Securing Your Microservices with a Declarative Model

Richard Seroter
SENIOR DIRECTOR OF PRODUCT, PIVOTAL

@rseroter

Overview

The role of security in microservices

Problems with the status quo

What is Spring (Cloud) Security?

What is OAuth 2.0?

OAuth 2.0 authorization code grant type

Creating a Resource Server

OAuth 2.0 password credential grant type

Creating a custom Authorization Server

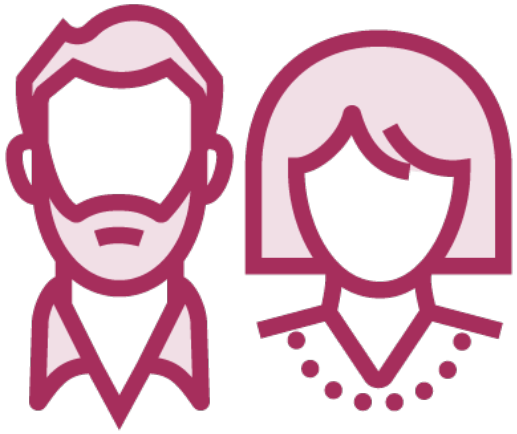
OAuth 2.0 client credentials grant type

Adding method access rules

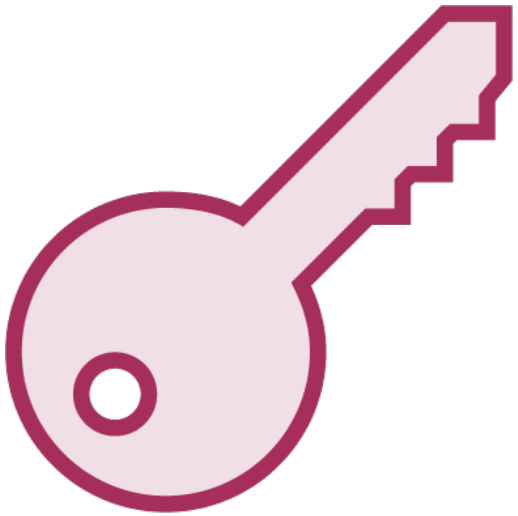
Advanced token options

Summary

The Role of Security in Microservices

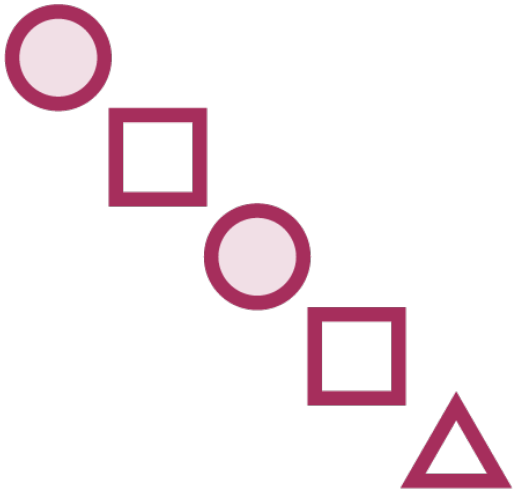


User Authentication / Authorization



Single Sign-On

Data Security



Interoperability

Problems with the Status Quo

Credentials embedded in applications

Unnecessary permissions

Differentiating users and machines

Not optimized for diverse clients

Spring (Cloud) Security

Service authorization
powered by OAuth 2.0.

What Is OAuth 2.0?

Protocol for conveying authorization

Provides authorization flow for various clients

Obtain limited access to user accounts

Separates idea of user and client

Access token carries more than identity

NOT an authentication scheme

Actors in an OAuth 2.0 Scenario

Resource Owner

Entity that grants access to a resource. Usually, you!

Resource Server

Server hosting the protected resource.

Client

App making protected resource requests on behalf of resource owner.

AuthZ Server

Server issuing access tokens to clients.

Glossary of OAuth 2.0 Terms

- Access Token
- Refresh Token
- Scope
- Client ID / Secret
- OpenID Connect
- JWT

