Layering Scoped-based Authorization

Current State

Secured with Oauth2, OIDC and JWT

User credentials and data is stored in an Identity server, away from the application code

None of our services handle passwords, only the Identity server does.

User Identification Claim

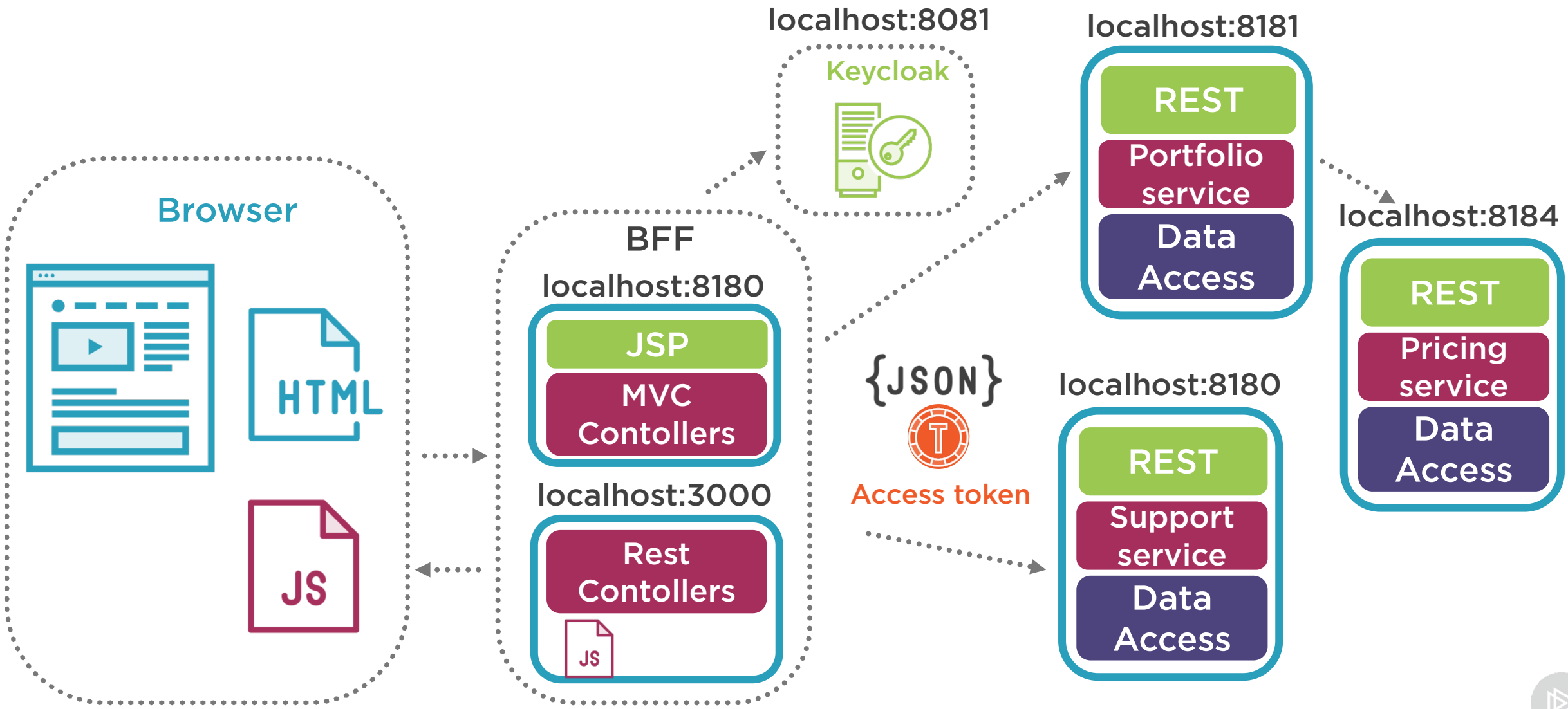
Globally Unique Identifier (GUID)

