O'REILLY®

Spring Security for REST APIs

September 2020

Welcome!

Not that Josh… :)

I’m Josh Cummings, a maintainer of Spring Security

You’ll find me at

https://{gh}/spring-projects/spring-security

https://{gh}/terracotta-bank/terracotta-bank

https://{gh}/eugenp/tutorials

I'm am building a REST API and need to secure it with OAuth 2.0

I need to modernize the security of an existing REST API

What are your goals?

I want to get a better understanding of Spring Security in general

I want to understand the tradeoffs of various REST API security strategies

Local Authentication

Who Is It?

How the App Is Organized

https://github.com/jzheaux/oreilly-spring-security-rest-apis

- Fork it
- Each branch is a chapter
- Each commit is the “answer” to an exercise
- Unit tests are there to check your solution

Exercise: Adding Spring Security

- Remove the `SecurityAutoConfiguration` exclusion
- Restart the app
- Run `Module1_Tests` - the first test should pass

- Report on 2-3 differences with how the app behaves now that Spring Security is added. For example:
  - Try http://localhost:8080/goals, before and after
  - Try http://localhost:8080/2l3kne23, before and after

This exercise should take between **3-5 minutes**

The Security Filter Chain

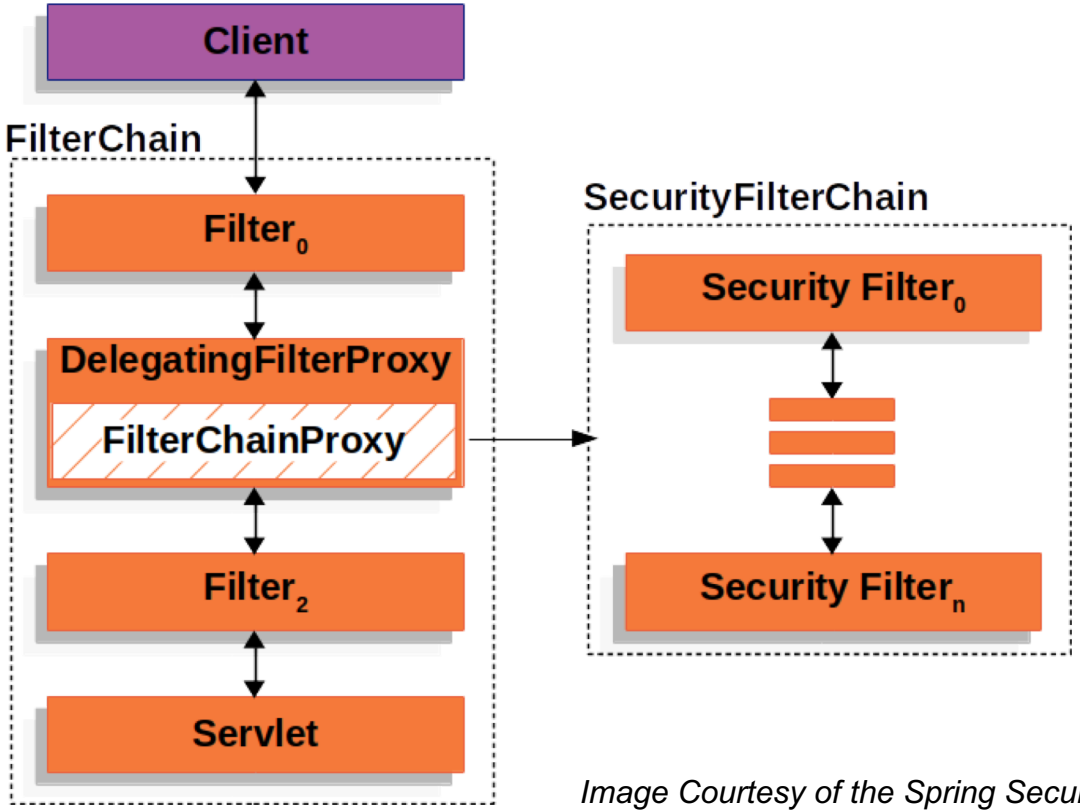
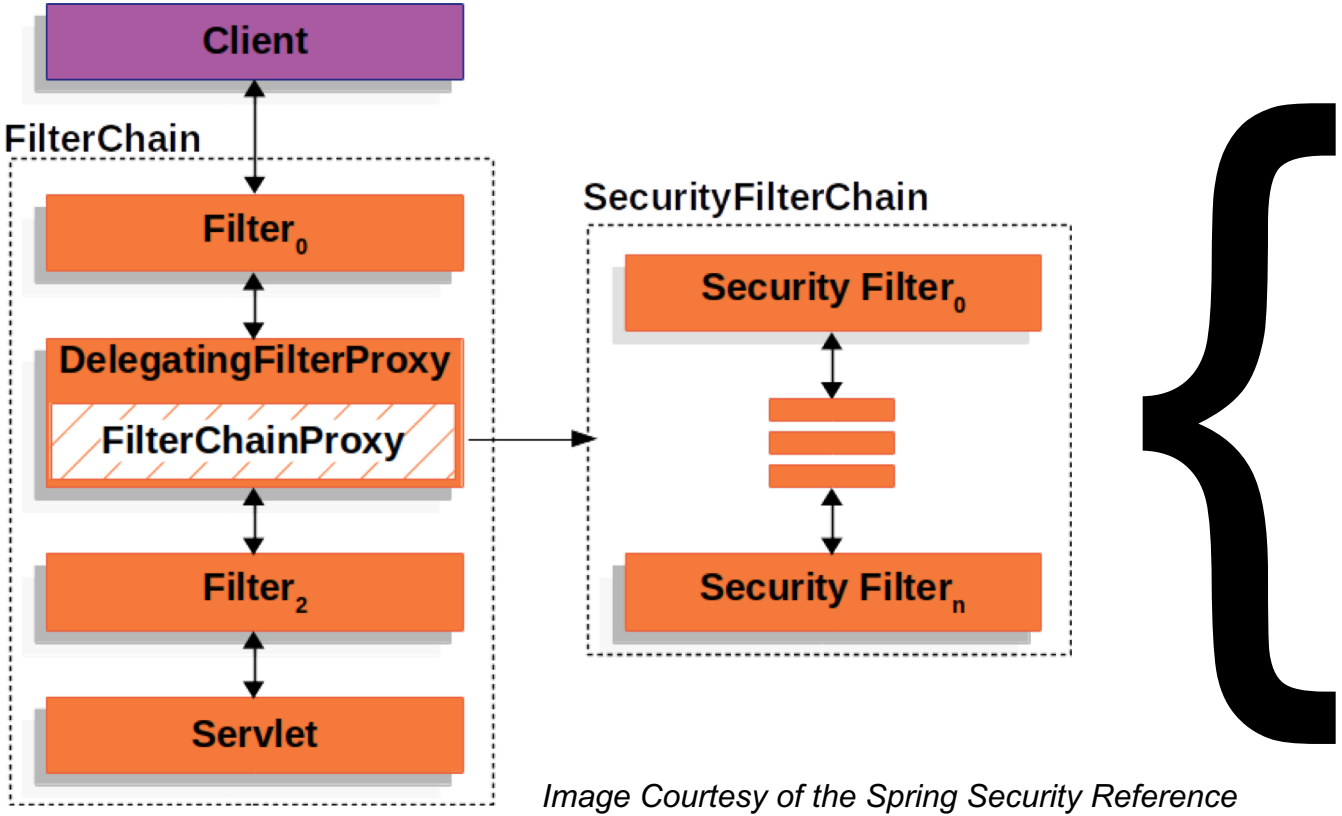


