1. **Spring Security là gì?**

Spring Security hoạt động theo mô hình client-server. Khi một client gửi một request đến server, server sẽ xác thực người dùng và phân quyền để đảm bảo rằng người dùng chỉ có thể truy cập vào những tài nguyên mà họ được phép truy cập.

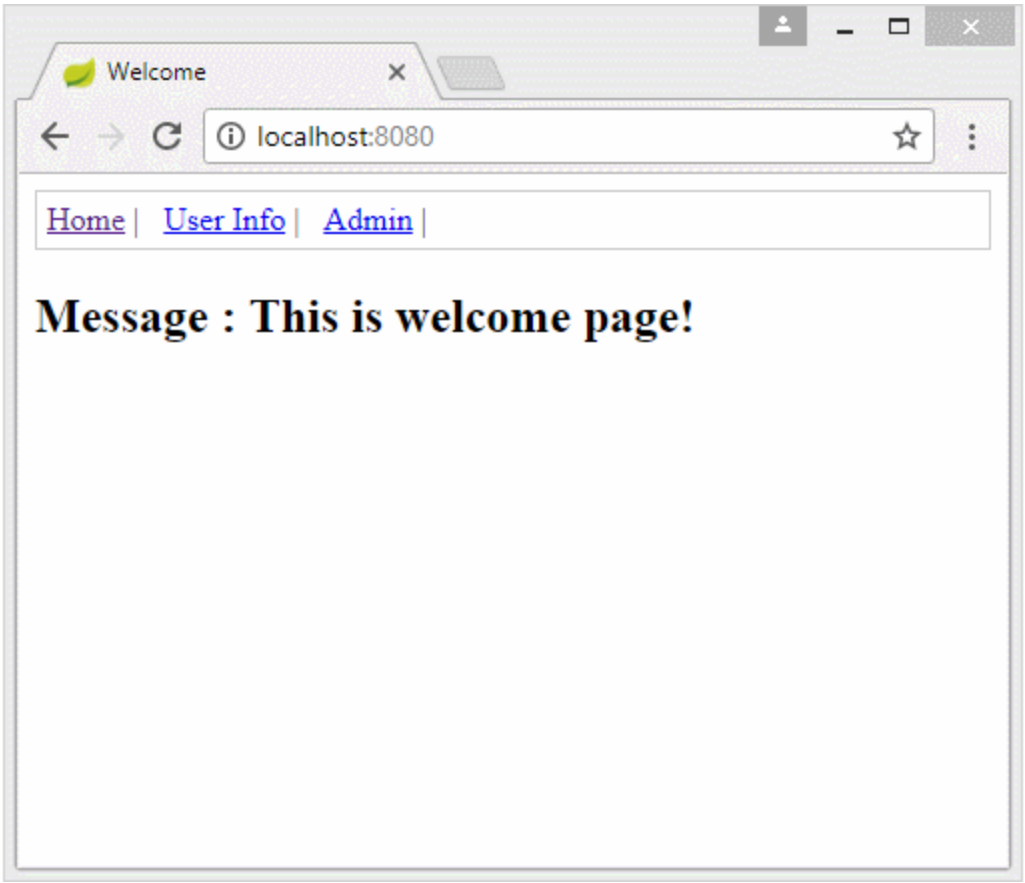
Ứng dụng phân quyền tùy thuộc vào user đăng nhập vào hệ thống là user hay admin mà ta cho phép họ vào trang web tương ứng.

Ví dụ.

Trang Home thì ai vào cũng được.

Trang Admin thì chỉ có admin được vào và thấy được trang. Nếu là role user và vào trang Admin thì mình hiện thông báo lỗi bạn không có quyền.

Trang User Info thì user và admin được phép vào.



2. **Các khái niệm về Spring Security:**

Luôn phải setup 2 bước:

Bước 1: Authentication (Who?) : Khi nói về authentication là chức năng đăng nhập vào hệ thống. Authentication nghĩa ai là người dùng của hệ thống.

Bước 2: Authorization (What): Khi nói về authorization là về quyền hạn của Authentication được phép làm những công việc gì?

Trong ví dụ trên có Authentication là user và admin. Bước đầu tiên họ phải authentication. Xác thực mình là user trong hệ thống.

Tiếp đến tuỳ vào role của mình là admin hay user mà mình chỉ có quyền truy cập Authorization một số trang nhất định thuộc thẩm quyền của mình.