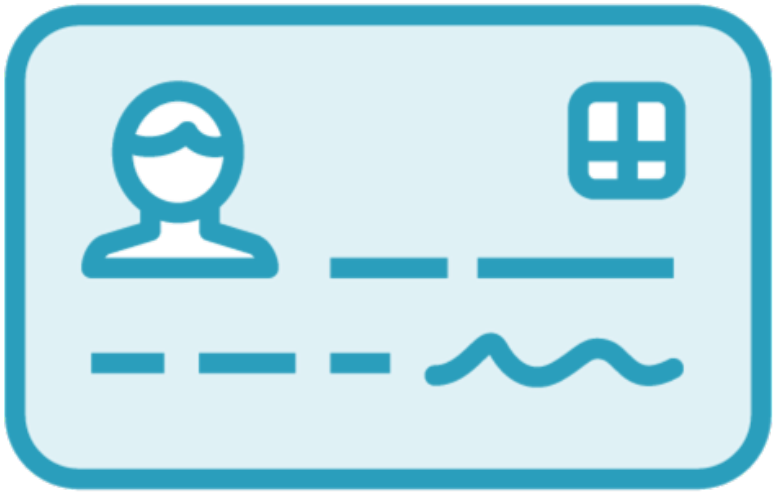
d6888fef-3bf2-46f3-94a2-039a0f7ff0a2

Username

joesmith

Security Architecture





Authentication

Who is the subject

Authorization

What can they do

TRAIN TICKET

T1 04234887692

NAME OF PASSENGER
JOHN SMITH

from
LONDON

to
PARIS

PRICE
60 EURO

TRAIN
4581

CARRIAGE №
4

PLATFORM
2

SEAT
35

DATE
15 APRIL

DEPATURE
18:45

ARRIVE
21:05

ECONOMY CLASS

TRAIN TICKET

NAME OF PASSENGER
JOHN SMITH

from
LONDON

to
PARIS

SEAT
35

TRAIN
4581

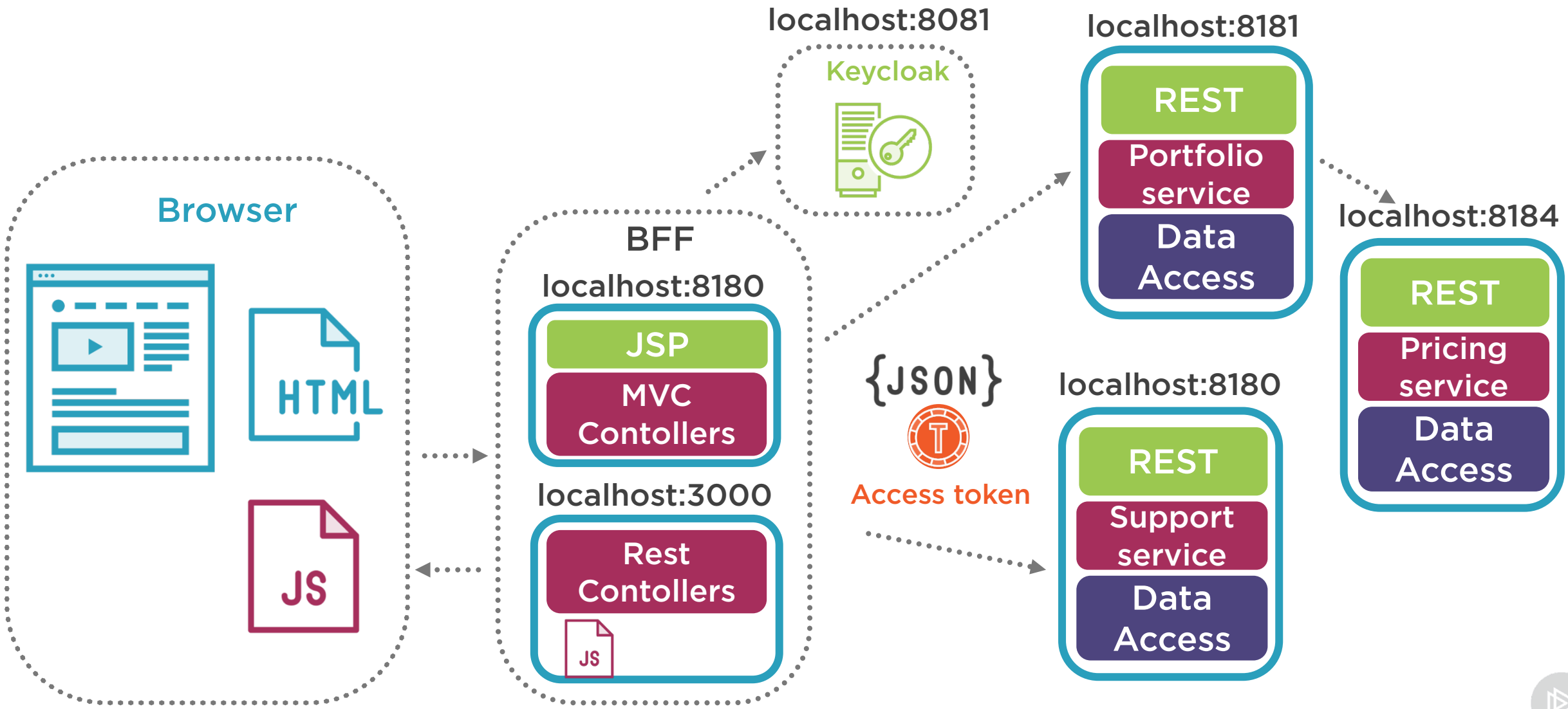
ECONOMY CLASS

More finer grained authorization

- Scopes
- Authorities

Demo

Security Architecture



AUTHORITIES

ROLES

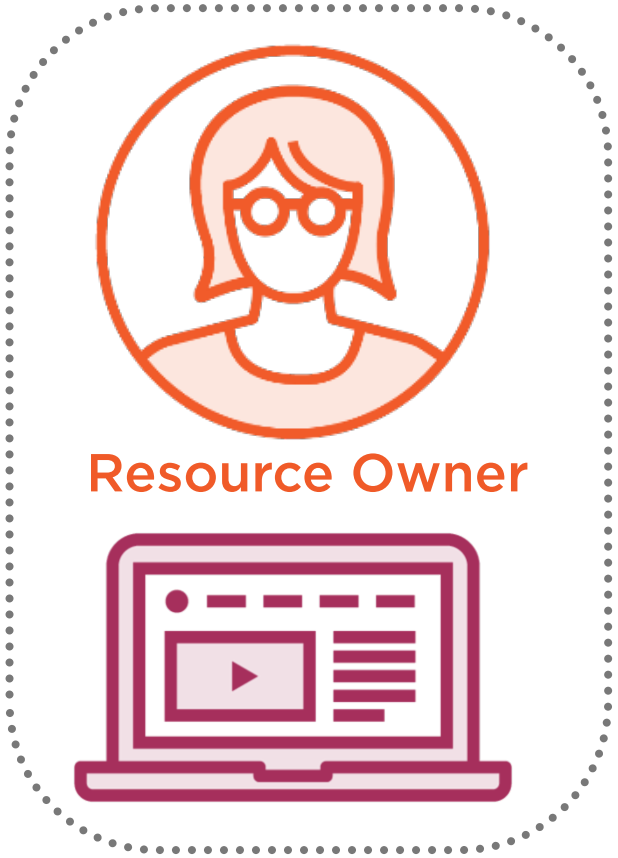
SCOPES

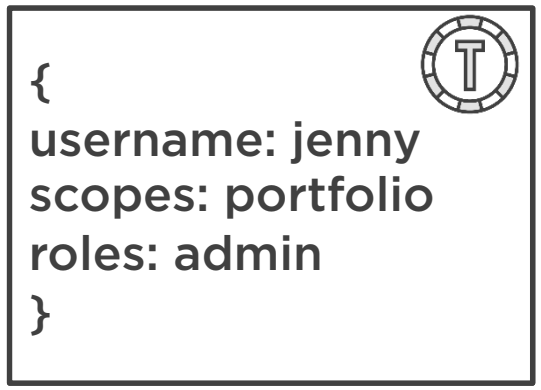
Scopes

