**Spring Security**

1. **Spring security: Authentication và Authorization**
2. **Spring security là gì?**

- Một tính năng cốt lõi của spring framework: quản lý phân quyền và xác thực người dùng, hỗ trợ các tiêu chuẩn và giao thức bảo mật.

- Tính năng nâng cao: CSRF protection, quản lý phiên, mã hóa mật khẩu,...

- Cơ chế hoạt động:

* Theo mô hình client-server.
* Dựa trên cơ chế lọc và sự kiện để can thiệp quá trình xử lý yêu cầu và phản hồi.

- Thành phần cơ bản: Authentication, Authorization, Authentication Provider.

1. **Authentication trong Spring security**

- Quá trình xác thực xem người dùng là ai.

- Authentication dựa trên các thông tin nhận dạng và thông tin bí mật. Ví dụ: tài khoản và mật khẩu, vân tay,...

- Spring security hỗ trợ xác thực thông qua:

* Form/Trang đăng nhập
* Các header authorization
* Các giá trị được máy khách cung cấp

- Spring security hỗ trợ cả trạng thái và phi trạng thái.

* **Trạng thái (Stateful)**: Lưu thông tin xác thực trong một phiên trên máy chủ. Máy chủ kiểm tra phiên hiện tại để xác định danh tính và quyền hạn mà không yêu cầu đăng nhập lại.
* **Phi trạng thái (Stateless)**: Máy chủ không lưu bất kỳ thông tin xác thực nào của người dùng mà chỉ sử dụng các mã token đã được đăng ký để xác thực.

1. **Authorization trong Spring security**

- Quá trình xác thực xem người dùng được làm gì hoặc được truy cập loại tài nguyên nào.

- Authorization dựa trên các thông tin về vai trò, nhóm, quyền hạn, chính sách.

- Spring security hỗ trợ xác thực bằng cách sử dụng các annotation hoặc XML. Các thành phần gồm:

* AccessDecisionManager
* AccessDecisionVoter
* SecurityExpressionHandler

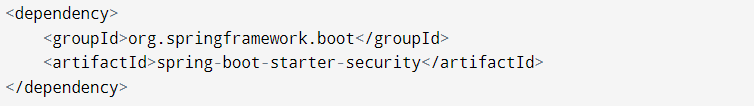
1. **Authentication Provider**

- Thành phần trong Spring security chịu trách nhiệm xác minh thông tin xác thực của người dùng.

- Mỗi Authentication Provider chỉ hỗ trợ một loại authentication cụ thể như: UsernamePasswordAuthenticationToken, JwtAuthenticationToken, PreAuthenticatedAuthenticationToken,...

1. **Cấu hình Spring security, quản lý đăng nhập và phân quyền**
2. **Cấu hình Spring security**

- Thêm các phụ thuộc cần thiết: **spring-boot-starter-security:**

* Với gradle: *implementation 'org.springframework.boot:spring-boot-starter-security'*
* Với maven: 

- Lớp cấu hình Spring security:

* Một số annotation: @Configuration, @EnableWebSecurity, @Bean
* SecurityFilterChain: định nghĩa và quản lý chuỗi bộ lọc bảo mật (xác thực, phân quyền, kiểm tra token,...)



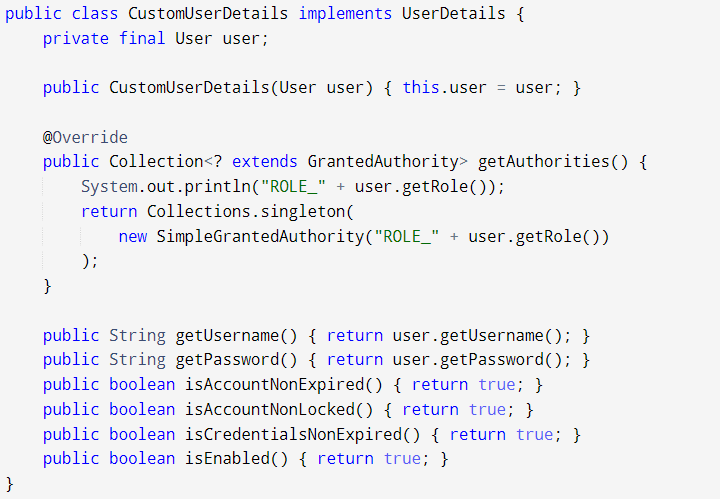
1. **Đăng nhập, phân quyền với JPA Hibernate**

- Khi có request từ client gửi đến server, Spring security chặn các request và tiến hành xác thực.

- **AuthenticationManager c**hịu trách nhiệm xử lý quá trình xác thực (authentication) người dùng.

