**Spring Security là gì ?**

Cung cấp các service bảo mật toàn diện cho các ứng dụng phần mềm doanh nghiệp dựa trên Java EE.

Spring Secutiry hỗ trợ một loạt các mô hình xác thực . Hầu hết các mô hình xác thực do các bên thứ 3 cung cấp, hoặc được phát triển bởi các cơ quan tiêu chuẩn có liene quan như Internet Task Force.

Ngoài ra, Spring Secutiry còn cung cấp các tính năng xác thực riêng, Nó hỗ trợ tích hợp xác thực với khoảng gần 30 công nghệ .

* Việc tích hợp nhanh Spring Security giúp ko cần phải thay đổi environment

**Spring Security**

Spring Security là một dự án nổi bật trong hệ sinh thái Spring. Spring Security cung cấp các dịch vụ bảo mật toàn diện cho các ứng dụng doanh nghiệp có nền tảng Java EE.

Spring Security cung cấp 2 cơ chế cơ bản:

• Authentication (xác thực): là tiến trình thiết lập một principal. Principal có thể hiểu là một người, hoặc

một thiết bị, hoặc một hệ thống nào đó có thể thực hiện một hành động trong ứng dụng của bạn.

• Authorization (phân quyền) hay Access-control: là tiến trình quyết định xem một principal có được

phép thực hiện một hành động trong ứng dụng của bạn hay không. Trước khi diễn tiến tới   
Authorization, principal cần phải được thiết lập bởi Authentication.

**Feature**

* Hỗ trợ toàn diện và mở rộng cho cả Xác thực và Ủy quyền
* Bảo vệ chống lại các cuộc tấn công như phiên cố định, clickjacking, cross yêu cầu trang web giả mạo, vv
* Servlet API integration
* Optional integration with Spring Web MVC

**Codebase** được chia làm các thành phần riêng biệt tách rõ các vùng chức năng khác nhau và phụ thuộc vào bên thứ 3. Hỗ trợ các ứng dụng độc lập, remote client, method security, cấp phép người dùng JDBC. Chứa các gói cấp cao nhất.

* org.springframework.security.core
* org.springframework.security.access
* org.springframework.security.authentication
* org.springframework.security.provisioning

**Các module security cho Spring Security**

* spring-security-remoting.jar: Cung cấp tích hợp với Spring Remoting. Ứng dục khách từ xa sử dụng tính năng Remote
* Web - spring-security-web.jar : Chứa filter và lien quan đến web-security. Bất kỳ thứ gì phụ vào API Servlet thì sẽ cần xác thực web và kiểm soát truy cập dựa trên URL
* Config - spring-security-config.jar : Chứa security namespace parsing code. Sử dụng Spring Security XML để cấu hình
* LDAP - spring-security-ldap.jar : Authentication & provisioning code ( mã cấp phép ) . Sử dụng khi cần xác thực LDAP hoặc quản lý các mục người dùng LDAP
* ACL - spring-security-acl.jar : Triển khai đối tượng ACL. Sử dụng để bảo mật cho instance object với ứng dụng.
* CAS - spring-security-cas.jar : Tích hợp CAS client integration. Nếu muốn sử dụng xác thực web Spring security với 1 máy chủ đăng nhập 1 lần CAS
* OpenID - spring-security-openid.jar : Hỗ trợ xác thực web OpenID xử dụng dể xác thực người dùng dựa trên máy chủ OpenID bên ngoài

**Summary**

* SecurityContextHolder, để cung cấp quyền truy cập vào SecurityContext.
* SecurityContext, để giữ xác thực và có thể yêu cầu thông tin bảo mật cụ thể.
* Authentication, để đại diện cho thành phần theo cách thức cụ thể của Spring Security.
* GrantedAuthority, để phản ánh các quyền trên toàn ứng dụng được cấp cho thành phần đó.
* UserDetails, cung cấp các thông tin cần thiết để xây dựng 1 đối tượng Authentication từ các DAO của ứng dụng hoặc nguồn dữ liệu bảo mật nào khác
* UserDetailsService, để tạo UserDetails khi được truyền trong một tên người dùng dựa trên chuỗi (hoặc ID chứng chỉ hoặc tương tự).

