[Bài đọc] Spring Security - Ủy quyền với Spring Security

Cho đến nay, chúng ta mới chỉ nói về xác thực, ví dụ: kiểm tra tên người dùng và mật khẩu.

Bây giờ chúng ta hãy xem xét quyền, hay đúng hơn là vai trò và quyền hạn trong Spring Security.

Ủy quyền là gì?

Lấy trang web thương mại điện tử điển hình của bạn. Nó có thể bao gồm các phần sau:

* Trang web chính. Giả sử URL của nó là www.youramazinshop.com.
* Có thể là một khu vực dành cho bộ phận hỗ trợ khách hàng, nơi họ có thể đăng nhập và xem những gì một khách hàng đã mua gần đây hoặc bưu kiện của họ ở đâu. URL của nó có thể là www.youramazinshop.com/callcenter.
* Một khu vực quản trị riêng biệt, nơi quản trị viên có thể đăng nhập và quản lý các tác nhân trung tâm gọi hoặc các khía cạnh kỹ thuật khác (như chủ đề, hiệu suất, v.v.) của cửa hàng trực tuyến. URL của nó có thể là www.youramazinshop.com/admin.

Điều này có những ý nghĩa sau, vì chỉ cần xác thực người dùng là không đủ nữa:

Rõ ràng là khách hàng sẽ không thể truy cập vào chức năng hỗ trợ khách hàng  hoặc khu vực quản trị. Anh ta chỉ được phép mua sắm trong trang web.

Một nhân viên hỗ trợ khách hàng sẽ không thể truy cập vào khu vực quản trị.

Trong khi đó, quản trị viên có thể truy cập web-shop, khu vực hỗ trợ khách hàng và khu vực quản trị.

Nói một cách đơn giản, bạn muốn cho phép các quyền truy cập khác nhau cho những người dùng khác nhau, tùy thuộc vào quyền hạn hoặc vai trò của họ.

Authorities là gì? Roles là gì?

Đơn giản:

Authorities (Quyền hạn - ở dạng đơn giản nhất) chỉ là một chuỗi, nó có thể là bất kỳ thứ gì như: user, ADMIN, ROLE\_ADMIN hoặc 53cr37\_r0l3.

Roles(Vai trò) là một thẩm quyền có tiền tố ROLE\_. Vì vậy, một vai trò được gọi là ADMIN cũng giống như một quyền hạn được gọi là ROLE\_ADMIN.

Sự phân biệt giữa vai trò và quyền hạn hoàn toàn là khái niệm và là điều thường khiến những người mới làm quen với Spring Security bối rối.

**Tại sao lại có sự phân biệt giữa vai trò và quyền hạn?**

Thành thật mà nói, tôi đã đọc tài liệu Spring Security cũng như một vài chuỗi StackOverflow liên quan về câu hỏi này và tôi không thể đưa ra câu trả lời chính xác, dứt khoát cho bạn.

**GrantedAuthority là gì? SimpleGrantedAuthority là gì?**

Tất nhiên, Spring Security không cho phép bạn thoát khỏi việc chỉ sử dụng Strings. Có một lớp Java đại diện cho Chuỗi quyền hạn của bạn, một lớp phổ biến là SimpleGrantedAuthority.

public final class SimpleGrantedAuthority implements GrantedAuthority {

private final String role;

@Override

public String getAuthority() {

return role;

}

}

(Lưu ý: Cũng có các lớp quyền hạn khác, cho phép bạn lưu trữ các đối tượng bổ sung (ví dụ: principal) cùng với chuỗi của bạn, tôi sẽ không trình bày chúng ở đây. Hiện tại, chúng ta sẽ chỉ sử dụng SimpleGrantedAuthority.)

**1. UserDetailsService: Nơi lưu trữ và nhận quyền hạn?**

Giả sử bạn đang lưu trữ người dùng trong ứng dụng của riêng mình (UserDetailsService), bạn sẽ có một bảng Users

Bây giờ, bạn chỉ cần thêm một cột có tên " authorities" vào nó. Đối với bài viết này, ta đã chọn một cột chuỗi đơn giản ở đây, mặc dù nó có thể chứa nhiều giá trị được phân tách bằng dấu phẩy. Ngoài ra, ta cũng có thể có một bảng AUTHORITIES hoàn toàn riêng biệt.

|  |  |  |  |
| --- | --- | --- | --- |
| **Bảng Users với Permissions** | | | |
| **username** | **password** | **authorities** |  |
| [john@doe.com](mailto:john@doe.com) | {bcrypt}…​ | ROLE\_ADMIN |  |
| [my@callcenter.com](mailto:my@callcenter.com) | {sha256}…​ | ROLE\_CALLCENTER |  |

Điều duy nhất còn lại cần làm là điều chỉnh UserDetailsService của bạn để đọc cột authorities đó.

public class MyDatabaseUserDetailsService implements UserDetailsService {

UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

User user = userDao.findByUsername(username);

List<SimpleGrantedAuthority> grantedAuthorities = user.getAuthorities().map(authority -> new SimpleGrantedAuthority(authority)).collect(Collectors.toList()); *// (1)*

return new org.springframework.security.core.userdetails.User(user.getUsername(), user.getPassword(), grantedAuthorities); *// (2)*

}

}

(1)   Bạn chỉ cần ánh xạ bất cứ thứ gì bên trong cột cơ sở dữ liệu của mình với danh sách SimpleGrantedAuthority.