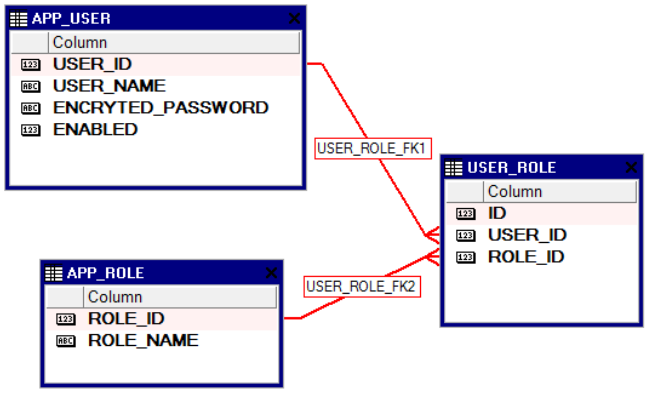
3. Hướng dẫn xây dựng ứng dụng Spring Security: Step by Step

Luồng đi của ứng dụng như sau.

- ✓ User nhập vào username và password sau đó bấm login.
- ✓ Server sẽ nhận được request từ người dùng và chuyển tới controller tương ứng do ta cấu hình trong file configure của spring security.
- ✓ Controller sẽ gọi Service và Service sẽ gọi database để lấy thông tin authentication đúng không và role người dùng là gì?
- ✓ Sau khi có thông tin đúng thì trả kết quả lại cho người dùng.

Bước 1: Chuẩn bị database để lưu thông tin user và quyền

Dtabase để lưu thông tin người dùng và role (vai trò, được phép làm gì). Phục vụ cho việc truy vấn username và role có hợp lệ hay không?



Ứng dụng spring security sẽ lưu user name và quyền vào trong database.
Authentication: Ai được vào hệ thống? (user/admin)

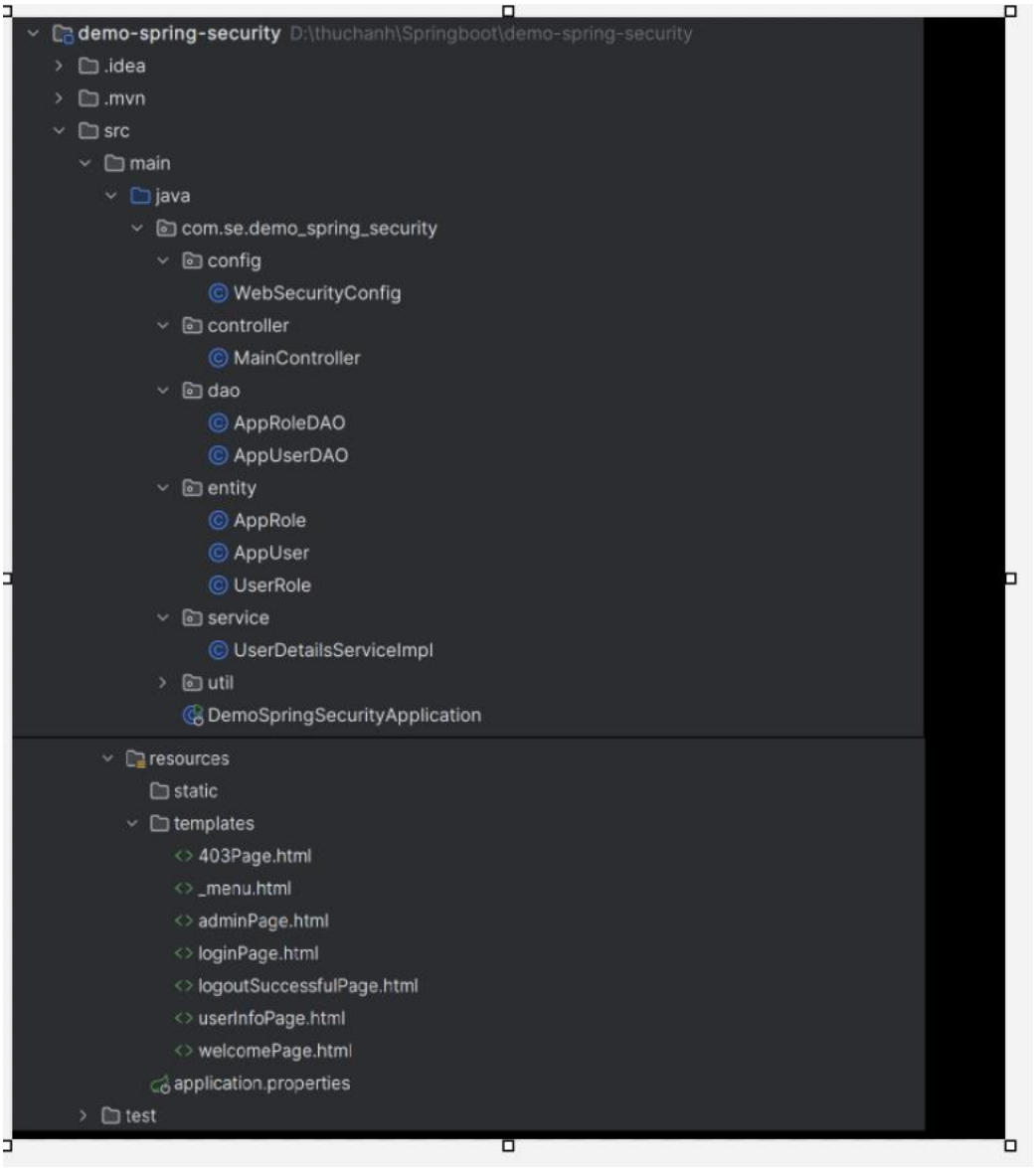
1. Table APP_USER dùng để lưu thông tin username và password. Khi người dùng đăng nhập họ truyền user name và password vào form sau đó code của mình sẽ query trong database xem là username và password có đúng như trong database không?

