FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

SPRING BOOT SECURITY 3

GIẢNG VIÊN: NGUYỄN NGHIỆM

AGENDA

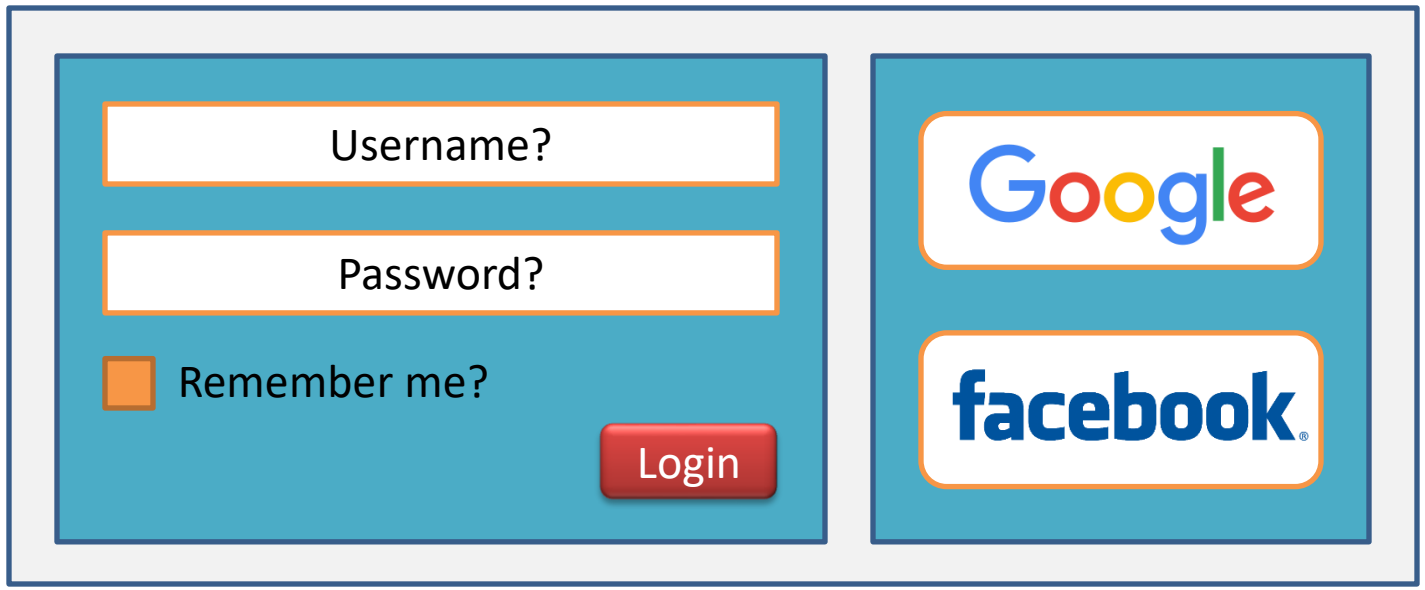
- Xác thực tự mạng xã hội
- Sử dụng @PreAuthorize
- Lập trình bảo vệ tùy biến
- Tùy biến giao diện với #authentication và Thymeleaf-extras
- Sử dụng AuthService

