FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

SPRING BOOT SECURITY 1

GIẢNG VIÊN: NGUYỄN NGHIỆM

AGENDA

- Giới thiệu Web Security
- Spring Authentication
- Xây dựng Spring Bean AuthService xử lý thông tin đăng nhập
- Tùy biến Login Form
- Các thành phần Spring Security
  - UserDetails, UserDetailsService, InMemoryUserDetailsManager
  - PasswordEncoder, PasswordEncoderFactories, DelegatingPasswordEncoder
  - SecurityFilterChain

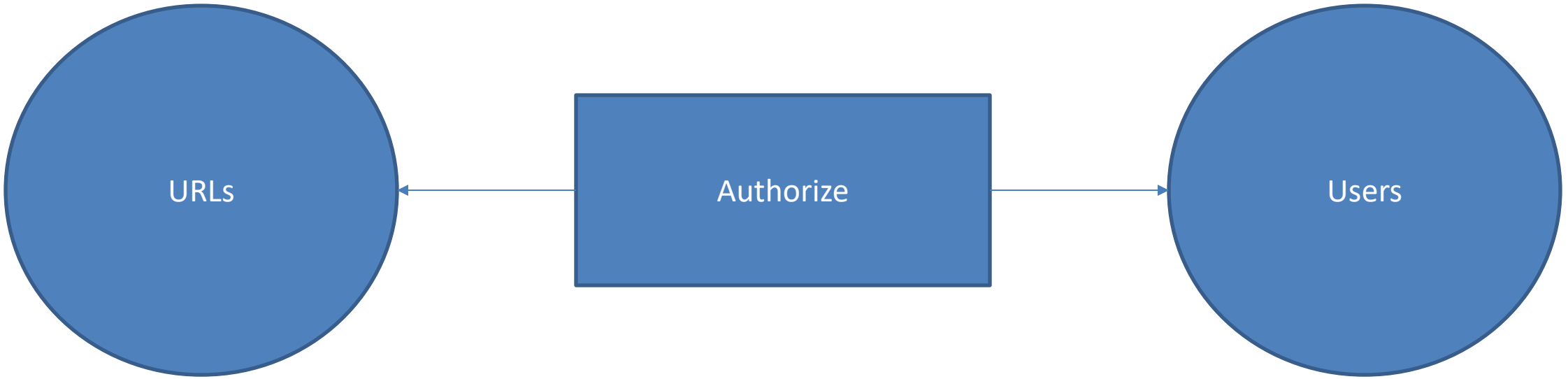
Web Security

- Bảo mật web (Web Security) là các giải pháp làm cho ứng dụng web trở nên an toàn
- Một số biểu hiện thường gặp
  - Yêu cầu đăng nhập mới được phép truy xuất, thậm chí phải đăng nhập với vai trò được cấp phù hợp.
  - Hình thức đăng nhập có thể thực hiện trực tiếp từ website hoặc từ mạng xã hội (Google, Facebook…)
  - Giao diện của một trang web có thể thay đổi khác nhau trước và sau khi đăng nhập
  - Để truy xuất trang web bạn cần phải xác minh với một số hình thức như Capcha, hình ảnh, tích chọn "Tôi không phải robot"…

- **XSS** – Cross-Site Scripting
  - Ngăn chặn thực hiện của script chứa trong data cần hiển thị
- **CORS** – Cross-Site Resource Sharing
  - Ngăn chặn truy xuất tài nguyên trên một website khác
- **CSRF** – Cross-Site Request Forgery
  - Ngăn chặn các request giả lập từ chương trình
- **Validation**
  - Kiểm soát tính hợp lệ của dữ liệu đầu vào
- **Authentication (Xác thực)**
  - Ngăn chặn thực hiện khi chưa đăng nhập
- **Authorization (Ủy thác)**
  - Ngăn chặn thực hiện khi đăng nhập không đúng vai trò

- Spring Security là một framework, cung cấp các kỹ thuật trong việc thực hiện phòng vệ ứng dụng web
- Đa dạng hình thức phân quyền sử dụng
  - Cấu hình
  - @Annotation
  - Lập trình
  - Giao diện
- Chuẩn hóa mô hình dữ liệu người sử dụng và phương pháp mã hóa mật khẩu
- Tùy biến hình thức đăng nhập (Login UI), đăng xuất, xử lý lỗi truy cập không đúng vai trò
- Đăng nhập với user từ mạng xã hội (facebook, gmail...)