Image Courtesy of the Spring Security Reference

Filter Order



Defensive Filters

Authentication Filters

Authorization Filters

Image Courtesy of the Spring Security Reference

Basic Authentication Filter

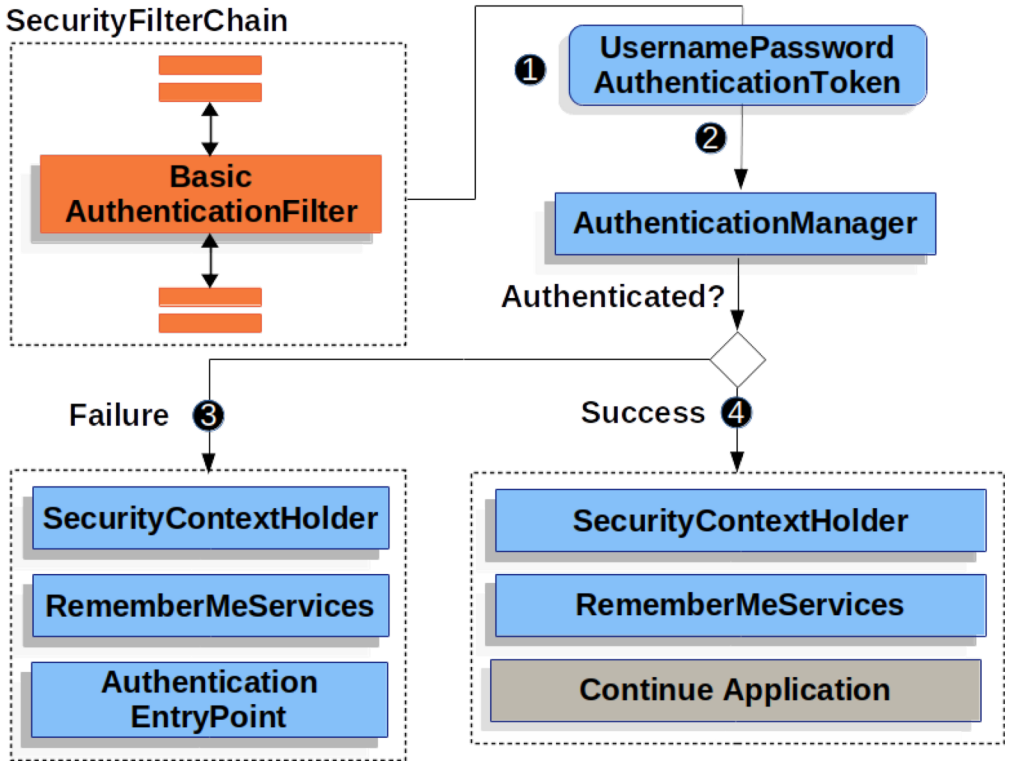


Image Courtesy of the Spring Security Reference

Exercise: Adding `UserDetailsService`

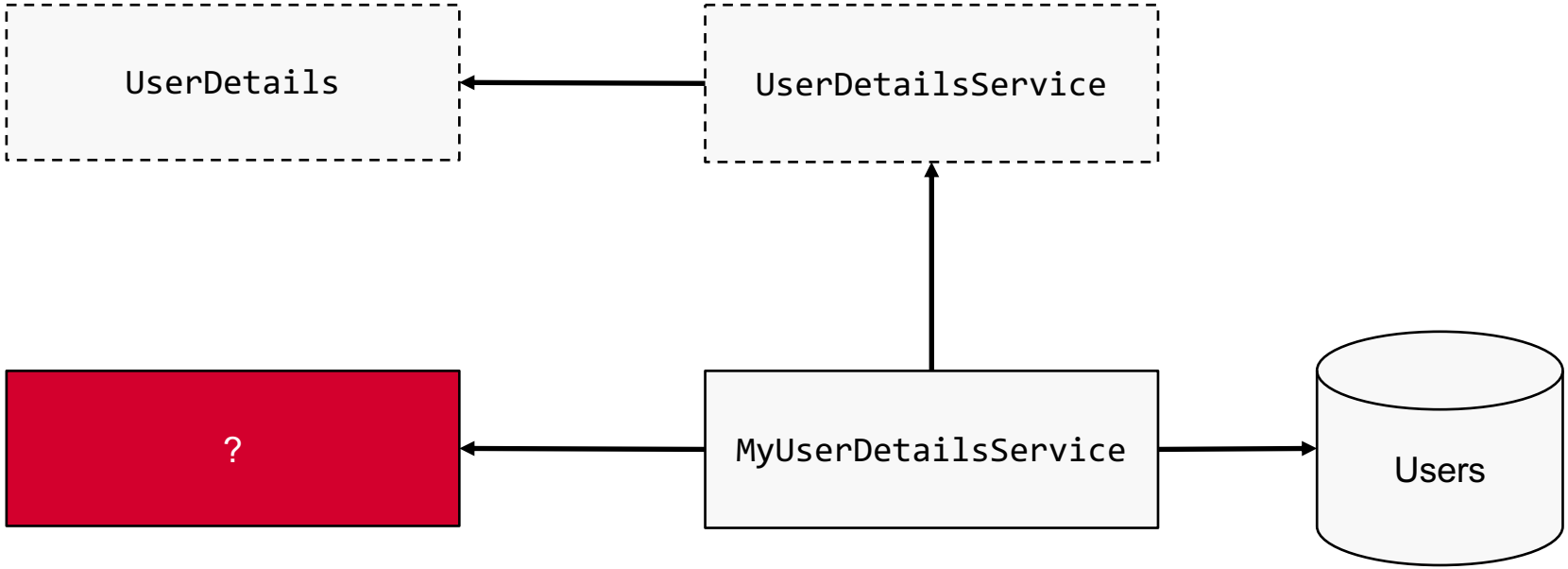
- Create a `@Configuration` class
- Create a Spring Security `User` using `user/password`
- Expose `InMemoryUserDetailsManager` as a `@Bean`

- Run `Module1_Tests` - the second test should pass
- Try the `/goals` endpoint
- Share one of the app's goals that you will definitely do! :)

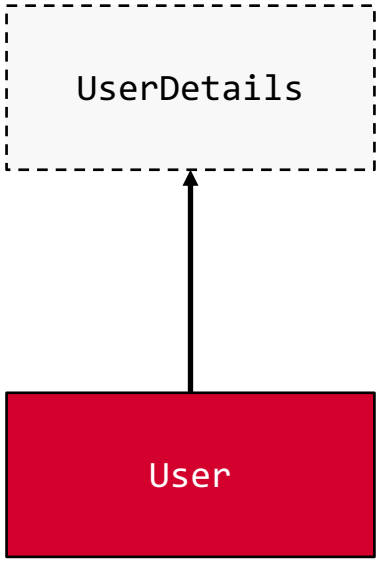
- *Stretch*: Configure with a hashed password