OAuth2

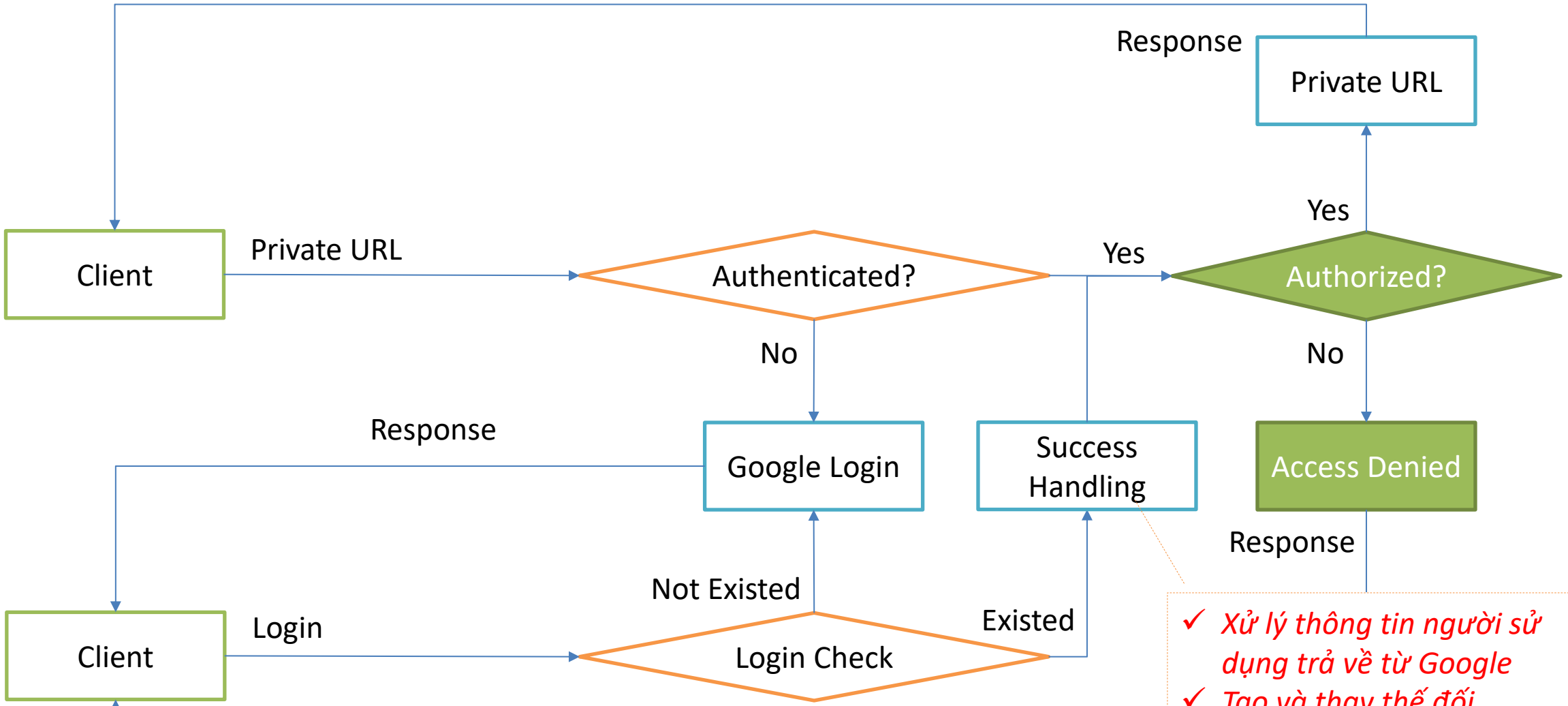
WHAT IS OAUTH?

- ***OAuth*** *(Open Authorization) là một framework ủy quyền chuẩn mở dựa trên nền tảng internet.*
- ***OAuth*** *(phát âm là "oh-auth") cho phép các dịch vụ của bên thứ ba như Facebook và Google..., cung cấp thông tin tài khoản của người sử dụng mà không để lộ.*

OAuth2 Case Study



Google Authentication Flow



CÔNG VIỆC CẦN THỰC HIỆN

- Tạo ứng dụng trên tài khoản Google, lấy ClientID và Secret và khai báo vào application.properties để website tương tác với Google trong quá trình xác thực
- Khai báo thư viên cần thiết (OAuth2)
- Cấu hình đăng nhập từ Google
- Viết mã xử lý sau khi xác thực thành công
  - Đọc hồ sơ người sử dụng từ Google (Principal)
  - Tạo đối tượng Authentication mới từ thông tin người dung đọc từ hồ sơ
  - Thay thế đối tượng Authentication hiện tại bằng đối tượng Authentication mới tạo ra
  - Các phần việc tùy chọn: Truy vấn thông tin bổ sung từ CSDL nếu chưa tồn tại thì thêm mới User vào CSDL nếu cần
  - Chuyển về Private URL đã yêu cầu trước đó hoặc trang chủ nếu vào đăng nhập trực tiếp