AUTHENTICATION & AUTHORIZATION



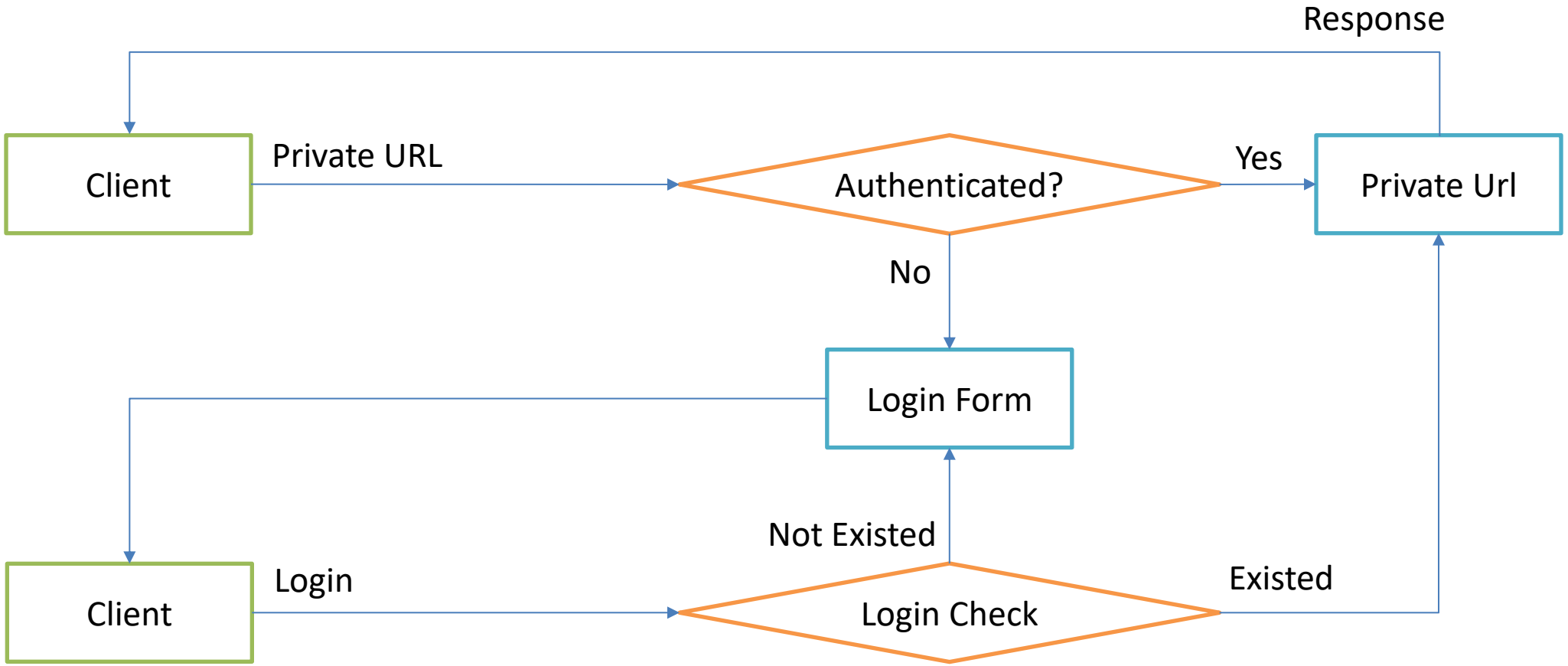
Tài nguyên cần bảo vệ

Cấu hình phân quyền sử dụng
Cấu hình form đăng nhập

Tài khoản người sử dụng

Authentication

Authentication Flow



Authentication Case Study

Controller

- ✓ @/
- ✓ @/poly/url0
- ✓ @/poly/url1
- ✓ @/poly/url2
- ✓ @/poly/url3
- ✓ @/poly/url4

Authorizing

1. Phải đăng nhập mới được phép truy xuất các tài nguyên:
   - ✓ @/poly/url0
   - ✓ @/poly/url1
   - ✓ @/poly/url2
   - ✓ @/poly/url3
   - ✓ @/poly/url4
2. Sử dụng form đăng nhập mặc định và tùy biến