This exercise should take between **5-7 minutes**

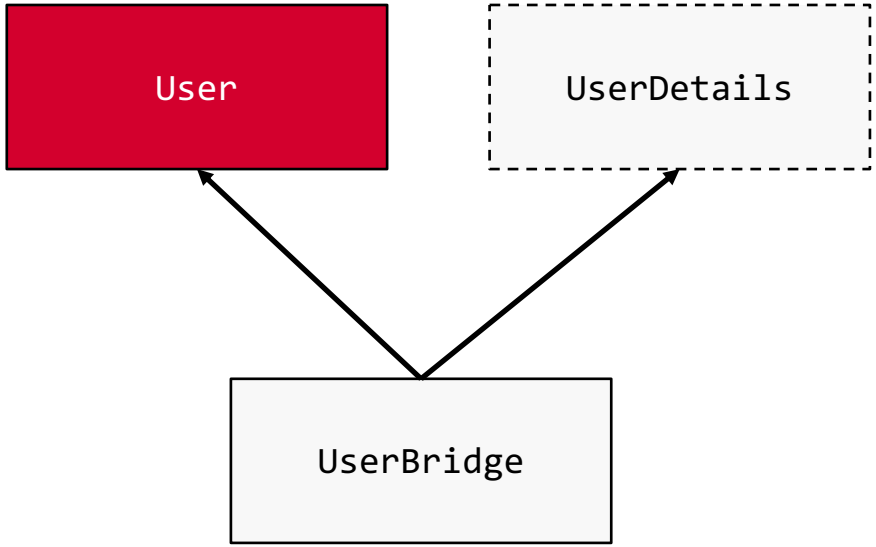
Customizing the Principal



Customizing the Principal



OR



Exercise: Customizing Principals

- Implement `UserDetailsService`, calling `UserRepository`
- Return an instance of a `private` inner bridge class
- Replace the exposed `UserDetailsService` with this one
- Add some users into `GoalsInitializer`

- Run `Module1_Tests` - the third test should pass
- Try the `/goals` endpoint
- Report on your new users' goals

This exercise should take between **7-10 minutes**

Exercise: Using Principals

- Update `GoalController#make` to lookup the username
- Use `@CurrentSecurityContext`

- Run `Module1_Tests` - the fourth test should pass
- Try the `POST /goal` endpoint
- Report on how the `POST /goal` endpoint behaves

This exercise should take between **3-5 minutes**

How secure is this?

Local Authorization

Is the Request Allowed?

Exercise: Authorizing Requests

- Add `@EnableGlobalMethodSecurity`
- Add `@PreAuthorize` to each controller method
  - What authority do you think each one should require?

- Run `Module2_Tests` - the first test should pass
- Try the `/goals` endpoint with the `haswrite` user and report
- Let's discuss: Which is better, filter- or method-based?
  - Stretch: Try adding filter-based to get a comparison

This exercise should take between **5-7 minutes**

Exercise: Insecure Direct Object References

- Add `@PostAuthorize` annotations confirm ownership

- Run `Module2_Tests` - the second test should pass
- Try looking up a goal that doesn't belong to your user
- Let's discuss: What's the appropriate response code?

This exercise should take between **5-7 minutes**

Exercise: Filtering Results

- Add `@PostFilter` to `/goals` to filter on ownership
- Add `@Query` to filter on ownership at the query level

- Run `Module2_Tests` - the third test should pass
- Try listing goals for different users
- Let's discuss: Where should the filter be placed?

This exercise should take between **5-7 minutes**

Exercise: Authorization Beans

- Create the `GoalAuthorizer` bean together with Josh
- Use the authorization bean in your annotations

- Run `Module2_Tests` - the fourth test should pass
- Try adding a record - why doesn't it work?
- Time for general Q&A and then a break!

This exercise should take between **10-12 minutes**

Ingress

Is the Request Safe?

The Most Popular Spring Security Hack

```
.csrf().disable()
```

Exercise: Configuring CORS

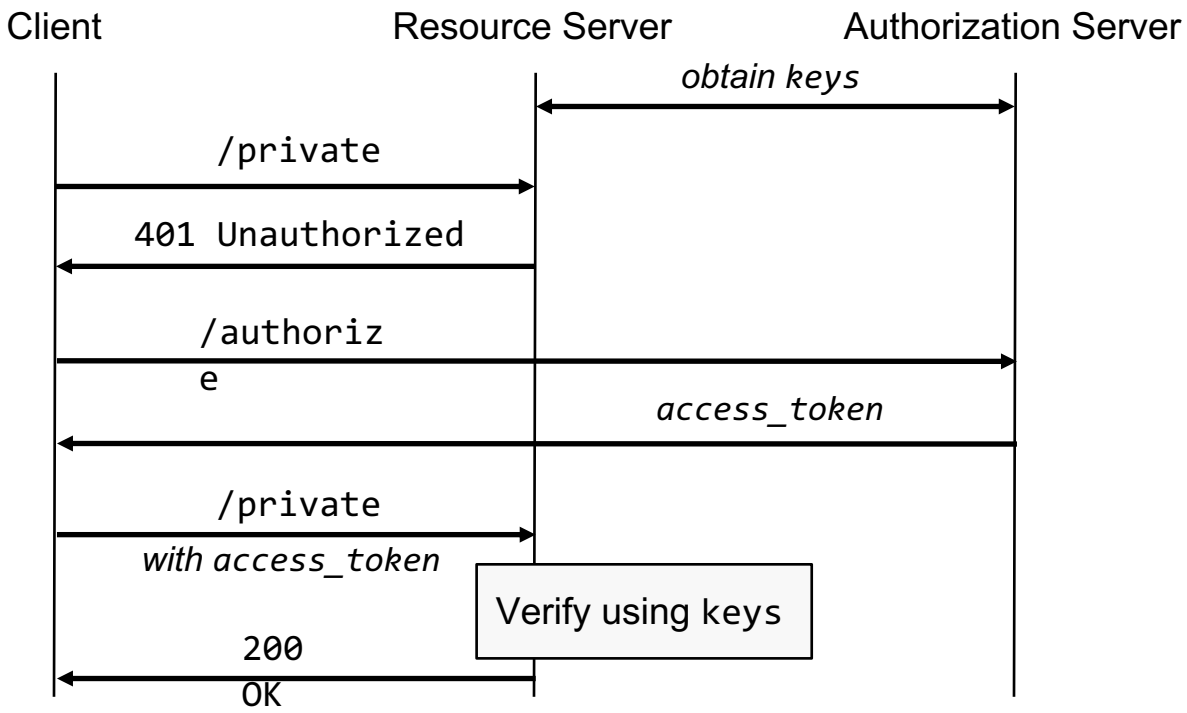
- Add the `@CrossOrigin` annotation to `/goals`
- Add `cors()` to the Spring Security DSL