Client ID và Secret

CLIENT ID & CLIENT SECRET

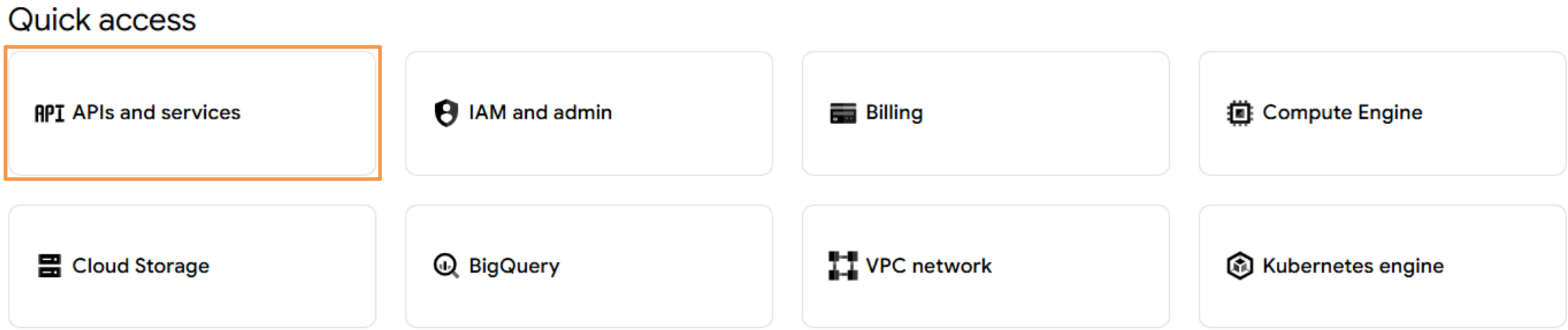
application.properties

```
spring.security.oauth2.client.registration.google.client-id=<<client-id>>
spring.security.oauth2.client.registration.google.client-secret=<<client-secret>>
```

- Khai báo các thông số Client ID và Client Secret vào application.properties để ứng dụng web kết nối với Google
- Client ID và Client Secret được cung cấp bởi Google, thực hiện theo hướng dẫn các slide sau để tạo và chép lấy id và secret