How Spring Supports OAuth 2.0

Code annotations

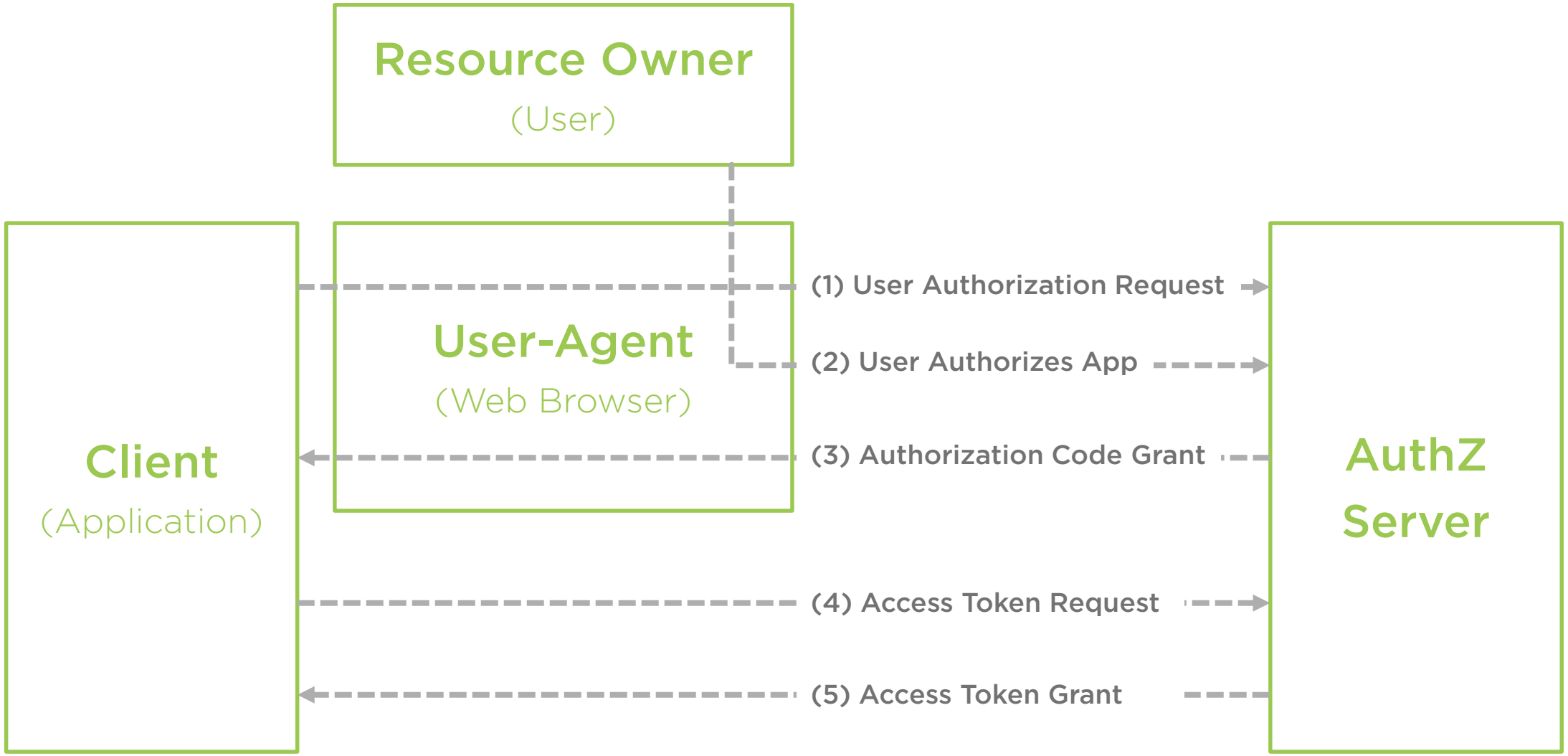
Token storage options

OAuth 2.0 endpoints

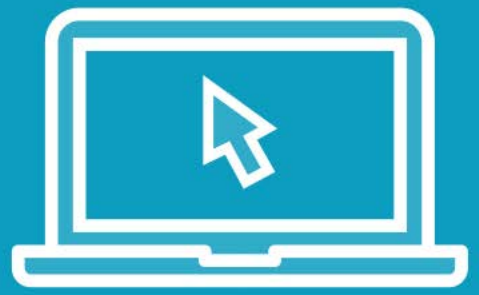
Numerous extensibility points

Abstract OAuth Flow

OAuth 2.0 Grant Type: Authorization Code



Demo



Build back-office Toll reporting site

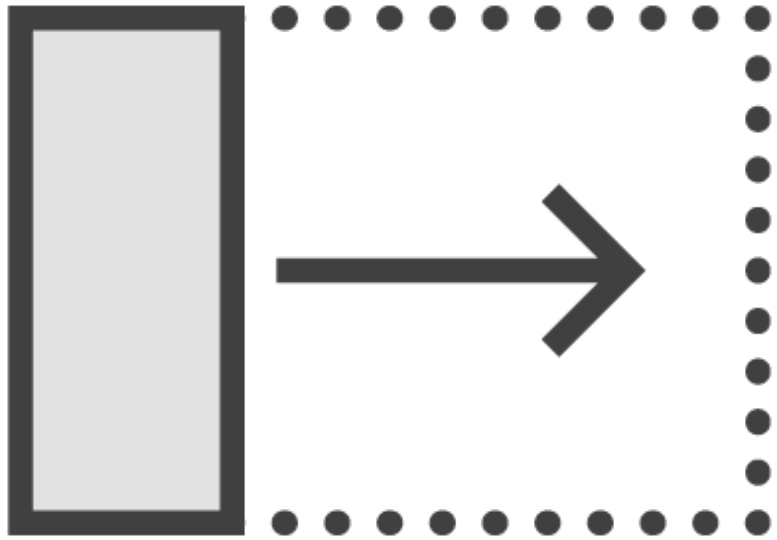
Add OAuth2 annotations to controller

Authenticate and authorize via GitHub

Watch redirects during flow

Choose which pages to protect