- Run `Module3_Tests` - the two tests should pass
- Stand up the sample app and click the button to see the response
- Stand up the malicious app and see if you can add a goal
- Let's discuss: What are the security tradeoffs for turning on `withCredentials`?

Distributed Authorization with JWT

A Step Towards Security Convergence

JWT Authentication Flow



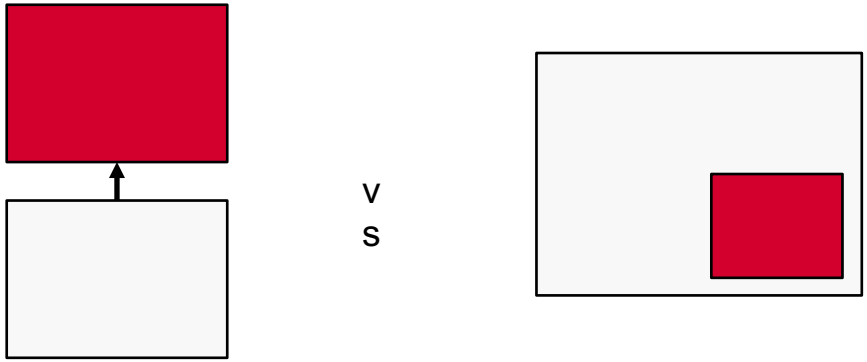
Exercise: Bearer Token Auth

- Add the `spring-security-starter-oauth2-resource-server` and `spring-security-oauth2-jose` dependencies
- Add `spring.security.oauth2.resourceserver.jwt.issuer-uri`
- Add `oauth2ResourceServer` to the DSL and specify `jwt`

- Run `Module4_Tests` - the first test should pass
- Stand up the authorization server
- Obtain a token using the `token-for` script
- Try the `/goals` endpoint with the token

This exercise should take between **5-7 minutes**

Pulse Check: Composition



- Use more than one component
- Don’t inherit unwanted behavior
- Avoid spending your inheritance

How are you representing scopes at your company?

Exercise: Canonicalize Authorities

- Modify `UserRepositoryJwtAuthenticationConverter` to turn the `scope` attribute into a list of `GrantedAuthority` instances
- Specify the converter in the Spring Security DSL

- Run `Module4_Tests` - the second and third tests should pass
- Try the `POST /goal` endpoint
  - Did it work? Why?

This exercise should take between **5-7 minutes**

Programmatic vs Declarative

Declarative

Decoupled from business logic

Logicless or low-logic

Framework-managed

Programmatic

Woven into business logic

The full Java language

You-managed

Exercise: Programmatic Security

- Modify `GoalController#read()` to check the `SecurityContextHolder` for the `user:read` authority.
- If present, add the user's full name to the result

- Run `Module4_Tests` - the fourth test should pass
- Try the `GET /read` endpoint to see the full name included
- Let's discuss: Is there a way to avoid programmatic security?

This exercise should take between **5-7 minutes**

Second Most Popular Spring Security (Testing) Hack

```
secure = false
```

Exercise: Testing

- Change the existing failing unit test by including the appropriate scope in the test configuration
- Add a test of your own

- The tests should pass
- Share what you tested on the discussion page

This exercise should take between **5-7 minutes**

Poll: Multi-tenancy

A user can log in to your app in multiple ways that all resolve to the same user (for example, Login in with Facebook)

A user can login and they are tied to a specific SaaS instance of your product (for example, a user logging in with me@nike.com would redirect them to your nike.yourapp.com instance)