User Source

Users

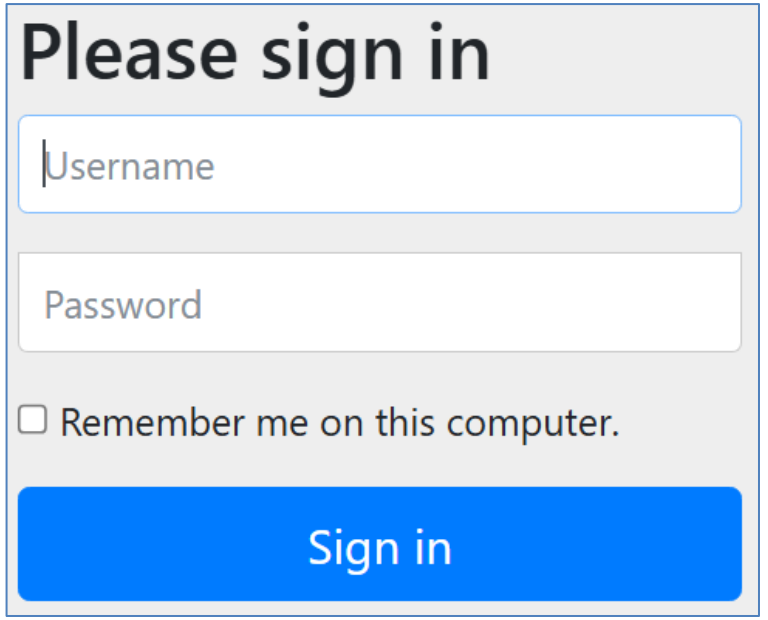
Username	Password	Enabled
user@gmail.com	123	True
admin@gmail.com	123	True
both@gmail.com	123	True

AUTHENTICATION CASE STUDY

1

Truy xuất ***Home*** chưa đăng nhập

2



Truy xuất ***URL<i>*** chưa đăng nhập

3

Truy xuất ***URL<i>*** đã đăng nhập

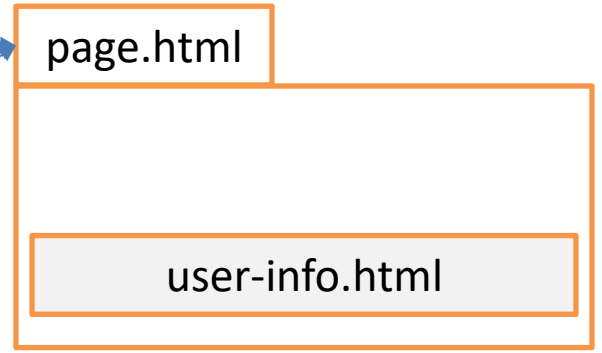
THƯ VIỆN CẦN THIẾT (DEPENDENCY)

- Khai báo bổ sung thư viện Security vào pom.xml

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
        <groupId>org.thymeleaf.extras</groupId>
        <artifactId>thymeleaf-extras-springsecurity6</artifactId>
</dependency>
```

CONTROLLER

```
@Controller
public class MyController {
	@RequestMapping("/")
	public String home(Model model) {
		model.addAttribute("message", "@/ => home()");
		return "page";
	}
	@RequestMapping("/poly/url0")
	public String method0(Model model) {
		model.addAttribute("message", "@/poly/url0 => method1()");
		return "page";
	}
	@RequestMapping("/poly/url1")
	public String method1(Model model) {
		model.addAttribute("message", "@/poly/url1 => method1()");
		return "page";
	}
	...@/poly/url1 -> @/poly/url4
}
```



VIEW (THYMELEAF)

page.html

```
<a href="/">Home</a> |
<a href="/poly/url0">URL0</a> |
<a href="/poly/url1">URL1</a> |
<a href="/poly/url2">URL2</a> |
<a href="/poly/url3">URL3</a> |
<a href="/poly/url4">URL4</a>
<hr>
<h1>Page: <b th:text="${message}"></b></h1>
<hr>
<div th:replace="~{/user-info.html}"></div>
```

user-info.html

```
<div th:switch="${@auth.authenticated}">
    <ul th:case="true">
        <li>Username: <b>[[${@auth.username}]]</b></li>
        <li>Roles: <b th:text="${@auth.roles}"></b></li>
        <li><a href="/logout">Sign-out</a></li>
        <li th:if="${@auth.hasAnyRole('ADMIN')}">
            <a href="#">Goto Administration</a>
        </li>
    </ul>
    <ul th:case="false">
        <li>Bạn chưa đăng nhập</li>
    </ul>
</div>
```

SPRING BEAN AUTHSERVICE

