1. Goals

Understand what tokens are, why JWT is a solid option and how to set it up with Spring Security.

2. Lesson Notes

Token Implementations

SAML (or the WS* space)

- XML based

- many encryption and signing options

- expressive but you need a pretty advanced XML stack

Simple Web Token

- joint venture between Microsoft, Google, Yahoo

- created as a direct reaction to making a much simpler version of SAML

- to simple, not enough cryptographic options (just symetric)

JWT (JSON Web Tokens)

- the idea is that you are representing the token using JSON (widely supported)

- symmetric and asymmetric signatures and encryption

- less options/flexibility than SAML but more than SWT

- JWT hit the sweet spot and became widely adopted pretty quickly

- JWT - an emerging protocol (very close to standardization)

JWT structure

- a JWT token has 2 parts:

1. Header

- metadata

- info about algos / keys used

2. Claims

- Reserved Claims (issuer , audience, issued at, expiration, subject, etc)

- Application specific Claims

JWT with Spring Security OAuth

For the Authorization Server:

- we’re defining the JwtAccessTokenConverter bean and the JwtTokenStore

- we’re also configuring the endpoint to use the new converter

Note that we're using symmetric signing - with a shared signing key.

For the Resource Server:

- we should define the converter here as well, using the same signing key

Note that we don’t have to because we’re actually sharing the same Spring context in this case. If the Authorization Server would have been a separate app - then we would have needed this converter, configured exactly the same as in the Resource Server.

3. Resources

- jwt.io

- Spring Security and Angular JS