Creating a Resource Server and Routing Tokens to Downstream Services



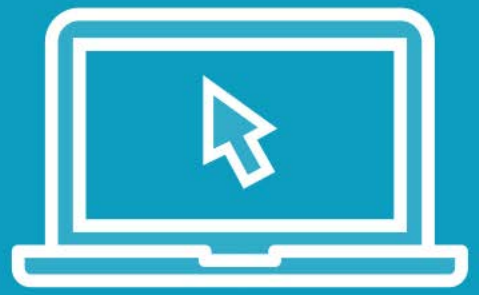
Use @EnableResourceServer annotation

Can combine AuthZ and Resource servers

user-info-uri and token-info-uri properties

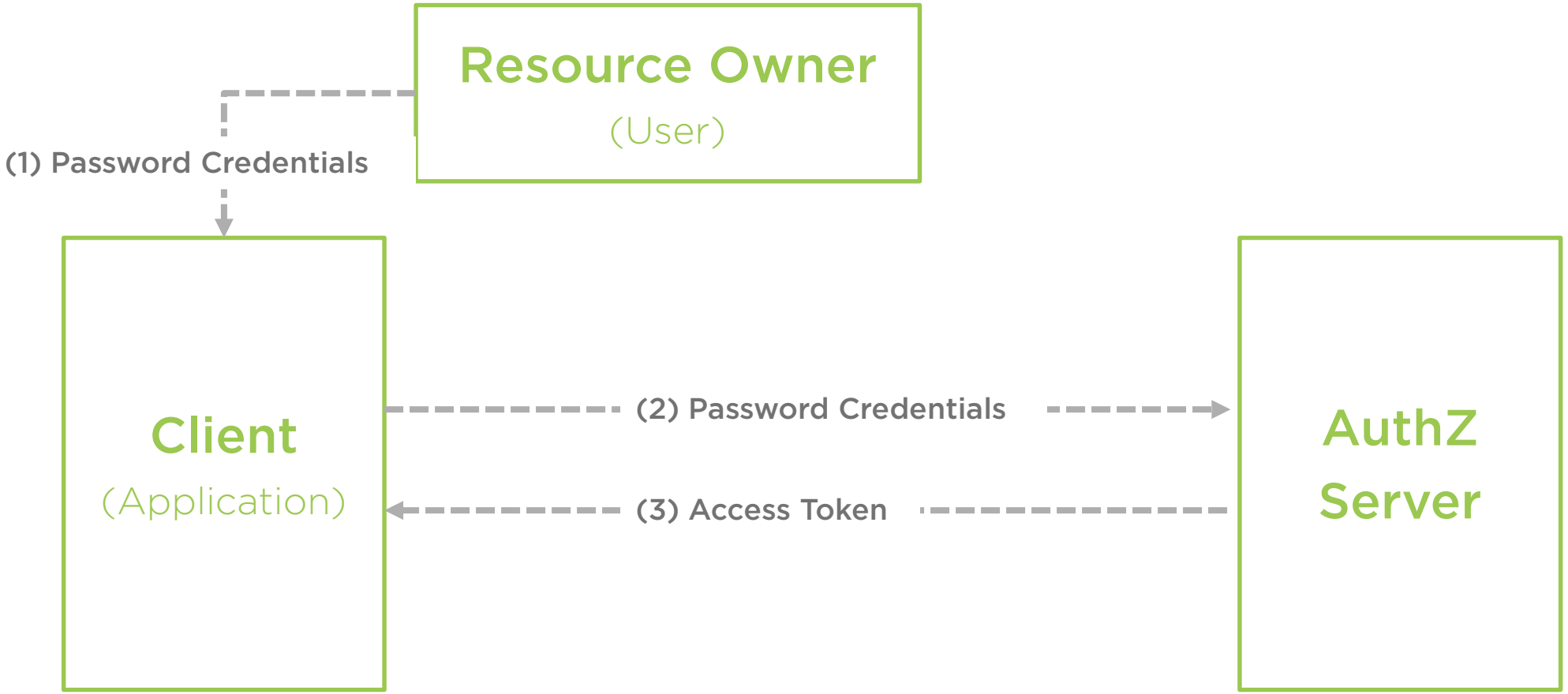
OAuth2RestTemplate bean

Demo



- Create resource server to return toll data
- Add @EnableResourceServer annotation
- Update application.properties
- Test via HTTP calls
- Update web client to securely call service