In Oauth2.0 scopes, define the scope of the access the resource owner has approved the client to perform on their behalf.

Scope Claim

```
{
    scp : portfolio-service
}
{
    scopes: portfolio-service
}
```


Resource Owner


{
username: jenny
scopes: portfolio
roles: admin
}

/admin


Resource Server

To avoid name collisions

- Roles are prefixed with: ROLE_
- Scopes are prefixed with: SCOPE_
- Authorities are not prefixed

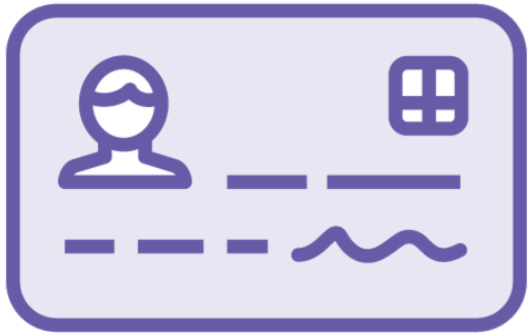
Granted Authorities

```
hasRole("ADMIN")

hasAuthority("ROLE_ADMIN")

hasAuthority("SCOPE_ADMIN")
```

@EnableGlobalMethodSecurity

```
@PreAuthorize("hasRole('ADMIN')")
@PreAuthorize("hasAuthority('SCOPE_portfolio-service-admin')")
@PreAuthorize("hasAnyAuthority('ROLE_ADMIN',
                               'SCOPE_portfolio-service-admin')")
```

