SPRING SECURITY

CYBERSOFT.EDU.VN

Spring Security

- Sercurity là gì?
- Sercurity Context.
- Đối tượng UserDetails.
- Lớp UserDetailsService.
- Các bước cấu hình Security.
- Đăng nhập trang quản trị, phân quyền.
- Đăng nhập trang người dùng, phân quyền.
- Viết chức năng đăng ký thành viên.

Spring Security là gì?

- **Spring security** là một framework (công cụ) **cung cấp và xử lý các vấn đề về xác thực** (authentication) **và phân quyền** (authorization) cho các ứng dụng web.
- **Spring security** sẽ **tự động tạo form đăng nhập**, **sau khi đăng nhập một đối tượng user sẽ được lưu trong session**, đối tượng user này sẽ gồm các thông tin như username, password, các quyền…
- **Spring Security** chống được các kỹ thuật hacking tinh vi:
  - ✓ **Session fixation:** Tấn công chiếm quyền điều khiển session của người dùng.
  - ✓ **Clickjacking:** Click chuột tự động (ví dụ click vào nút Like Facebook mà không xin phép người dùng).
  - ✓ **CSRF (Cross-site request forgery):** Tạo truy vấn (request) giả mạo truyền từ trang này sang trang khác.

Spring Security

❑ **Spring Security** cung cấp 2 cơ chế cơ bản:

✓ **Authentication**(Xác thực): Là tiến trình **xác thực** (kiểm tra) **danh tính của một người dùng hoặc một hệ thống** khác đang truy cập vào hệ thống bảo mật hiện tại.

  ✓ **Authentication** tương tác với người dùng thông qua form và xác thực dựa trên tên người dùng mà mật khẩu (password-based authentication).

✓ **Authorization** (Phân quyền): Là tiến trình **quyết định xem người dùng hoặc hệ thống sau khi xác thực có được quyền thực hiện một hành động nào đó trong ứng dụng** của bạn hay không.

❖ **Các hình thức phân quyền thường gặp:**

  ✓ **Role-based authorization**: Phân quyền dựa trên vai trò của người dùng.

  ✓ **Object-based authorization**: Phân quyền theo đối tượng.

SecurityContext

- **SecurityContext:** là interface cốt lõi của Spring Security, **lưu trữ tất cả các chi tiết liên quan đến bảo mật trong ứng dụng**.
- **SecurityContextHolder:** Lớp này **lưu trữ security context hiện tại của ứng dụng**, bao gồm chi tiết của **principal** đang tương tác với ứng dụng.
  - ✓ **Principal** có thể hiểu là **một người, một thiết bị hoặc một hệ thống nào đó** có thể thực hiện một hành động trong ứng dụng của bạn.

❖ Đoạn code dưới đây giúp lấy username của principal đã được xác thực:

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

if (principal instanceof UserDetails) {
    String username = ((UserDetails) principal).getUsername();
} else {
    String username = principal.toString();
}
```

UserDetails

- **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.

✓ **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.

UserDetailsService

❑ UserDetailsService

❖ Là một interface có duy nhất một phương thức:

```
UserDetails loadUserByUsername(String username) throws 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 UserDetails.

❑ GrantedAuthority

✓ Là một **quyền được cấp cho principal**. Các quyền đều có tiền tố là **ROLE_**

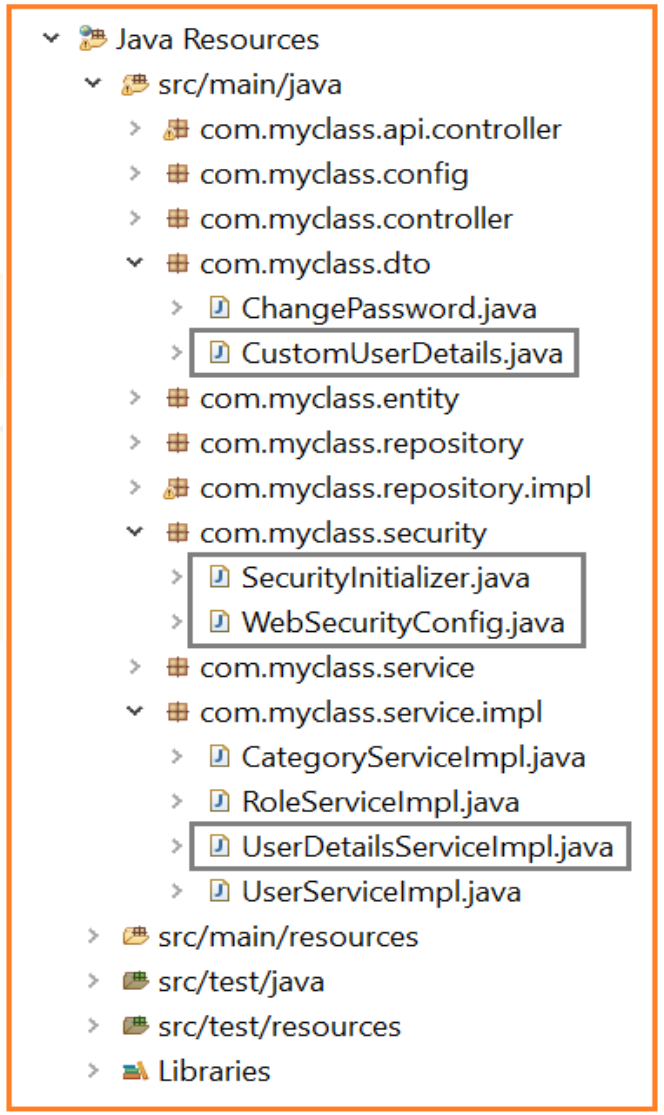
- ✓ Ví dụ: ROLE_ADMIN, ROLE_MEMBER,...