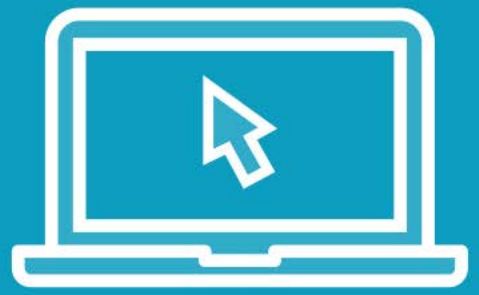
OAuth 2.0 Grant Type: Resource Owner Password Credentials



Creating a Custom Authorization Server

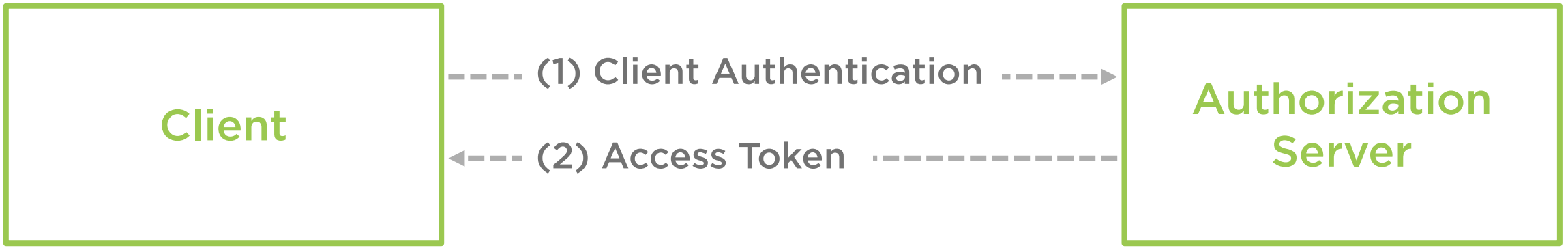
- Lit up with @EnableAuthorizationServer
- Set of endpoints provided
- Property settings drive behavior
- Secured with Basic Authentication

Demo



- Create new Spring app
- Add Authorization Server annotation
- Set properties for the server
- Test token generation with API calls
- Update resource server to use custom authorization server
- Test resource server with API calls
- Create application that relies on system user
- Call resource server from client app with password credentials

OAuth2 Grant Type: Client Credentials

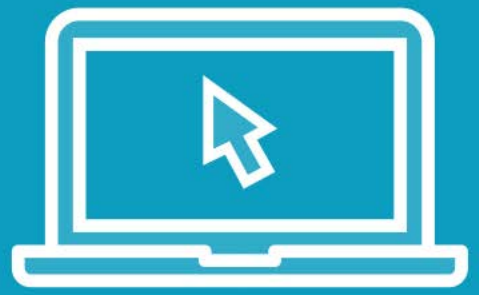


```
@PreAuthorize(”
#oauth2.hasScope('READ') and
hasAuthority('ROLE_ADMIN')")

@RequestMapping("/")

public String secureCall()

        return "success!";

    }
```

Adding Access Rules

- Can method be called?
- Check OAuth2 scope
- Check user role

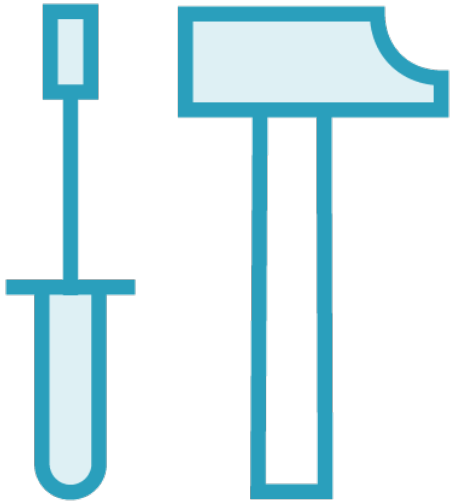
Demo



- Create extension class to get OAuth2 filters
- Add role filter to method and test
- Add OAuth2 filter to method and test
- Apply *both* filter to method and test

Advanced Token Options

JdbcTokenStore for scaling AuthZ servers



Token extensibility via TokenEnhancers

OpenID Connect for authentication and JWTs for encrypted tokens

Summary

- Overview
- The role of security in microservices
- Problems with the status quo
- What is Spring (Cloud) Security?
- What is OAuth 2.0?
- OAuth 2.0 authorization code grant type
- Creating a Resource Server
- OAuth 2.0 password credential grant type
- Creating a custom Authorization Server
- OAuth 2.0 client credentials grant type
- Adding method access rules
- Advanced token options