```java
@Service("auth")
public class AuthService {
    public Authentication getAuthentication() {
        return SecurityContextHolder.getContext().getAuthentication();
    }
    public String getUsername() {
        return this.getAuthentication().getName();
    }
    public List<String> getRoles() {
        return this.getAuthentication().getAuthorities().stream()
                .map(au -> au.getAuthority().substring(5)).toList();
    }
    public boolean isAuthenticated() {
        String username = this.getUsername();
        return (username != null && !username.equals("anonymousUser"));
    }
    public boolean hasAnyRole(String... rolesToCheck) {
        var grantedRoles = this.getRoles();
        return Stream.of(rolesToCheck).anyMatch(role -> grantedRoles.contains(role));
    }
}
```

Thymeleaf

- ${@**auth.username**} *hoặc* *${@auth.getUsername()}*
- ${@**auth.roles**} *hoặc* *${@auth.getRoles()}*
- @{@**auth.authenticated**} *hoặc* *@{@auth.isAuthenticated()}*
- ${@**auth.hasAnyRole**(...)}

SECURITY CONFIGURATION

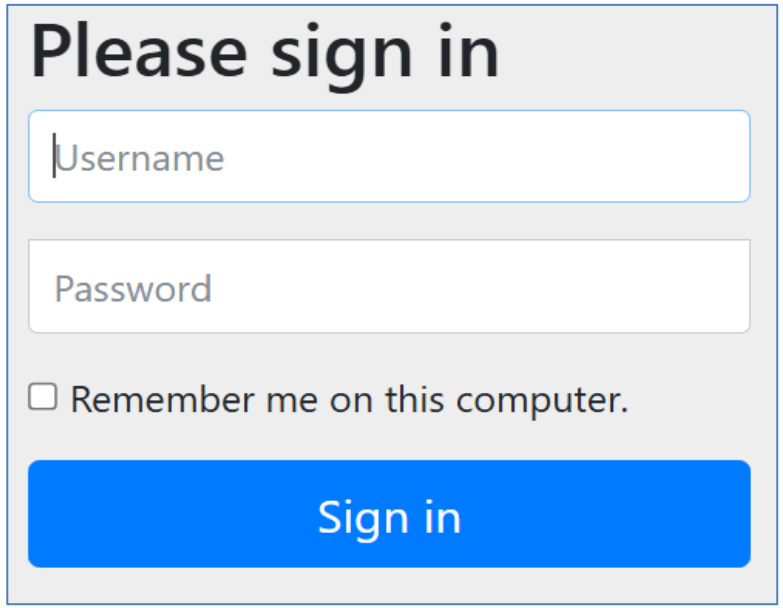
```java
@Configuration
@EnableWebSecurity
public class SecurityConfig {
    @Bean
    public PasswordEncoder passwordEncoder() { // PasswordEncoder
        return PasswordEncoderFactories.createDelegatingPasswordEncoder();
    }
    @Bean
    public UserDetailsService userDetailsService(PasswordEncoder pe) { // User Source
        String pass = pe.encode("123");
        UserDetails user1 = User.withUsername("user@gmail.com").password(pass).roles().build();
        UserDetails user2 = User.withUsername("admin@gmail.com").password(pass).roles().build();
        UserDetails user3 = User.withUsername("both@gmail.com").password(pass).roles().build();
        return new InMemoryUserDetailsManager(user1, user2, user3);
    }
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception { // Authorize
        ... cấu hình bảo mật ở đây
        return http.build();
    }
}
```

SECURITY CONFIGURATION

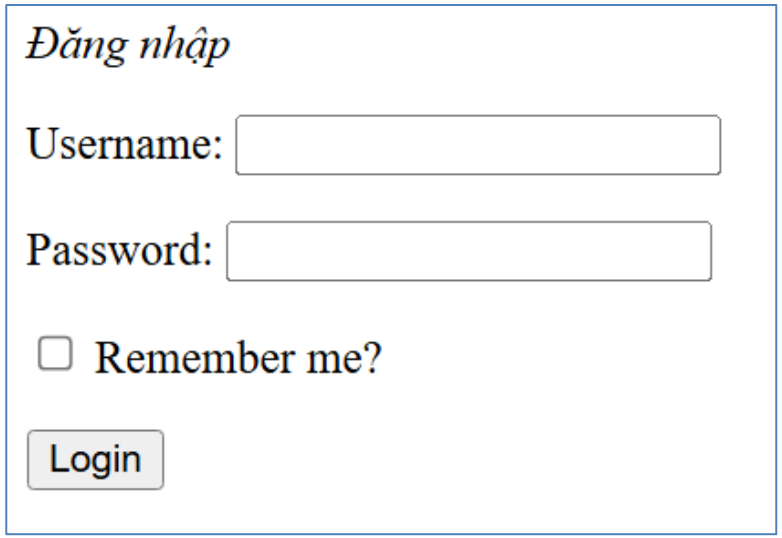
```
@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        // Bỏ cấu hình mặc định CSRF và CORS
        http.csrf(config -> config.disable()).cors(config -> config.disable());
        // Phân quyền sử dụng
        http.authorizeHttpRequests(config -> {
                config.requestMatchers("/poly/**").authenticated();
                config.anyRequest().permitAll();
        });
        // Form đăng nhập mặc định
        http.formLogin(config -> config.permitAll());
        // Ghi nhớ tài khoản
        http.rememberMe(config -> {
                config.tokenValiditySeconds(3*24*60*60);
        });
        return http.build();
}
```

CUSTOM LOGIN FORM

- Spring Security xác thực thông qua form đăng nhập mặc được được cung cấp sẵn hoặc tùy biến do lập trình viên thiết kế



Form đăng nhập mặc định



Form đăng nhập tùy biến

- Với form đăng nhập tùy biến thì bạn cần cấu hình các thông số kỹ thuật theo yêu cầu của Spring Security

CONTROLLER & VIEW