(2)   Một lần nữa, chúng ta đang sử dụng triển khai UserDetails cơ bản của Spring Security tại đây. Bạn cũng có thể sử dụng lớp của riêng bạn triển khai UserDetails ở đây.

**2. AuthenticationManager: Nơi lưu trữ và nhận authorities?**

Khi người dùng đến từ một ứng dụng của bên thứ ba, chẳng hạn như Atlassian Cloud, bạn sẽ cần tìm hiểu xem họ đang sử dụng khái niệm gì để hỗ trợ authorities. Atlassian Crowd có các khái niệm về "roles", nhưng không dùng nó để ủng hộ "groups".

Vì vậy, tùy thuộc vào sản phẩm thực tế bạn đang sử dụng, bạn cần ánh xạ điều này tới Spring Security authority, trong AuthenticationProvider của bạn.

public class AtlassianCrowdAuthenticationProvider implements AuthenticationProvider {

Authentication authenticate(Authentication authentication)

throws AuthenticationException {

String username = authentication.getPrincipal().toString();

String password = authentication.getCredentials().toString();

atlassian.crowd.User user = callAtlassianCrowdRestService(username, password); *// (1)*

if (user == null) {

throw new AuthenticationException("could not login");

}

return new UserNamePasswordAuthenticationToken(user.getUsername(), user.getPassword(), mapToAuthorities(user.getGroups())); *// (2)*

}

*// other method ignored*

}

Lưu ý: Đây không phải là mã Atlassian Crowd thực tế, nhưng phục vụ cho mục đích của nó. Bạn xác thực với một dịch vụ REST và lấy lại đối tượng  JSON User, đối tượng này sau đó được chuyển đổi thành đối tượng atlassian.crowd.User.

Người dùng đó có thể là thành viên của một hoặc nhiều nhóm, được cho là chỉ là các chuỗi ở đây. Sau đó, bạn có thể chỉ cần ánh xạ các nhóm này với "SimpleGrantedAuthority" của Spring.

**Xem lại WebSecurityConfigurerAdapter cho Authorities**

Cho đến nay, chúng ta đã nói nhiều về việc lưu trữ và truy xuất các authorities cho người dùng đã xác thực trong Spring Security. Nhưng làm cách nào để bạn bảo vệ các URL với authorities khác nhau với Spring Security’s DSL? Đơn giản:

@Configuration

@EnableWebSecurity

public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.authorizeRequests()

.antMatchers("/admin").hasAuthority("ROLE\_ADMIN") *// (1)*

.antMatchers("/callcenter").hasAnyAuthority("ROLE\_ADMIN", "ROLE\_CALLCENTER") *// (2)*

.anyRequest().authenticated() *// (3)*

.and()

.formLogin()

.and()

.httpBasic();

}

}

(1)   Để truy cập vào /admin (ví dụ: người sử dụng) cần phải được xác thực và có quyền (một chuỗi đơn giản) ROLE\_ADMIN.

(2)   Để truy cập vào /callcenter, bạn cần được xác thực và có quyền ROLE\_ADMIN hoặc ROLE\_CALLCENTER.

(3)   Đối với bất kỳ yêu cầu nào khác, bạn không cần vai trò cụ thể nhưng vẫn cần được xác thực.

Lưu ý rằng mã trên (1,2) tương đương với mã sau:

http

.authorizeRequests()

.antMatchers("/admin").hasRole("ADMIN") *// (1)*

.antMatchers("/callcenter").hasAnyRole("ADMIN", "CALLCENTER") *// (2)*

(1)   Thay vì gọi "hasAuthority", bây giờ bạn gọi "hasRole". Lưu ý: Spring Security sẽ tìm kiếm authority có tên ROLE\_ADMIN trên người dùng được xác thực.

(2)   Thay vì gọi "hasAnyAuthority", bây giờ bạn gọi "hasAnyRole". Lưu ý: Spring Security sẽ tìm kiếm authority có tên ROLE\_ADMIN hoặc ROLE\_CALLCENTER trên người dùng đã xác thực.

**hasAccess và SpEL**

Cuối cùng, nhưng không kém phần quan trọng, cách mạnh mẽ nhất để định cấu hình ủy quyền (authorizations) là với phương thức truy cập (access). Nó cho phép bạn chỉ định khá nhiều biểu thức SpEL hợp lệ.

http

.authorizeRequests()

.antMatchers("/admin").access("hasRole('admin') and hasIpAddress('192.168.1.0/24') and @myCustomBean.checkAccess(authentication,request)") // (1)

(1)   Bạn đang kiểm tra xem người dùng có ROLE\_ADMIN, với địa chỉ IP cụ thể cũng như kiểm tra bean tùy chỉnh.

*(Nguồn: https://www.marcobehler.com/guides/spring-security#\_authorization\_with\_spring\_security)*

Last modified: Sunday, 15 August 2021, 10:50 PM