Các bước cấu hình AdminSecurity

- ✓ **Bước 1**: Tải thư viện **Spring Sercurity**.
- ✓ **Bước 2**: Định nghĩa hàm **findByEmail** trong tầng Repository.
- ✓ **Bước 3**: Tạo đối tượng **CustomUserDetails** implement từ **UserDetails** để thêm một số thuộc tính cho lớp UserDetails.
- ✓ **Bước 4**: Định nghĩa lớp **UserDetailsServiceImpl** imlement từ interface **UserDeatilsService** để load thông tin và quyền của người dùng.
- ✓ **Bước 5**: Tạo lớp **AdminSecurityConfig** kế thừa từ lớp **WebSecurityConfigurerAdapter** để cấu hình Security.
- ✓ **Bước 6**: Tạo lớp **SecurityInitializer** kế thừ từ lớp **AbstractSecurityWebApplicationInitializer**.
- ✓ **Bước 7**: Khai báo lớp **AdminSecurityConfig** vào DispatcherServlet.

Cấu trúc thư mục

Cấu trúc thư mục



Thư viện sử dụng

```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
    <version>5.1.5.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>5.1.5.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.thymeleaf.extras</groupId>
    <artifactId>thymeleaf-extras-springsecurity4</artifactId>
    <version>3.0.2.RELEASE</version>
</dependency>
```

UserRepository

Trả về một đối tượng User nếu như tìm thấy khớp email.

UserRepository

```
public User findByEmail(String email) {
    String hql = "FROM users WHERE email = :email";
    try {
        Session session = sessionFactory.getCurrentSession();
        Query<User> query = session.createQuery(hql, User.class);
        query.setParameter("email", email);
        return query.getSingleResult();
    }
    catch (HibernateException e) {
        e.printStackTrace();
    }
    return null;
}
```

CustomUserDetails

Mặc định Spring chỉ cung cấp cho UserDetails thuộc tính username, password và một danh sách chứa các quyền của người dùng để lưu thông tin vào Session.

```java
public class CustomUserDetails extends User implements UserDetails{
    private static final long serialVersionUID = 1L;

    // Mặc định lớp User chỉ có username, password, role
    public CustomUserDetails(String username, String password,
                Collection<? extends GrantedAuthority> authorities) {
        super(username, password, authorities);
    }

    // Nếu muốn thêm thông tin để lưu vào session bạn có thể thêm ở đây
    // Ví dụ bạn có thể thêm fullname, avatar,...
}
```

Lớp CustomUserDetail mở rộng từ lớp User của Spring Sercurity cho phép thêm các thuộc tính vào nếu muốn.

UserDetailsService

```java
@Service
public class UserDetailsServiceImpl implements UserDetailsService {

    @Autowired
    private UserRepository userRepository;

    public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {
        // Lấy ra user có email giống với email người dùng gửi lên từ form đăng nhập
        User user = userRepository.findByEmail(email);
        if(user == null) throw new  UsernameNotFoundException("Không tìm thấy tài khoản!");

        // Tạo danh sách chứa tên quyền cả người dùng
        List<GrantedAuthority> authorities = new ArrayList<GrantedAuthority>();
        String roleName = user.getRole().getName(); // Lấy ra tên quyền
        authorities.add(new SimpleGrantedAuthority(roleName)); // Lưu vào danh sách

        // Trả về đối tượng chứa thông tin email, password và quyền
        return new CustomUserDetails(user.getEmail(), user.getPassword(), authorities);
    }

}
```

Lớp Service dùng để lấy ra thông tin tài khoản và quyền từ database sau đó gán vào cho đối tượng CustomUserDeails.

