ResourceSecurityConfiguration` que basicamente é uma adaptação da classe do sistrh-api `tcepb.rh.config.ResourceConfig`. Copiamos também a chave pública utilizada no sistrh-api, ela está localizada em `src/main/resources/certs/public.txt`



Referências:

1. Spring Boot + OAuth 2 Client Credentials Grant - Hello World Example. URL: <https://www.javainuse.com/spring/springboot-oauth2-client-grant> >. Acessado em: 12/09/2019;