Scoles



OIDC

OpenID Connect

JWT

Jason Web Token

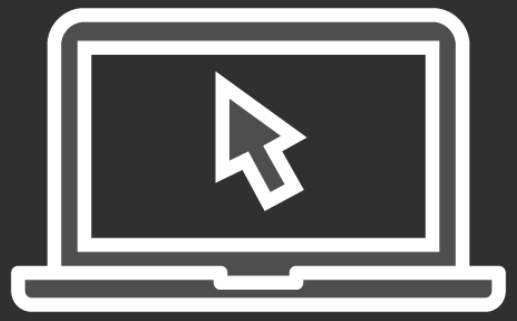
Overview

How to modify the authorization request to the authorization server using a custom: Oauth2AutherizationRequestResolver

Identify and address security vulnerabilities in our application

Why a valid token is not enough for authorization, and how to add additional token validation

Demo

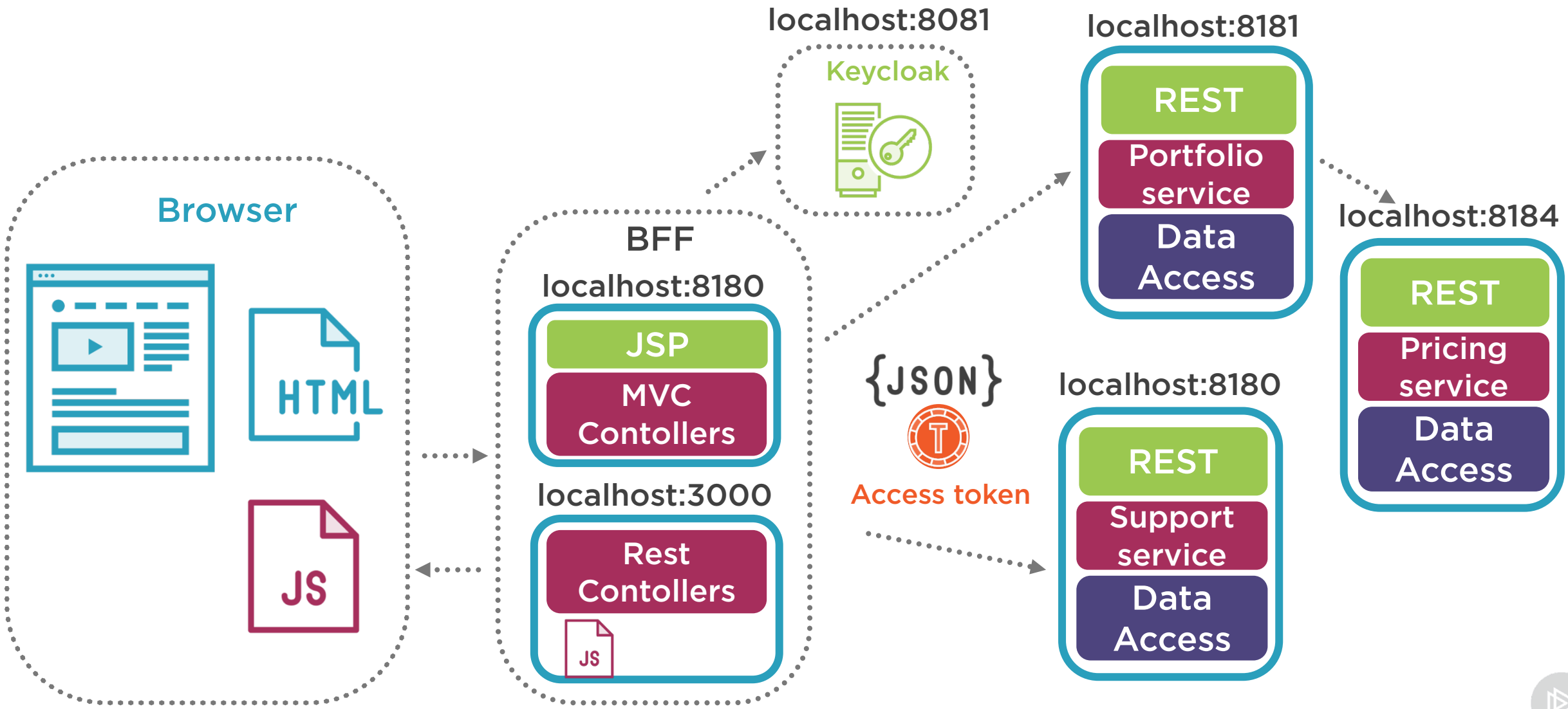


Resource Server

Actions to validate the token:

- Check the signature
- Check token is not expired
- Check if issuer URI was provided check issuer matches