```
@Controller
public class LoginController {
    @RequestMapping("/login/{action}")
    public String login(Model model, @PathVariable("action") String action) {
        switch(action) {
            case "form" -> model.addAttribute("message", "Đăng nhập");
            case "success" -> model.addAttribute("message", "Đăng nhập thành công");
            case "failure" -> model.addAttribute("message", "Sai thông tin đăng nhập");
            case "exit" -> model.addAttribute("message", "Đăng xuất thành công");
        }
        return "login";
    }
}
```

/login/*form*
/login/*success*
/login/*failure*
/login/*exit*

```
<i th:text="${message}"></i>
<form action="/login/check" method="post">
    <p>Username: <input name="username">
    <p>Password: <input name="password">
    <p><input name="remember-me" type="checkbox"> Remember me?
    <p><button>Login</button>
</form>
```

CUSTOM LOGIN FORM

```
// Form đăng nhập
http.formLogin(config -> {
        config.loginPage("/login/form");
        config.loginProcessingUrl("/login/check");
        config.defaultSuccessUrl("/login/success");
        config.failureUrl("/login/failure");
        config.permitAll();
        config.usernameParameter("username");
        config.passwordParameter("password");
});
```

Địa chỉ hiển thị form đăng nhập

Địa chỉ submit form đăng nhập

Địa chỉ xử lý đăng nhập thành công

Địa chỉ xử lý đăng nhập thất bại

username và password là các tên mặc định nên có thể bỏ

CUSTOM LOGOUT & REMEMBER ME

```
// Ghi nhớ tài khoản
http.rememberMe(config -> {
        config.tokenValiditySeconds(3*24*60*60);
        config.rememberMeCookieName("remember-me");
        config.rememberMeParameter("remember-me");
});
// Đăng xuất
http.logout(config -> {
        config.logoutUrl("/logout");
        config.logoutSuccessUrl("/login/exit");
        config.clearAuthentication(true);
        config.invalidateHttpSession(true);
        config.deleteCookies("remember-me");
});
```

Remember-me là tên mặc định nên có thể bỏ

Địa chỉ đăng xuất

Địa chỉ xử lý đăng xuất thành công

Các xử lý mặc định

CÁC THÀNH PHẦN SPRING SECURITY

UserDetails

- UserDetails là interface định nghĩa mô hình thông tin của người sử dụng đăng nhập.

```
public interface UserDetails extends Serializable {
    String getUsername(); // tên đăng nhập
    String getPassword(); // mật khẩu
    Collection<? extends GrantedAuthority> getAuthorities(); // vai trò
    boolean isEnabled(); // Kích hoạt chưa?
    boolean isAccountNonExpired(); // Không bao giờ hết hạn
    boolean isAccountNonLocked(); // Không bao giờ khóa
    boolean isCredentialsNonExpired(); // Mật khẩu không bao giờ hết hạn
}
```

Các thuộc tính trạng thái

- User là class thực hiện theo interface UserDetails với các thuộc tính trạng thái là true
  - Ví dụ:

```
UserDetails user = User.withUsername(username).password(password).build();
```

UserDetailsService

- UserDetailsService là interface định nghĩa hoạt động tải dữ liệu tài khoản người sử dụng vào hệ thống Security
- InMemoryUserDetailsManager và JdbcUserDetailsManager là các class thực thi theo UserDetailsService, trong đó:
  - InMemoryUserDetailsManager quản lý nguồn dữ liệu user trực tiếp trong bộ nhớ.
    - Ví dụ: new InMemoryUserDetailsManager(user1, user2, user3)
  - JdbcUserDetailsManager quản lý nguồn dữ liệu user từ CSDL (giới thiệu bài sau).

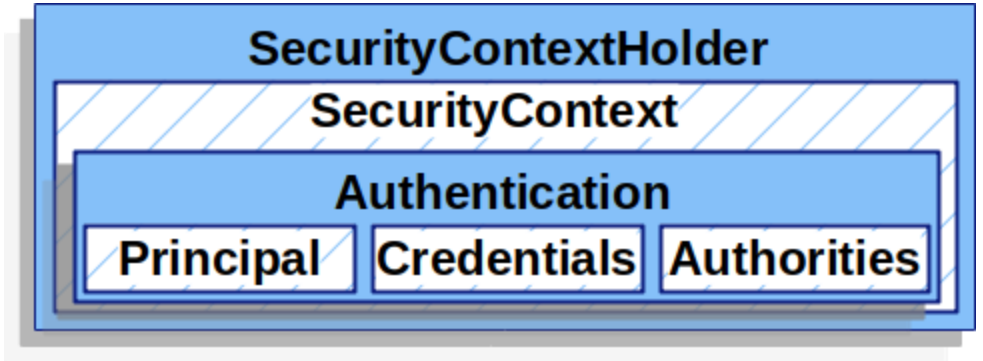
UserDetailsService

- PasswordEncoder là interface định nghĩa hoạt động mã hóa mật khẩu và so sánh mật khẩu đã mã hóa

```
public interface PasswordEncoder {
    // Mã hóa mật khẩu
    String encode(CharSequence rawPassword);
    // So sánh mật khẩu
    boolean matches(CharSequence rawPassword, String encodedPassword);
}
```

- PasswordEncoderFactories là lớp tiện ích cung cấp các phương thức tạo ra các đối tượng thực hiện theo tiêu chuẩn của PasswordEncoder.
- DelegatingPasswordEncoder là class thực hiện theo PasswordEncoder

TỔ CHỨC QUẢN LÝ THÔNG TIN ĐĂNG NHẬP



- SecurityContextHolder là lớp tiện ích cung cấp phương thức getContext() để đọc lấy SecurityContext
- SecurityContext cung cấp phương thức getAuthentication() để đọc lấy đối tượng chưa thông tin đăng nhập hiện tại.
- Authentication chứa 3 thông tin quan trọng gồm
  - Principal: thường là UserDetails
  - Credentials: thường là mật khẩu
  - Authorites: là các vai trò của người đăng nhập

TỔ CHỨC QUẢN LÝ THÔNG TIN ĐĂNG NHẬP

- Đoạn mã sau đây đọc lấy thông tin của người sử dụng sau khi đăng nhập

```
// Đọc lấy đối tượng Authentication
Authentication auth = SecurityContextHolder.getContext().getAuthentication();
// Đọc lấy đối tượng UserDetails
UserDetails user = (UserDetails) auth.getPrincipal();
// Đọc lấy Username
String username = auth.getName();
// Đọc lấy vai trò
List<String> roles = auth.getAuthorities().stream()
                         .map(au -> au.getAuthority().substring(5)).toList();
```

SUMMARY

- ☑ Giới thiệu Web Security
- ☑ Spring Authentication
- ☑ Xây dựng Spring Bean AuthService xử lý thông tin đăng nhập
- ☑ Tùy biến Login Form
- ☑ Các thành phần Spring Security
  - ☑ UserDetails, UserDetailsService, InMemoryUserDetailsManager
  - ☑ PasswordEncoder, PasswordEncoderFactories, DelegatingPasswordEncoder
  - ☑ SecurityFilterChain


TIN HỌC VĂN PHÒNG QUỐC TẾ THEO CHUẨN ICDL
Syllabus 5 - Office 2007
LUẬT KINH TẾ
LUẬT KINH TẾ
KỸ THUẬT SOẠN THẢO VÀ TRÌNH BÀY VĂN BẢN
TIN HỌC VĂN PHÒNG QUỐC TẾ THEO CHUẨN ICDL
Syllabus 5 - Office 2007
Cảm ơn