- Spring Boot tự động cung cấp Bean này cho quá trình xác thực, hoặc có thể tùy chỉnh trong class cấu hình bảo mật: 

- AuthenticationManager cung cấp thông tin xác thực cho **AuthenticationProvider**, AuthenticationProvider gọi đến các phương thức để lấy thông tin từ cơ sở dữ liệu: 

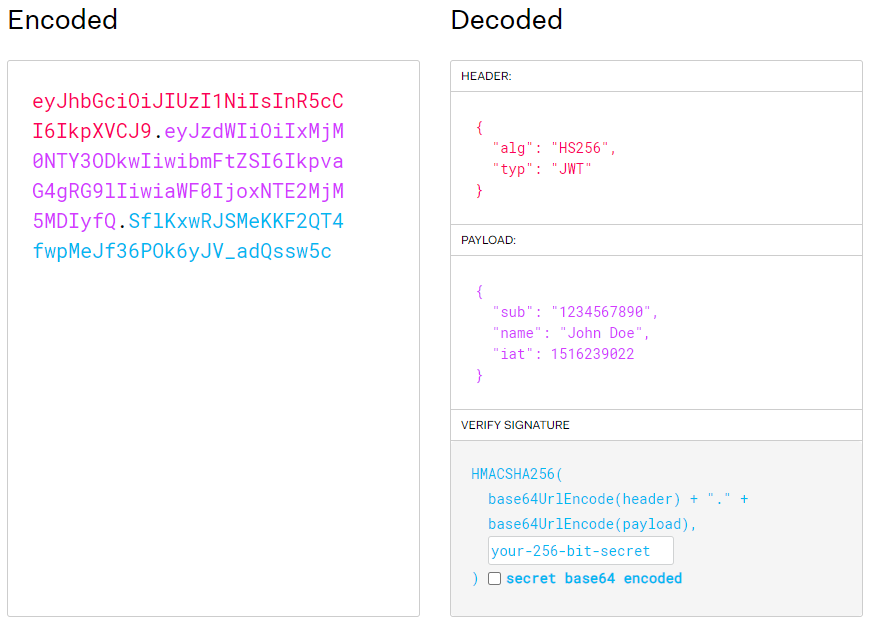
- Triển khai một lớp từ interface UserDetails mà Spring security cung cấp, cho phép lấy ra các thông tin như tên đăng nhập, mật khẩu đã mã hóa, các quyền 

1. **JWT - Json Web Token**
2. **JWT là gì?**
3. **JWT và cấu trúc của nó?**

- Giống như một hộ chiếu kỹ thuật số, đảm bảo dữ liệu được xác minh và đáng tin cậy. Tuân theo tiêu chuẩn mã nguồn mở RFC 7519.

- Gồm 3 phần: header, payload và signature. Được phân tách bởi dấu chấm.

* **Header**: Thông tin về loại token (JWT) và thuật toán mã hóa (HMAC SHA256, RSA,...).
* **Payload**: Thông tin mà token mang theo, có thể tùy chỉnh, nhưng cũng có một số thuộc tính tiêu chuẩn (claims) như iss (issuer), sub (subject), exp (expiration),...
* **Signature**: Tạo ra bằng cách mã hóa header và payload với thuật toán đã chỉ định và một khóa bí mật. Điều này giúp xác thực rằng token không bị thay đổi.

- Ví dụ (thuật toán HS256): 

- Claims: Các thông tin được truyền tải trong payload của JWT:

* **Registered claims**: Các claims đã được định nghĩa sẵn như iss (issuer - bên cung cấp, phát hành JWT), exp (expiration time - thời hạn của JWT), sub (subject - Chủ thể của JWT), aud (audience - người nhận thông tin),...
* **Public claims**: Các claims được định nghĩa bởi người dùng JWT.
* **Private claims**: Các claims do các bên sử dụng tùy chỉnh cho mục đích riêng.

1. **Các loại JWT**

- **Access Token**: Token dùng để xác thực và ủy quyền cho người dùng khi truy cập vào các API hoặc dịch vụ.

- **Refresh Token**: Token dùng để lấy một access token mới mà không cần đăng nhập lại.

1. **Cách thức hoạt động**

- **Xác thực người dùng**: Người dùng đăng nhập, server xác thực thông tin.

- **Tạo JWT**: Nếu xác thực thành công, server tạo một JWT bao gồm thông tin người dùng và gửi lại cho client (token).

- **Lưu trữ JWT**: Client lưu trữ token (local, session hoặc cookies).

- **Gửi kèm token trong request**: Client cần gửi kèm token khi truy cập vào các tài nguyên được bảo vệ.

**- Xác thực JWT**: Server nhận được yêu cầu và xác thực JWT. Nếu token hợp lệ, yêu cầu được xử lý, nếu không, yêu cầu bị từ chối.

1. **Quản lý JWT**

- Thời gian sống: Thiết lập giới hạn thời gian của token để tăng tính bảo mật.

- Thu hồi token: Cần có cơ chế thu hồi token khi cần thiết (người dùng đăng xuất, phiên làm việc hết hạn,...).

1. **Xác thực và bảo mật API với JWT**

-