A user can be logged into multiple instances of our app at once

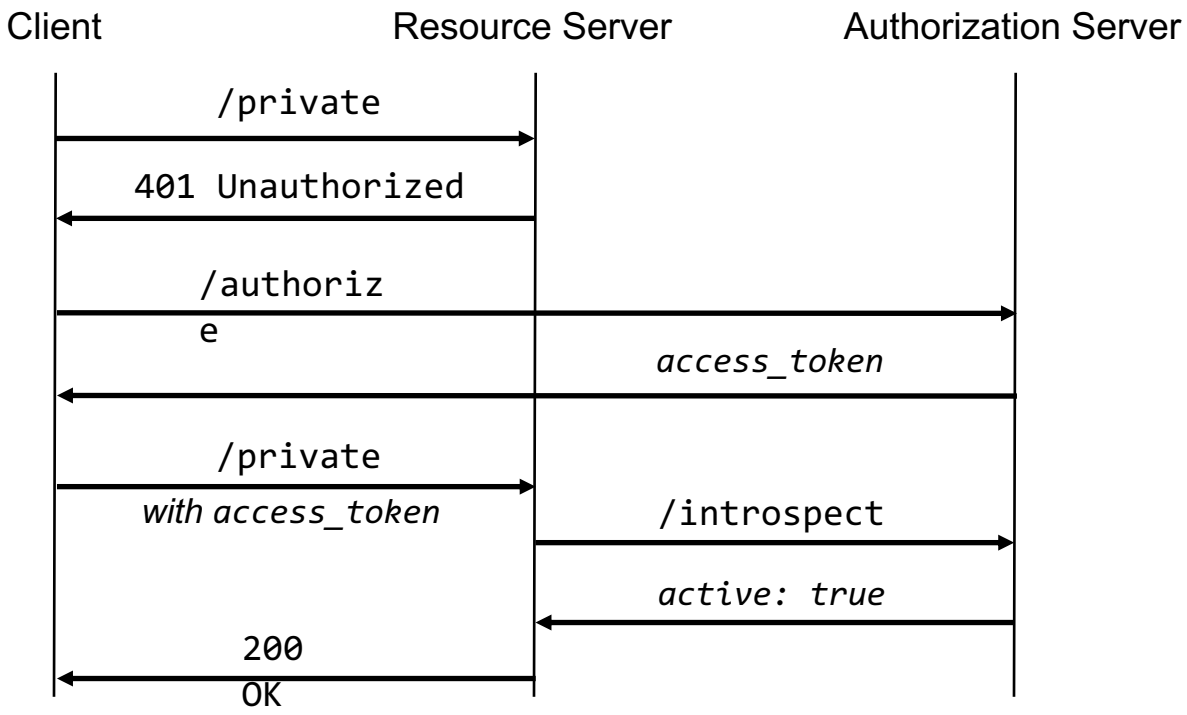
A tenant gets full control over the authentication experience of their user base

We don’t support multiple tenants at this time

Distributed Authorization with Opaque Tokens

JWT’s more secure, less performant cousin

Opaque Authentication Flow



JWT

By Value

Occasional Authorization Server calls

Client can see values

Can’t expire immediately

Opaque

By Reference

Frequent Authorization Server calls

Client can’t see values

Can expire immediately

Exercise: Opaque Tokens

- Add the `com.nimbusds:oidc-oauth2-sdk` dependency
- Add `.opaquetoken.introspection-uri`, `.client-id`, and `.client-secret`
- Change `oauth2ResourceServer` to specify `opaquetoken`

- Run `Module5_Tests` - the first test should pass
- Try the `/goals` endpoint with the token