**Security, SecurityContext và Authentication**

**SecurityContext** là interface cốt lõi của Spring Security, lưu trữ tất cả các chi tiết liên quan đến bảo mật trong ứng dụng. Khi chúng ta kích hoạt Spring Security trong ứng dụng thì SecurityContext cũng sẽ được kích hoạt theo.

Chúng ta sẽ không truy cập trực tiếp vào SecurityContext, thay vào đó sẽ sử dụng lớp **SecurityContextHolder**. Lớp này lưu trữ security context hiện tại của ứng dụng, bao gồm chi tiết của principal đang tương tác với ứng dụng. Spring Security sẽ dùng một đối tượng **Authentication** để biểu diễn thông tin này. Đoạn code dưới đây sẽ giúp chúng ta lấy được username của principal đã được xác thực (username ở đây ta nên hiểu là username trong cặp username - password mà người dùng nhập vào khi đăng nhập):

Object principal = SecurityContextHolder.getContext().getAuthentication().getPrincipal();

if (principal instanceof UserDetails) {

String username = ((UserDetails) principal).getUsername();

} else {

String username = principal.toString();

}

**UserDetails**

là một interface cốt lõi của Spring Security. Nó đại diện cho một principal nhưng theo một cách mở rộng và cụ thể hơn. Vậy **UserDetails** cung cấp cho ta những thông tin gì? **UserDetails** bao gồm các method sau:

* getAuthorities(): trả về danh sách các quyền của người dùng
* getPassword(): trả về password đã dùng trong qúa trình xác thực
* getUsername(): trả về username đã dùng trong qúa trình xác thực
* isAccountNonExpired(): trả về true nếu tài khoản của người dùng chưa hết hạn
* isAccountNonLocked(): trả về true nếu người dùng chưa bị khóa
* isCredentialsNonExpired(): trả về true nếu chứng thực (mật khẩu) của người dùng chưa hết hạn
* isEnabled(): trả về true nếu người dùng đã được kích hoạt

Chúng ta có thể thấy **UserDetails** mới chỉ cung cấp các phương thức để truy cập các thông tin cơ bản của người dùng. Để mở rộng thêm các thông tin, chúng ta sẽ tạo một lớp **CustomUserDetails** implements org.springframework.security.userdetails.**UserDetails** (tên lớp là tùy ý, bạn đặt tên thế nào cũng được).

Câu hỏi tiếp theo đặt ra là ta sẽ tạo implementation của UserDetails ở đâu trong ứng dụng? Câu trả lời là ta sẽ dùng **UserDetailsService**. UserDetailsService là một interface có duy nhất một phương thức:

UserDetails **loadUserByUsername**(String username) throws UsernameNotFoundException;

Tham số truyền vào chỉ gồm có username của người dùng. Ta sẽ tìm kiếm trong CSDL, record thỏa mãn username. Nếu không tìm thấy, ta sẽ ném ra ngoại lệ UsernameNotFoundException.

Phương thức loadUserByUsername() sẽ trả về một implementation của UserDetails. Implementation ở đây có thể là:

* org.springframework.security.core.userdetails.User
* CustomUserDetails implements org.springframework.security.userdetails.UserDetailsmà mình đã nói ở trên

Nhiệm vụ của chúng ta là cần phải tạo một lớp UserDetailsServiceImpl implements UserDetailsService.

#### GrantedAuthority

Ở phần trên, mình đã đề cập đến phương thức getAuthorities(). Phưong thức này sẽ trả về một tập hợp các đối tượng **GrantedAuthority**. Một GrantedAuthority là một quyền được ban cho principal. Các quyền đều có tiền tố là **ROLE\_**, ví dụ như ROLE\_ADMIN, ROLE\_MEMBER ...