**Authentication**

-Là quá trình xác thực vào 1 hệ thống xem thông tin tài khoản đó có đúng không để truy cập vào hệ thống.

**Authorization**

-Là quá trình sau khi đã xác thực vào hệ thống thì nó sẽ xem thông tin này, người dùng này có những quyền hạn gì , và được phép làm những việc gì trong hệ thống.

**WorkFlow**

B1: Đầu tiên là request của người dùng truy cập vào hệ thống.

B2: Request đó sẽ đi qua các list filter , đầu tiên là FilterChainProxy là Filter đầu tiên trong các chuỗi filter và chịu trách nghiệm xem yêu cầu này cần được xác thực phân quyền hay không(để dùng để xác thực).

B3: Nếu mà yêu cầu đó được xác thực thì nó sẽ định tuyến đến các filter khác để xác thực Authentication như UsernamePasswordAuthentication hoặc JwtAuthenticationFilter -> Mục đích xem thông tin đăng nhập có đúng hay không.

B4:Phân quyền (Authorization) : Nếu đăng nhập là hợp lệ thì sẽ định tuyến đến các filter phân quyền trẳng hạn như FilterSecurityInterceptor ->Mục đích xem với cái thông tin mà người dùng đưa thì sẽ được thực hiện những tác vụ nào trong hệ thống.

B5:Xử lý yêu cầu phản hồi : Nếu xác thực và phân quyền thành công thì sẽ chuyền đến các thành phần trong ứng dụng để xử lý –> Sau khi xử lý các phản hồi được tạo ra và gửi lại cho người dùng thông qua trình duyệt.

B6:Quản lý phiên bảo mật: quản lý thông tin xác thực của người dùng trong một phiên làm việc và duy trì nó.

**List Filter**

**-DelegatingFilterProxy:**

+)Là 1 filter của spring Framework được xử dụng đê tích hợp các filter không phải là Spring-manager(filter không được quản lý bởi IoC Container) vào 1 ứng dụng Spring.

+)Khi sử dụng DelegatingFilterProxy ta có thể khai báo trong cấu hình của Spring và Spring sẽ quản lý nó như 1 bean và duy trì vòng đời của bean đó.

+)Cách hoạt đông: +) Trong cấu hình của Spring sẽ định nghĩa một “DelegatingFilterProxy” và cung cấp tên của Filter mà nó sẽ đại diện để sử dụng.

+)Khi ứng dụng khởi đổng Spring sẽ tạo ra một instace của “DelegatingFilterProxy” và gắn nó và chuỗi FilterChain.

+)khi một yêu cầu đến,“DelegatingFilterProxy” sẽ chuyển nó đến Filter thực sự mà nó đại diện bằng cách sử dụng ApplicationContext để lấy bean Filter tương ứng.

+)Filter thực xự sẽ xử lý.

**-FilterChainProxy:**

+)Là 1 filter đặc biệt mà thường được sử dụng để điều hướng các yêu cầu HTTP qua một chuỗi các filter khác nhau.Nó được sử dụng để tổ chức và quản lý các bước filter một cách hiệu quả.

+)Nó hoạt động bằng cách định nghĩa một hoặc nhiều chuỗi filter,một chuỗi chứa một tập hợp các filter cụ thể được áp dụng tuần tự khi một yêu cầu HTTP đến .Một chuỗi filter thường đại diện cho một loại yêu cầu cụ thể hoặc một tập hợp các quy tắc bảo mật.

+)Khi một yêu cầu HTTP đến, FilterChainProxy sẽ xác định chuỗi filter phù hợp dựa treenc ác quy tắc cấu hình và sau đó chuyển yêu cầu qua các filter trong chuỗi này.Các filter này sẽ xử lý yêu cầu ,thực hiện các chức năng như xác thực người dùng, kiểm tra quyền truy cập và bảo vệ tài nguyên.

**-SecurityContextHolder:**

+)Là 1 filter dùng để lưu trữ thông tin người dùng sau khi đang xác thực trong quá trình thực thi các yêu cầu Http.

+)Thường được sử dụng truy cập thông tin xác thực của người dùng từ bất kỳ nơi nào trong hệ thống.

+)Nó có 2 phương thức phổ biến là:

+)GetContext():Trả về đối tượng SecurityContext hiện tại chứa thông tin về người dùng đang xác thực và các quyền truy cập của họ.

+)SetContext():dùng để thiết lập 1 đối tượng xác thực mới, nhưng nó thường được sử dụng tự động bởi Spring Security.

**JWT(Json Web Token)**

-Nó là 1 phương tiện tiêu chuẩn để truyền thông tin xác thực giữa các bên dưới dạng một đoạn mã JSON.Thường được sử dụng trong các ứng dụng web và dịch vụ API để xác thực và ủy quyền người dùng.Nó gồm 3 phần là Header,Payload,Signature.

->Header:Chứa Các thông tin về loại token và thuật toán mã hóa được sử dụng.