This exercise should take between **5-7 minutes**

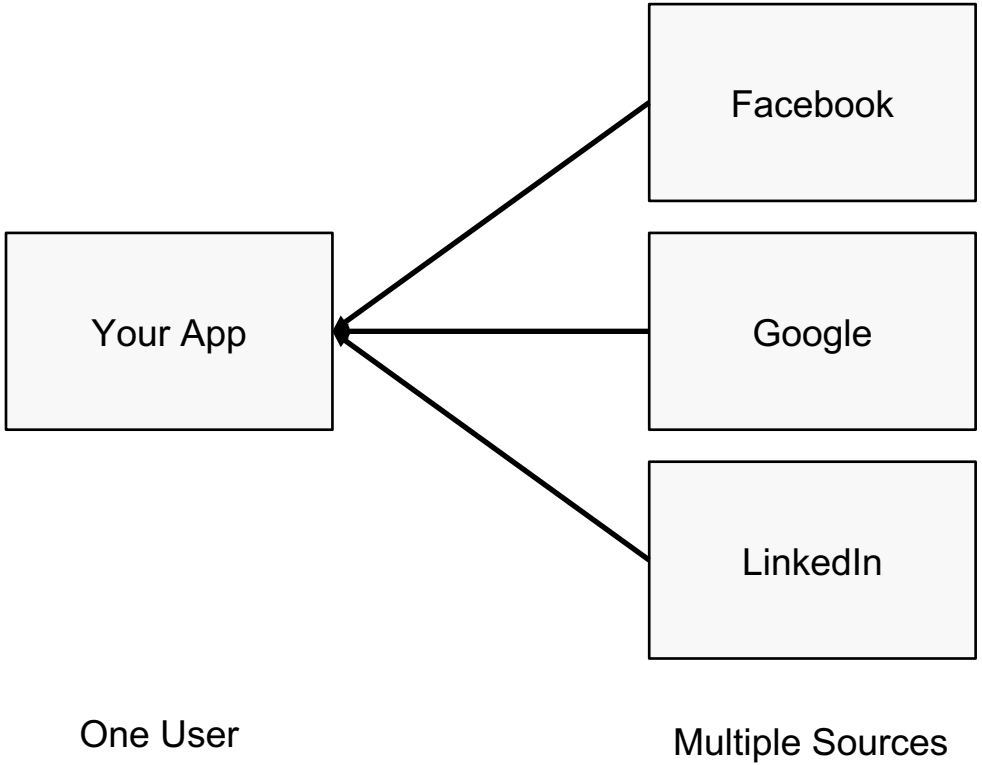
Exercise: Normalize Authorities

- Modify the `UserRepositoryOpaqueTokenIntrospector` add a `GrantedAuthority` called `goal:share` that's derived from whether or not the `User` has a premium subscription.

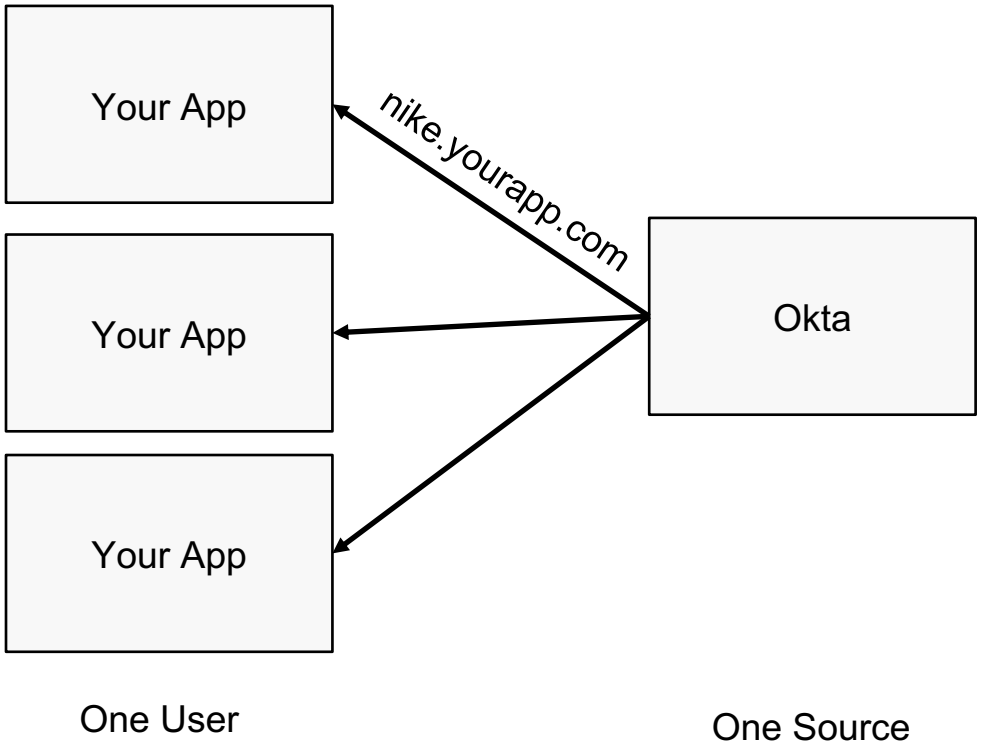
- Run `Module5_Tests` - the second and third tests should pass
- Try the `POST /share` endpoint to share a goal with another user

