-/\*AuthenticationManager là một interface trong Spring Security, cung cấp phương thức authenticate().  
 Phương thức này nhận một đối tượng Authentication làm đầu vào và trả về một đối tượng Authentication  
 đã được xác thực hoàn toàn.\*/

-/\*UsernamePasswordAuthenticationToken là một lớp con của Authentication.

Trong Spring Security,Authentication là một interface đại diện cho thông tin xác thực của một người dùng.  
UsernamePasswordAuthenticationToken là một lớp cụ thể của Authentication, được thiết kế để đơn giản hóa việc trình bày tên người dùng và mật khẩu.\*/

-/\* Authentication: Đây là một interface trong Spring Security, đại diện cho thông tin xác thực của một người dùng.  
 Khi authenticationManager.authenticate() được gọi, nó trả về một đối tượng Authentication chứa thông tin chi tiết về người dùng đã được xác thực\*/

-/\*Trong quá trình này, AuthenticationManager sẽ sử dụng UserDetailsService để tìm kiếm người dùng dựa trên tên người dùng được cung cấp. Nếu người dùng được tìm thấy, nó sẽ so sánh mật khẩu được cung cấp với mật khẩu đã được mã hóa lưu trữ. Nếu hai mật khẩu này khớp, quá trình xác thực sẽ thành công và AuthenticationManager sẽ trả về một đối tượng Authentication chứa thông tin chi tiết về người dùng đã được xác thực\*/

*/*\*Việc sử dụng GrantedAuthority thay vì String mang lại một số lợi ích:  
-Tính linh hoạt: GrantedAuthority là một interface, do đó bạn có thể tạo các lớp triển khai riêng của mình  
để thêm các thông tin bổ sung nếu cần.  
-Tính an toàn: Khi sử dụng GrantedAuthority, bạn có thể chắc chắn rằng mỗi đối tượng trong collection đều  
biểu diễn một quyền. Trong khi đó, nếu sử dụng String, bạn có thể vô tình thêm vào một chuỗi không phải là quyền.  
-Tính nhất quán: Spring Security sử dụng GrantedAuthority trong nhiều phần của khung như UserDetails và  
Authentication. Việc sử dụng cùng một kiểu giúp mã của bạn trở nên nhất quán hơn.\*/  
// SimpleGrantedAuthority đại diện cho String authority

@Override  
protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) throws ServletException, IOException {  
 try {  
 String jwt = getJwtFromRequest(request);  
 if (StringUtils.*hasText*(jwt) && jwtTokenProvider.validateToken(jwt)) {  
 String username = jwtTokenProvider.getUserNameFormJWT(jwt);  
 UserDetails userDetails = userPrincipleService.loadUserByUsername(username);  
 if (userDetails != null) {  
 */\*Hàm tạo này thường được sử dụng để đặt xác thực người dùng vào Security Context Holder  
 của Spring Security. Bằng cách chỉ thêm danh sách quyền đã được cấp vào hàm tạo, bạn đang  
 nói rằng người dùng đã được xác thực thành công, ngay cả khi danh sách đó trống.\*/* UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken  
 (userDetails, null, userDetails.getAuthorities());  
 *//tao object chứa chi tiết như địa chỉ IP nguồn và ID phiên* authenticationToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));  
 SecurityContextHolder.*getContext*().setAuthentication(authenticationToken);  
 }  
 }  
 } catch (Exception e) {  
 *log*.error("Fail on set user authentication", e);  
 }  
 filterChain.doFilter(request, response);  
}