JAVA BACKEND DEVELOPMENT PROGRAM

Spring Security

OUTLINE

- Spring Security
- Core Component
- Spring Security Expressions
- Secure REST Service

SPRING SECURITY

- Authentication vs Authorization

- Authentication: Verify who a user is.

- Authorization: Verify what access does a user have.

SPRING SECURITY

- Spring Security is another module provided by Spring team
- Spring Security integrates with the Servlet Container by using a standard Servlet Filter.
  - This means it works with any application that runs in a Servlet Container.
  - More concretely, you do not need to use Spring in your Servlet-based application to take advantage of Spring Security.

SPRING SECURITY

- Spring Security Core Components for Authorization:
    - SecurityContextHolder — to provide access to the SecurityContext
    - SecurityContext — to hold the Authentication (and possibly request-specific security information).
    - Authentication — to represent an authenticated user
    - UserDetails — to provide the necessary information to build an Authentication object from your application's DAOs or other source of security data.
    - GrantedAuthority — to reflect the application-wide permissions granted to a principal.

SPRING SECURITY

- Spring Security Core Components for Authentication:
  - AuthenticationManager — to perform authentication process base on the supplied AuthenticationProvider
  - AuthenticationProvider — to tell AuthenticationManager how to authenticate a user
  - UserDetailsService — to create a UserDetails when passed in a String-based username (or certificate ID or the like).

JWT

- Json Web Token (JWT) — it is an open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object

- In its compact form, JSON Web Tokens consist of three parts separated by dots (.)
  - Header
  - Payload
  - Signature

JWT


```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```


HEADER: ALGORITHM & TOKEN TYPE


```
{
  "alg": "HS256",
  "typ": "JWT"
}
```


PAYLOAD: DATA


```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```


VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

SPRING SECURITY & JWT

- Now, how do they work together?
    - Before users can use a web service, they need to be authenticated
    - Users need to provide their username and password to the Authentication Server
    - Authentication Server then authenticates these information (can be done using Spring Security as well, but detail not mentioned here, demo code will be provided for your interest)
    - If provided username and password are correct, then an JWT will be generated using certain key and algorithm, containing user identification information and associated authorities

SPRING SECURITY & JWT

- After being authenticated, users can then use our web service
- When users send requests to our web service, the JWT need be included in every request (normally in the request header)
- When web service receive the requests, it needs to decrypt the JWT using the same key and algorithm to authenticate the JWT
- If the JWT is valid, then it means that the current user is already logged in
- Then we need to use Spring Security to store the current user's authorities, so that Spring Security can perform Authorization for us.
- But of course, we need to configure Spring Security in our web service

SPRING SECURITY & JWT

- Now, lets take a look at code that achieves what we just described

SPRING SECURITY EXPRESSIONS

- Ant pattern — Apache Ant provide a directory tree to represent the path

- There are some rules:
  - A single star (*) matches zero or more characters *within a path name*
  - A double star (**) matches zero or more characters *across directory levels*
  - A question mark (?) matches exactly one character within a path name

SPRING SECURITY EXPRESSIONS

- For example, we have following directory structure

    1. bar.txt
    2. src/bar.c
    3. src/baz.c
    4. src/test/bartest.c

- For following Patterns, what files do they match?

    - *.c
    - src/*.c
    - */*.c
    - **/*.c
    - **/bar.*

ANY QUESTIONS?