Authorization: Người dùng vào hệ thống thì có quyền gì?

2. Table APP_ROLE dùng để xác định xem user sau khi login thành công thì được phép vào những trang nào.

   Ví dụ admin vào được 2 trang user và admin page. Nhưng user chỉ được phép vào 1 trang là user page.
3. Table USER_ROLE là table dùng để nối 2 bảng APP_USER và APP_ROLE, nó được dùng để cho phép 1 user có thể có nhiều quyền.

   Ví dụ như admin có thể vào cả 2 trang user và admin.



Bước 2: Thêm dependencies cần thiết trong pom.xml

Thêm các dependencies spring security, thymeleaf, mysql (mariadb, h2,…), lombok, devtool ,jpa.

```
<dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
```

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
```

Bước 3: Tạo form login . theo framework thymeleaf

Tạo form login .

Khi người dùng click vào nút submit thì action mình dùng là /j_spring_security_check cái này là mặc định của spring.

```
<h3>Enter user name and password:</h3>
<form name='f' th:action="@{/j_spring_security_check}" method='POST'>
    <table>
        <tr>
            <td>User:</td>
            <td><input type='text' name='username' value=''></td>
        </tr>
        <tr>
            <td>Password:</td>
            <td><input type='password' name='password' /></td>
        </tr>
        <tr>
            <td>Remember Me?</td>
            <td><input type="checkbox" name="remember-me" /></td>
        </tr>
        <tr>
            <td><input name="submit" type="submit" value="submit" /></td>
        </tr>
    </table>
</form>
```

Bước 4: Tạo file WebSecurityConfig để cấu hình cho Spring security.

Trong thư mục configure/WebSecurityConfig.

```
@Override
    protected void configure(HttpSecurity http) throws Exception {

        http.csrf().disable(); //CSRF ( Cross Site Request Forgery) là kĩ thuật tấn công bằng cách sử dụng quyền chứng thực của người sử dụng đối với 1 website khác

        // Các trang không yêu cầu login như vậy ai cũng có thể vào được admin hay user hoặc guest có thể vào các trang
        http.authorizeRequests().antMatchers("/", "/login", "/logout").permitAll();

        // Trang /userInfo yêu cầu phải login với vai trò ROLE_USER hoặc ROLE_ADMIN.
        // Nếu chưa login, nó sẽ redirect tới trang /login.sau Mình dung hasAnyRole để cho phép ai được quyền vào
        // 2  ROLE_USER và ROLEADMIN thì ta lấy từ database ra cái mà mình chèn vô ở bước 1 (chuẩn bị database)
        http.authorizeRequests().antMatchers("/userInfo").access("hasAnyRole('ROLE_USER', 'ROLE_ADMIN')");

        // Trang chỉ dành cho ADMIN
        http.authorizeRequests().antMatchers("/admin").access("hasRole('ROLE_ADMIN')");

