**Spring Security**

**A computer screen shot of a computer code

Description automatically generated**

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Suppose if you want give method/endpoint level access then use below.

Here Admin has access to get all products where user can see individual product while putting product\_id.

A screen shot of a computer code

Description automatically generated

Here you need to add below annotation (EnableMethodSecurity) for method level security.

A yellow marker on a white background

Description automatically generated

We should not hardcode username/pwd as above.

Follow the steps below to resolve this problem.

Create the below class to store user info in database.

Here we need to create jpaRepository and pass user info with the primary key.

A screen shot of a computer

Description automatically generated

A screen shot of a computer code

Description automatically generated

References: -

<https://www.youtube.com/watch?v=R76S0tfv36w>

https://www.youtube.com/watch?v=NcLtLZqGu2M

JWT Token

A screenshot of a computer

Description automatically generated

Here Application first time authenticate user and generate JWT token then further onwards use generated token and not asking username/pwd again and again.

Here are three component header, payload and verify signature.

All the component called as **claim** in jwt.

Here you need to add below jwt dependencies in pom.xml

A screenshot of a computer program

Description automatically generated

To generate secret, as below

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A close-up of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen shot of a program

Description automatically generated

A computer screen shot of a program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A diagram of a flowchart

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A diagram of a company

Description automatically generated

What's the difference between @Secured and @PreAuthorize in spring security

if you wanted to do something like access the method only if the user has Role1 and Role2 the you would have to use @PreAuthorize @PreAuthorize("hasRole('ROLE\_role1') and hasRole('ROLE\_role2')") Using @Secured({"role1", "role2"}) is treated as an OR

## Explain Spring Security Architecture using Spring Boot?

Let us understand how Spring Security Works.  
A diagram of a security system

Description automatically generated  
[Understand Spring Security Architecture and implement Spring Boot Security](https://www.javainuse.com/boot3/sec/3)

## How is Security mechanism implemented using Spring

Spring Security is a powerful and highly customizable authentication and access-control framework. It is the de-facto standard for securing Spring-based applications. Spring Security is a framework that focuses on providing both authentication and authorization to Java applications. Like all Spring projects, the real power of Spring Security is found in how easily it can be extended to meet custom requirements.  
**Spring makes use of the DelegatingFilterProxy for implementing security mechanisms.** It is a Proxy for standard Servlet Filter, delegating to a Spring-managed bean that implements the Filter interface. Its the starting point in the springSecurityFilterChain which instantiates the Spring Security filters according to the Spring configuration  
Some of the features of Spring Security are

* Comprehensive and extensible support for both Authentication and Authorization
* Protection against attacks like session fixation, clickjacking, cross site request forgery, etc
* Servlet API integration Optional integration with Spring Web MVC

## What is OAuth2 Authorization code grant type? How to implement it using Spring Boot Security?

OAuth (Open Authorization) is a simple way to publish and interact with protected data.  
It is an open standard for token-based authentication and authorization on the Internet. It allows an end user's account information to be used by third-party services, such as Facebook, without exposing the user's password.  
The OAuth specification describes five grants for acquiring an access token:

* Authorization code grant
* Implicit grant
* Resource owner credentials grant
* Client credentials grant
* Refresh token grant

Consider the use case of Quora. Go to Quora.com.  
If you are a new user you need to signup. You can signup using google or facebook account. When doing so you are authorizing Google or Facebook to allow quora to access you profile info with Quora. **This authorizing is done using OAuth**. Here you have in no way shared your credentials with Quora.  
[Understanding What Is OAuth2](https://www.javainuse.com/spring/spring-boot-oauth-introduction)  
[Spring Boot OAuth2 Part 1 - Getting The Authorization Code](https://www.javainuse.com/spring/spring-boot-oauth-authorization-code)  
[Spring Boot OAuth2 Part 2 - Getting The Access Token And Using it to fetch data.](https://www.javainuse.com/spring/spring-boot-oauth-access-token)

#### **19. How would you implement JWT Authentication in Spring Security?**

To implement JWT Authentication in Spring Security, start by adding the necessary dependencies to your build file. Then, create a JwtTokenProvider class that will generate and validate tokens. This class should include methods for creating tokens, setting expiration times, and validating tokens.

Next, create an implementation of UserDetailsService to load user-related data. It should override the ‘loadUserByUsername’ method.

Then, configure Spring Security to use JWT authentication. Create a SecurityConfig class extending WebSecurityConfigurerAdapter. Override ‘configure(HttpSecurity http)’ to specify security measures. Use ‘.antMatchers()’ to set endpoints, ‘.authenticated()’ to require all requests to be authenticated, and ‘.sessionManagement()’ to manage sessions.

Finally, add filters for login and token processing. The login filter authenticates users and generates a token. The token processing filter validates tokens from incoming requests.

## 11. What are some security annotations that can involve Spring Expression Language (SpEL)?

Spring Security uses SpEL for expression support. Expressions are evaluated with a root object as part of the evaluation context. The following annotations that allow the expression attributes to use authorization checks:

* @PreAuthorize: This annotation is used to specify a security constraint that must be satisfied before a method is invoked. The constraint is specified as an expression that is evaluated against the security context.
* @PostAuthorize: This annotation is similar to @PreAuthorize, but the expression is evaluated after the method has been invoked.
* @PreFilter: We use this annotation to specify a security constraint that must be satisfied before a list of elements is processed. The constraint is specified as an expression that is evaluated against each element in the list.
* @PostFilter: This annotation is the opposite of @PreFilter because the expression is evaluated after the list has been processed.
* @Secured: This annotation is used to specify a list of security roles that are allowed to access a method or class. The roles are specified as strings, and the user must have at least one of the specified roles to be granted access.

## 12. What are some important filter classes for Spring Security?

The Spring Security framework provides several filters that can be used to secure web applications. Some important filter classes include:

* **UsernamePasswordAuthenticationFilter:** This filter is used to authenticate a user using a username and password.
* **BasicAuthenticationFilter:** This filter is used to authenticate a user using basic authentication.
* **SessionManagementFilter:** We use this to manage user sessions and prevent session hijacking.
* **SecurityContextPersistenceFilter:** This filter is used to store the security context of a user across multiple requests.
* **ExceptionTranslationFilter:** This filter is used to handle exceptions thrown during the security process and convert them into HTTP responses.
* **FilterSecurityInterceptor:** This filter is used to enforce security constraints on HTTP requests based on the configured security rules.