Security Architecture

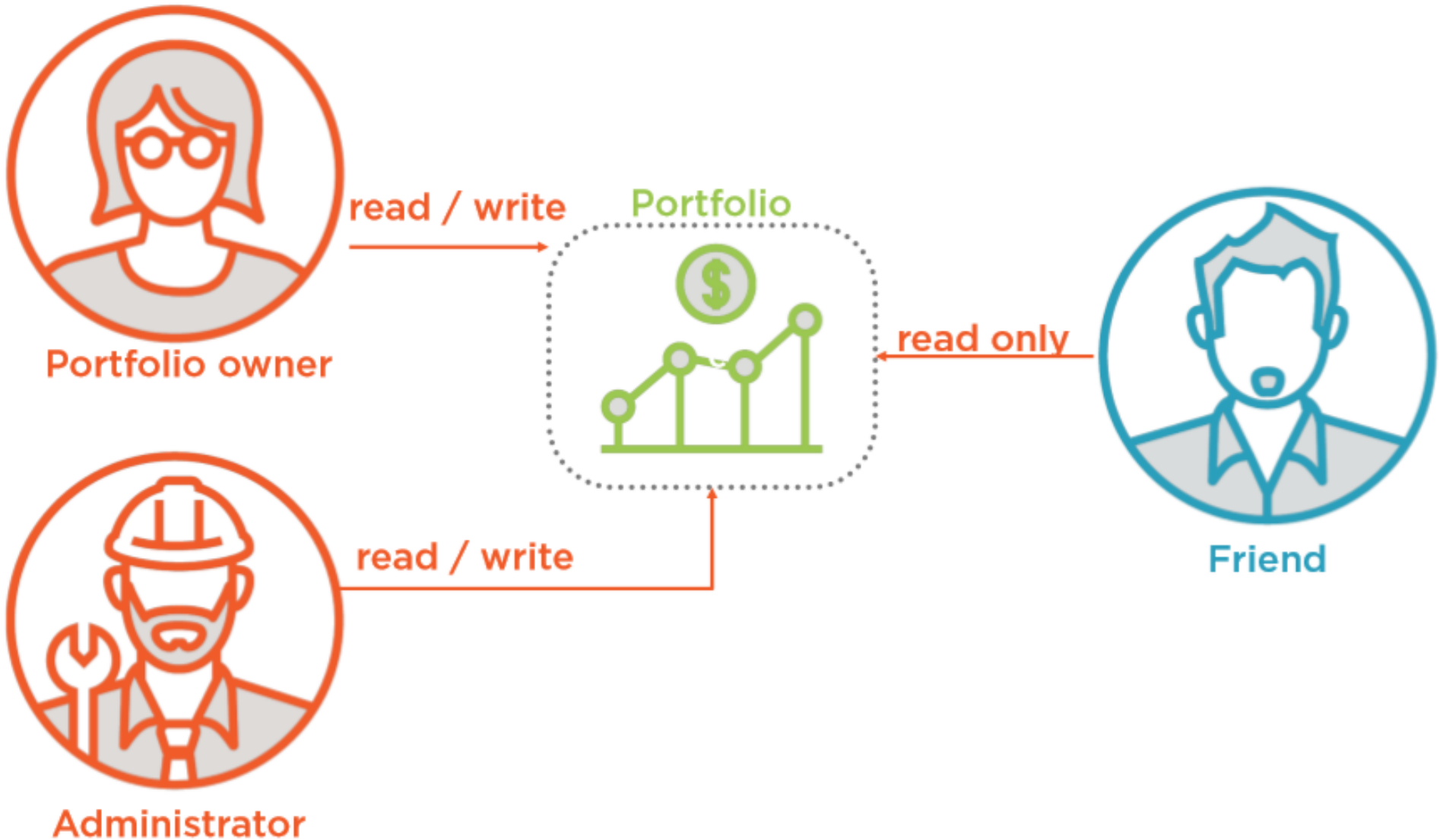


Spring Security

Browser

URLs

Methods and Domain objects


Portfolio owner
read / write
Portfolio
read only
Friend
Administrator
read / write

All Done!

Spring Security 5.2 +

- Opaque tokens
- Jwt bearer and token exchange grants

Defence in Depth

Ticket

Passport

Plane

Seat

Defence in Depth

Oauth2 and OIDC to authenticate the user and client, issued a token

Check token signature and expiry to ensure it is valid

Check issuer and audience to ensure source and destination of the token

Scopes to ensure the client has received consent from the resource owner, and roles to ensure the resource owner is entitled to give the consent

Security is important