        // Khi người dùng đã login, với vai trò user .
        // Nhưng cố ý  truy cập vào trang admin
        // Ngoại lệ AccessDeniedException sẽ ném ra.
        // Ở đây mình tạo thêm một trang web lỗi tên 403.html (mọi người có thể tạo bất cứ tên nào kh
        http.authorizeRequests().and().exceptionHandling().accessDeniedPage("/403");

        // Cấu hình cho Login Form.
        http.authorizeRequests().and().formLogin()//
                // Submit URL của trang login
```

```
                .loginProcessingUrl("/j_spring_security_check") // Bạn còn nhớ bước 3 khi tạo form login thì action của
nó là j_spring_security_check giống ở
                .loginPage("/login")//
                .defaultSuccessUrl("/userAccountInfo")//đây Khi đăng nhập thành công thì vào trang này. userAccountInfo
sẽ được khai báo trong controller để hiển thị trang view tương ứng
                .failureUrl("/login?error=true")// Khi đăng nhập sai username và password thì nhập lại
                .usernameParameter("username")// tham số này nhận từ form login ở bước 3 có input  name='username'
                .passwordParameter("password")// tham số này nhận từ form login ở bước 3 có input  name='password'
                // Cấu hình cho Logout Page. Khi logout mình trả về trang


                .and().logout().logoutUrl("/logout").logoutSuccessUrl("/logoutSuccessful");


        // Cấu hình Remember Me. Ở form login bước 3, ta có 1 nút remember me. Nếu người dùng tick vào đó ta sẽ dùng
cookie lưu lại trong 24h

        http.authorizeRequests().and() //
                .rememberMe().tokenRepository(this.persistentTokenRepository()) //
                .tokenValiditySeconds(1 * 24 * 60 * 60); // 24h

    }
    @Bean
    public PersistentTokenRepository persistentTokenRepository() {
        InMemoryTokenRepositoryImpl memory = new InMemoryTokenRepositoryImpl(); // Ta lưu tạm remember me trong memory
(RAM). Nếu cần mình có thể lưu trong database
        return memory;
    }
```

logout

2. Method thứ 2 là public BCryptPasswordEncoder passwordEncoder() Method này dùng để mã hoá password của người dùng Ví dụ người dùng nhập password là abc@123 thì nó sẽ mã hoá là $2a$10$PrI5Gk9L.tSZiW9FXhTS8O8Mz9E97k2FZbFvGFFaSsiTUIl.TCrFu. Mọi người có thể đọc cách encode và thư viện encode ở file EncrytedPasswordUtils trong github .

```
@Bean
   public BCryptPasswordEncoder passwordEncoder() {
       BCryptPasswordEncoder bCryptPasswordEncoder = new BCryptPasswordEncoder();
       return bCryptPasswordEncoder;
   }
```

3. Method thứ 3 là configureGlobal(AuthenticationManagerBuilder auth) throws Exception Trong Spring Security có một object quan trọng đó là UserDetailsService. Đây là object của Spring, nó nắm giữ thông tin quan trọng như Username này là ai trong hệ thống , UserName này có quyền gì. Chúng ta sẽ đi chi tiết trong bước 5 tiếp theo để hiểu nó làm được .

```
 @Autowired
    private UserDetailsServiceImpl userDetailsService;


    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {

        //gọi userDetailsService trong bước 5 tiếp theo
        auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());

    }
```

Bước 5: Tạo UserDetailsServiceImpl

File này sẽ implement UserDetailsService của Spring và định nghĩa cách kiểm tra username , password và quyền của user có hợp lệ hay không Khi user login vào hệ thống ta sẽ query xuống database để kiểm tra user có đúng trong database không và quyền là gì ?

```
    @Override
    public UserDetails loadUserByUsername(String userName) throws UsernameNotFoundException {
        // đầu tiên mình query xuống database xem có user  đó không
        AppUser appUser = this.appUserDAO.findUserAccount(userName);

        //Nếu không tìm thấy User thì mình thông báo lỗi
        if (appUser == null) {
            System.out.println("User not found! " + userName);
            throw new UsernameNotFoundException("User " + userName + " was not found in the database");
        }


        // Khi đã có user rồi thì mình query xem user đó có những quyền gì (Admin hay User)
        // [ROLE_USER, ROLE_ADMIN,..]
        List<String> roleNames = this.appRoleDAO.getRoleNames(appUser.getUserId());

        // Dựa vào list quyền trả về mình tạo đối tượng GrantedAuthority  của spring cho quyền đó
        List<GrantedAuthority> grantList = new ArrayList<GrantedAuthority>();
        if (roleNames != null) {
            for (String role : roleNames) {
                // ROLE_USER, ROLE_ADMIN,..
                GrantedAuthority authority = new SimpleGrantedAuthority(role);
                grantList.add(authority);
            }
        }