->Payload:Chứa các thông tin mở rộng về người dùng hoặc thông tin khác cần thiết.Có thể tùy chỉnh các trường thông tin trong payload để phán ảnh nhu cầu cụ thể của ứng dụng.

->Signature:Signature được tạo từ Header,Payload và một “secret” key.Nó được sử dụng để xác minh tính toàn vẹn của token và đảm bảo nó không bị sửa trong quá trình truyền tải.

**OAuth(Open Authorization)**

-Là một giao thức xác thực và ủy quyền được sử dụng để cho phép một ứng dụng yêu cầu quyền truy cập vào dữ liệu hoặc tài nguyên của một ứng dụng mà không cần chia sẻ mật khẩu.OAuth thường được sử dụng để cho phép người dùng đăng nhập vào một ứng dụng bằng tài khoản từ một dịch vụ khác như Goole,Facebook,Twitter, và để ứng dụng này truy cập vào dữ liệu của người dùng từ dịch vụ đó. Nó thường đi kèm với JWT(JSON Web Token).

**SSO(Sigle Sign-On)**

-Là một cơ chế xác thực cho phép người dùng đăng nhập 1 lần vào hệ thống và sau đó khi người dùng đó truy cập vào 1 tài nguyên khác của hệ thống thì sẽ không cần phải đăng nhập lại.

-Cơ chế hoạt động của SSO Web thường bao gồm ba phần chính:

+)Identity Provider(IDP):Là dịch vụ hoặc hệ thống chịu trách nhiệm xác thực người dùng và cung cấp thông tin về định danh của họ sau khi đăng nhập.IDP thường là 1 hệ thống quản lý danh tính như Active Directory,LDAP hoặc một dịch vụ đăng nhập bên thứ ba như Google,FaceBook,hoặc AuthO.

+)Service Provider(SP):Là các ứng dụng hoặc dịch vụ mà người dùng muốn truy cập.SP không cần phải xác thực người dùng mà chỉ cần tin tưởng IDP đã xác thực họ.

+)User Agent(User’s Brower):Là trình duyệ web mà người dùng sử dụng để truy cập các ứng dụng hoặc dịch vụ.

-Quy trình hoạt động:

->Người dùng truy cập một ứng dụng hoặc dịch vụ và yêu cầu đăng nhập.

->Ứng dụng chuyển hướng người dùng tới IdP để xác thực.

-> Người dùng đăng nhập vào IdP.

->IdP xác thực người dùng và tạo 1 phiên đăng nhập.

->IdP chuyển hướng người dùng trở lại ứng dụng hoặc dịch vụ với thông tin phiên đăng nhập.

->Service Provicer(SP) sử dụng thông tin phiên đăng nhập để xác thực người dùng và cung cấp quyền truy cập.

**App Configuration**

-HttpSecurity là 1 lớp cung cấp cơ chế cho phép bạn cấu hình cách ứng dụng xử lý việc bảo vệ các yêu cầu Http được gửi về hệ thống.Nó cho phép xác định các quy tắc như xác thực, ủy quyền,các quyền truy cập và cách thức xử lý ngoại lệ bảo mật.

-Các phương thức đi kèm với HttpSecurity:

+)authorizeHttpRequest():Đây là phương thức dùng để cấu hình việc ủy quyền cho các yêu cầu Http.

+)anyRequest():Đây là phương thức báo rằng tất cả các HttpRequest sẽ đều phải được ủy quyền.

+)authenticated():Đây là phương thức bắt buộc phải đăng nhập thì mới có thể được truy cập vào chúng.

+)and():Là dùng để nối các câu lệnh HttpSecurity với nhau.

+)httpBasic():Là 1 Http dùng để xác thực người dùng. Khi sử dụng HttpBasic Authentication , người dùng sẽ được yêu cầu cung cấp tên và mật khẩu thông qua tiêu đề Authorization của yêu cầu Http.Thông tin đăng nhập này sau đó được mã hóa và gửi dưới dạng base64.

**Work Flow JWT**

B1:Người dùng gửi 1 JWT đến server.

B2:Xác thực xem JWT đó có tồn tại hay không bằng cách hỏi 1 lớp đó là UserDetailService lấy thông tin người dùng đôi chiếu với phần Payload của JWT.

B3:UserDetailService sẽ tìm kiếm thông tin người dùng bằng cách gọi 1 lớp DB và với DB đó sẽ trả lại dữ liệu người dùng cho UserDetailService.

B4:So sánh và validate JWT data với người dùng vừa được tìm nếu đúng thì sẽ update vào SecurityContextHolder và gọi FilterChainProxy.

B5:Chuyển yêu cầu đến Controller.

A screenshot of a computer

Description automatically generated

->Dữ liệu token được gửi từ client đến server thông qua tiêu đề của Authorization.Lấy giá trị của tiêu đề Authorization để thực hiện trích xuất thông tin xác thực từ yêu cầu và dùng nó để xác định người dùng và quyền truy cập của họ.