This exercise should take between **5-7 minutes**

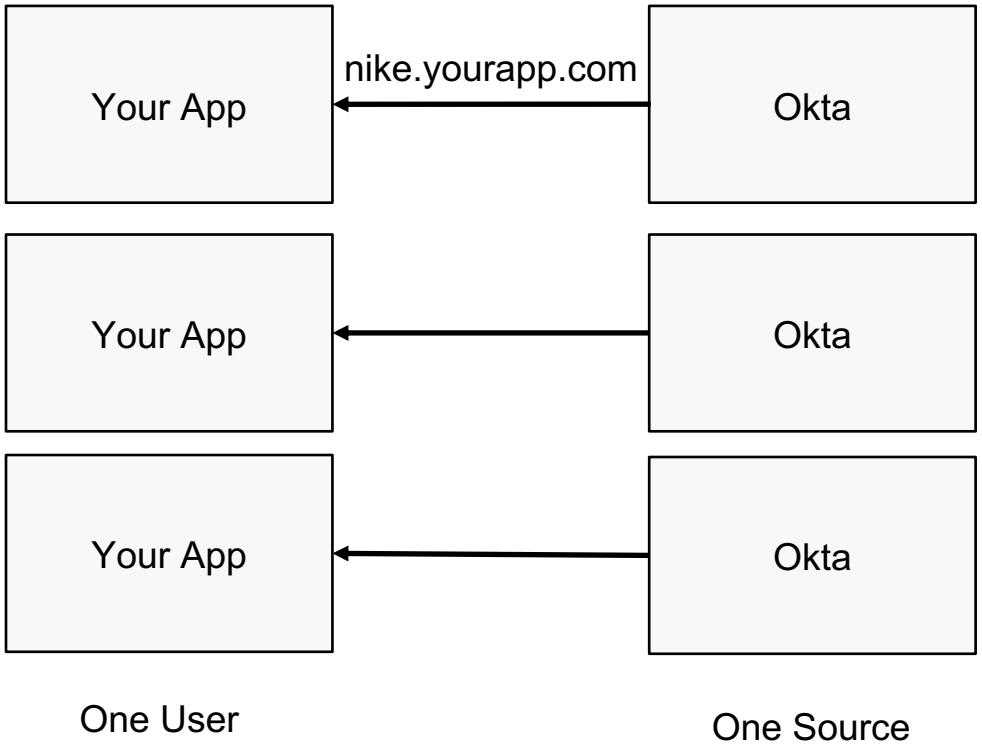
Identity Federation



SaaS


Your App
Your App
Your App
nike.yourapp.com
Okta
One User
One Source

SaaS + Whitelabeling



Exercise: Multi-tenancy

- Complete the `JwtOpaqueTokenAuthenticationManagerResolver` so it looks for a header called `tenant`. If the tenant is one, return the `jwtAuthenticationManager`. Otherwise, return the `opaqueTokenAuthenticationManager`.
- Change the DSL to use the `authenticationManagerResolver` instead of `jwt`.

- Run `Module5_Tests` - the fourth test should pass

This exercise should take between **5-7 minutes**

Egress

Collaborating with Other REST APIs

Browsers Propagate

Cookies

`Authorization: Basic` creds

But Not

`Authorization: Bearer` creds

Exercise: CORS

- Remove the `withCredentials` attribute from `@CorsMapping`

- Start up the front-end application and navigate to http://localhost:8081/bearer.htm to confirm that the application works
- Run `Module6_Tests` - the first test should pass

This exercise should take between **5-7 minutes**

Exercise: Passing the token

- Complete the `WebClient` configuration by including `ServletBearerExchangeFilterFunction` as a filter.
- In `GoalController`, replace the `UserRepository` dependency with `UserService`.
- Change the `read()` method to find the user's full name using `UserService`.

- Run `Module6_Tests` - the second test should pass

This exercise should take between **5-7 minutes**

What If I Need to Renew the Token?

```
ServletOAuth2AuthorizedClient
ExchangeFilterFunction
```

Thank you!

https://{gh}/jzheaux/oreilly-spring-securing-rest-apis
https://{gh}/spring-projects/spring-security
Josh Cummings - @jzheaux

Click to add slide title

- Click to edit master text styles
  - Second level
    - Third level
      - Fourth level

Click to add slide title for two-line title

- Click to edit master text styles
  - Second level
    - Third level
      - Fourth level

Click to add slide title

Click to edit subhead

- Click to edit master text styles
  - Second level
    - Third level
      - Fourth level

Click to add slide title

Click to edit subhead 1

- Edit master text styles
  - Second level
    - Third level
      - Fourth level
        - Fifth level

Click to edit subhead 2

- Edit master text styles
  - Second level
    - Third level
      - Fourth level
        - Fifth level

Click to add slide title

- Click to edit master text styles
  - Second level
    - Third level
      - Fourth level

Click to add slide title

- Click to edit master text styles
  - Second level
    - Third level
      - Fourth level

1. To add an image click on this red circle.
2. Look up. In the menu bar click **Replace image**.
3. Upload an image from your computer

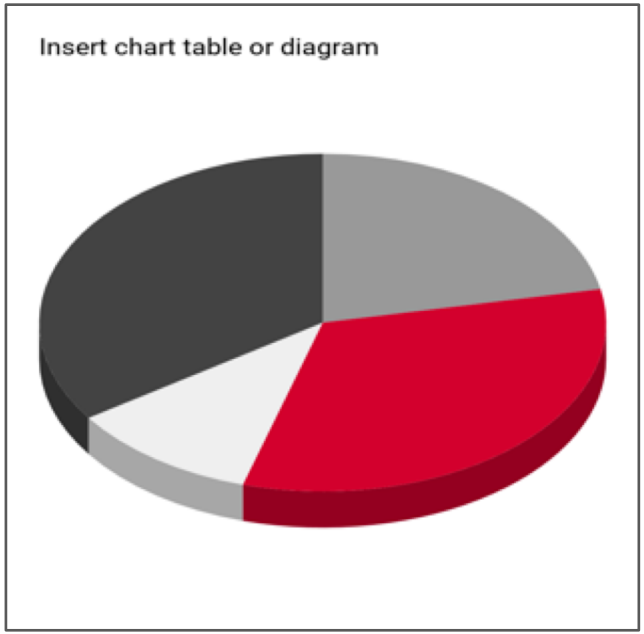
Click to add slide title

Text goes here

Click to add slide title

Text goes here

Click to add slide title



Title

Click to edit text here.

Title

Click to edit text here.

O'REILLY®

Thank you