        //Cuối cùng mình tạo đối tượng UserDetails của Spring và mình cung cấp cá thông số như tên , password và quyền
        // Đối tượng userDetails sẽ chứa đựng các thông tin cần thiết về user từ đó giúp Spring Security quản lý được phân quyền như ta đã
        // cấu hình trong bước 4 method configure
        UserDetails userDetails = (UserDetails) new User(appUser.getUserName(),
                appUser.getEncrytedPassword(), grantList);

        return userDetails;
    }
```

Bước 6: Tạo các điều hướng trong controller

Khi người dùng đã đang nhập thành công, họ có thể điều hướng tới các trang khác, các mapping trong Controller điều hướng người dùng tới các view tương ứng. Những điều hướng này nằm ở bước 4 method configure .

```
@RequestMapping(value = { "/", "/welcome" }, method = RequestMethod.GET)
    public String welcomePage(Model model) {
        model.addAttribute("title", "Welcome");
        model.addAttribute("message", "This is welcome page!");
        return "welcomePage";
    }

    //Đây là trang Admin
```

```
9      @RequestMapping(value = "/admin", method = RequestMethod.GET)
10     public String adminPage(Model model, Principal principal) {
11
12         User loginedUser = (User) ((Authentication) principal).getPrincipal();
13
14         String userInfo = WebUtils.toString(loginedUser);
15         model.addAttribute("userInfo", userInfo);
16
17         return "adminPage";
18     }
19
20     @RequestMapping(value = "/login", method = RequestMethod.GET)
21     public String loginPage(Model model) {
22
23         return "loginPage";
24     }
25
26     // khi người dùng logout khỏi hệ thống
27     @RequestMapping(value = "/logoutSuccessful", method = RequestMethod.GET)
28     public String logoutSuccessfulPage(Model model) {
29         model.addAttribute("title", "Logout");
30         return "logoutSuccessfulPage";
31     }
32
33     // khi người dùng đăng nhập thành công
34     @RequestMapping(value = "/userInfo", method = RequestMethod.GET)
35     public String userInfo(Model model, Principal principal) {
36
37         // Sau khi user login thanh cong se co principal
38         String userName = principal.getName();
39
40         System.out.println("User Name: " + userName);
41
42         User loginedUser = (User) ((Authentication) principal).getPrincipal();
43
44         String userInfo = WebUtils.toString(loginedUser);
45         model.addAttribute("userInfo", userInfo);
46
47         return "userInfoPage";
48     }
49
50     // khi người dùng là user mà thâm nhập trang admin thì mình vào đây
51     @RequestMapping(value = "/403", method = RequestMethod.GET)
52     public String accessDenied(Model model, Principal principal) {
53
54         if (principal != null) {
55             User loginedUser = (User) ((Authentication) principal).getPrincipal();
56
57             String userInfo = WebUtils.toString(loginedUser);
58
59             model.addAttribute("userInfo", userInfo);
60
61             String message = "Hi " + principal.getName() //
62                     + "<br> You do not have permission to access this page!";
63             model.addAttribute("message", message);
64
65         }
66
67         return "403Page";
68     }
```

Bước 7: Tạo Repository để query database

Chúng ta tạo file AppUserDAO sử dụng entity manager để tạo và thực thi câu lệnh SQL (có thể sử dụng JPA để query). Lớp UserDetailsServiceImpl sẽ nhúng AppUserDAO vào trong nó để thực hiện nhiệm vụ kiểm tra xem user có trong database không ?

```
@Repository
@Transactional
public class AppUserDAO {

    @Autowired
    private EntityManager entityManager;

    public AppUser findUserAccount(String userName) {
        try {
            String sql = "Select e from " + AppUser.class.getName() + " e " //
                    + " Where e.userName = :userName ";

            Query query = entityManager.createQuery(sql, AppUser.class);
            query.setParameter("userName", userName);

            return (AppUser) query.getSingleResult();
        } catch (NoResultException e) {
            return null;
        }
    }

}
```

Bước 8: Tạo các trang view cần thiết để hiện thị

Tham khảo các view (Thymeleaf) trong SourceCode

Bước 9: Chạy ứng dụng

Cấu hình cho Spring security.

Chú ý bước 4, nơi cấu hình và phân quyền trong Spring Security