WebSecurityConfig

```java
@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private UserDetailsService userDetailsService;

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
    }

    protected void configure(HttpSecurity http) throws Exception {

    @Override
    public void configure(WebSecurity web) throws Exception {
        web.ignoring().antMatchers("/static/*");
    }
}
```

Bean PasswordEncoder dùng để giải mã mật khẩu (Sử dụng thư viện JBcrypt)

Khai báo Service lấy thông tin user từ database và phương thức giải mã mật khẩu.

Phương thức cấu hình đăng nhập, phân quyền.

Cấu hình bỏ qua, không kiểm tra đăng nhập cho các file tĩnh.

Đăng nhập - Phân quyền

PHÂN QUYỀN
Quyền Admin sẽ được truy cập vào link bắt đầu bằng /admin

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.csrf().disable()
    .antMatcher("/admin/**")
    .authorizeRequests()
    .antMatchers("/admin/**")
    .hasAnyRole("ADMIN","MANAGER")
    .anyRequest().permitAll();

    http.formLogin()
    .loginPage("/admin/login")
    .loginProcessingUrl("/admin/login")
    .usernameParameter("email")
    .passwordParameter("password")
    .defaultSuccessUrl("/admin/home")
    .defaultSuccessUrl("/admin/login?error=true");

    http.logout()
    .logoutUrl("/admin/logout")
    .logoutSuccessUrl("/admin/login")
    .deleteCookies("JSESSIONID");

    http.exceptionHandling()
    .accessDeniedPage("/error/403");
}
```

=> Chỉ xử lý cho request bắt đầu bằng /admin.

=> Những request khác có thể truy cập link bất kỳ ngoại trừ link bắt đầu bằng /admin đã bị phân quyền.

=> Link tới trang đăng nhập (GET).

=> Link đăng nhập (POST).

=> Thuộc tính của form Spring sẽ lấy giá trị để đăng nhập.

=> Đăng nhập thành công chuyển hướng đến link này.

=> Đăng nhập thất bại chuyển hướng đến link này.

=> Link logout.

=> Sau khi logout sẽ chuyển đến link này.

=> Xóa cookie.

=> Truy cập vào link không có quyền sẽ chuyển đến link này.

Cấu hình

SecurityInitializer

```
public class SecurityInitializer extends AbstractSecurityWebApplicationInitializer{
    // Lớp này chỉ cần kế thừa từ AbstractSecurityWebApplicationInitializer là được, không cần code.
}
```

DispatcherServlet

```
public class WebInitializer extends AbstractAnnotationConfigDispatcherServletInitializer{

    @Override
    protected Class<?>[] getRootConfigClasses() {
        // TODO Auto-generated method stub
        return new Class[] {
                HibernateConfig.class,
                SwaggerConfig.class,
                WebSecurityConfig.class
            };
    }

    protected Class<?>[] getServletConfigClasses() {

    protected String[] getServletMappings() {

    protected Filter[] getServletFilters() {
}
```

Khai báo lớp cấu hình Security.

Controller - View

AuthenticationController

```
@Controller
public class AdminAuthenticationController {

    @GetMapping("admin/login")
    public String login(@RequestParam(required = false) String error,
            Model model) {
        if(error != null && !error.isEmpty()) {
            model.addAttribute("message", "Sai tài khoản hoặc mật khẩu!");
        }
        return "adminLogin";
    }
}
```

auth/login.jsp

```
<c:url value="/admin/login" var="action"/>
<form action="${ action }" method="post" class="md-float-material form-material">
    <div class="auth-box card">
        <div class="card-block">
            <div class="row m-b-20">
                <div class="col-md-12">
                    <h3 class="text-center txt-primary">Đăng nhập</h3>
                    <h6 class="text-danger text-center m-0">${ message }</h6>
                </div>
            </div>
            <div class="form-group form-primary">
                <label for="">Email</label>
                <input type="text" name="email" class="form-control">
            </div>
            <div class="form-group form-primary">
                <label for="">Mật khẩu</label>
                <input type="password" name="password" class="form-control">
            </div>
            <div class="row m-t-30">
                <div class="col-md-12">
                    <button class="btn btn-primary btn-md btn-block
                        waves-effect text-center m-b-20">LOGIN</button>
                </div>
            </div>
        </div>
    </div>
</form>
```

Tổng kết

- ✓ Sercurity là gì?
- ✓ Sercurity Context.
- ✓ Đối tượng UserDetails.
- ✓ Lớp UserDetailsService.
- ✓ Các bước cấu hình Security.
- ✓ Đăng nhập trang quản trị, phân quyền.
- ✓ Đăng nhập trang người dùng, phân quyền.
- ✓ Viết chức năng đăng ký thành viên.