1. Goal

Learn where URL level authorization is enough, where we need method level security and how to use it.

2. Lesson Notes

URL Security vs Method Level Security

URL level security is usually enough in MVC style of applications. For a REST API, where the same URL can represent multiple operations (based on the HTTP verb as well as other factors) - it's no longer enough.

Map URLs for Authorization

anyRequest

- maps to everything

antMatchers

- uses ant-style patterns to match requests by path

- **ex**:

```
antMatchers("/api/**")
antMatchers("/api/**")
```

regexMatchers

- uses a regex to match the requests by path

- **ex**:

```
regexMatchers("^/login.*").permitAll()
```

Authorize Mapped URLs

After the URLs are mapped, the fluent API can now configure their authorization:

- *anonymous*
- *authenticated*
- *fullyAuthenticated*
- *denyAll*
- *hasAnyAuthority / hasAnyRole / hasAuthority / hasRole / hasIp*
- *access*
- *permitAll*

One quick but important note about *anonymous*. Anonymous access on an URL means that **only an anonymous user will be able to access that URL**. If an authenticated user tries to access it - they'll get back a 403 Forbidden.

If you need to allow both anonymous and authenticated users access, simply use *permitAll*.

3. Resources

- Spring Security Reference - HTTP Security

- Spring Security Reference - Method Security