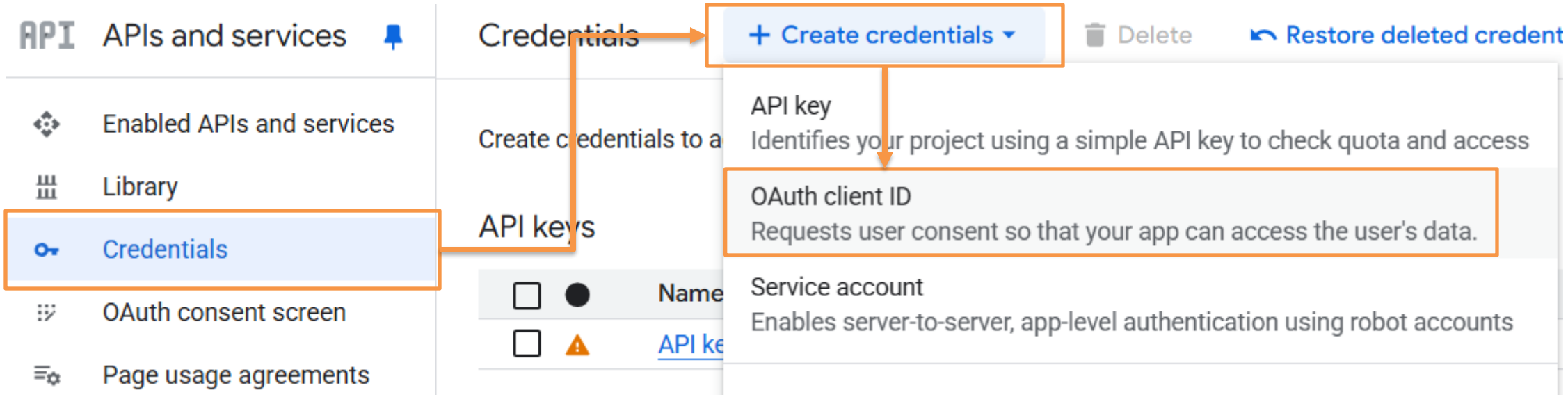
Tạo và sao chép Client ID và Secret 1

- Bước 1: Đăng nhập Google Mail
- Bước 2: Truy xuất https://console.cloud.google.com
- Bước 3: Click [API Apis and Services] trong phần Quick Access



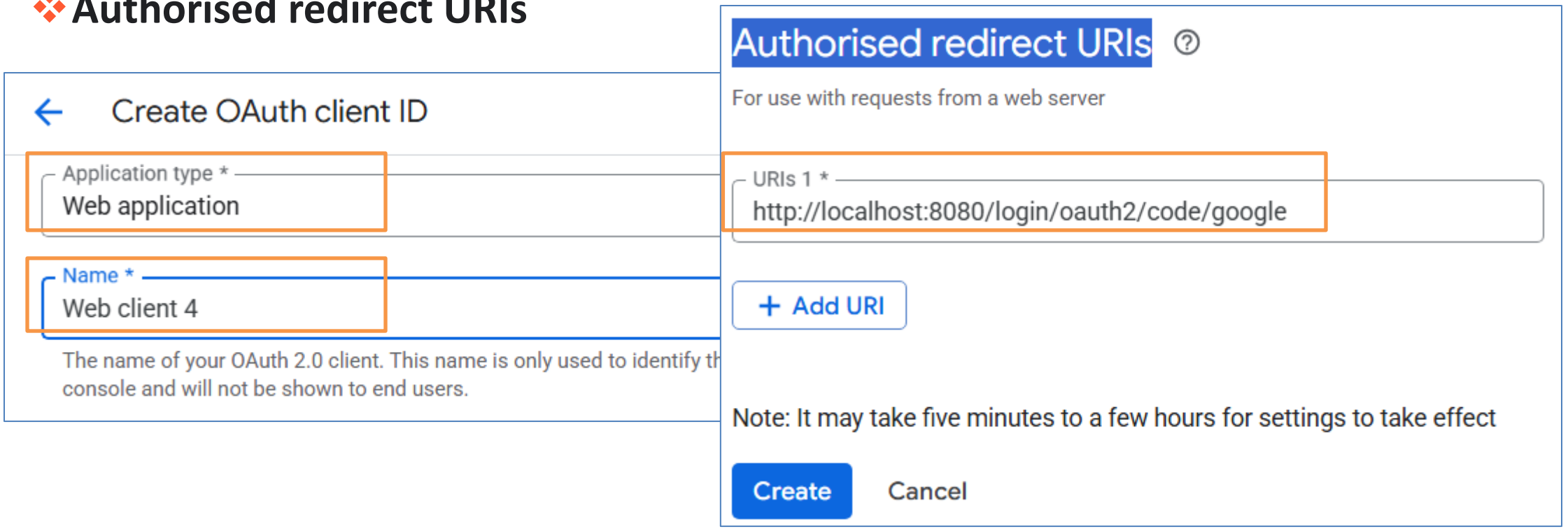
TẠO VÀ SAO CHÉP CLIENT ID VÀ SECRET 2

- Bước 4: [Credentials>Create Credentails>OAuth2 Client ID]



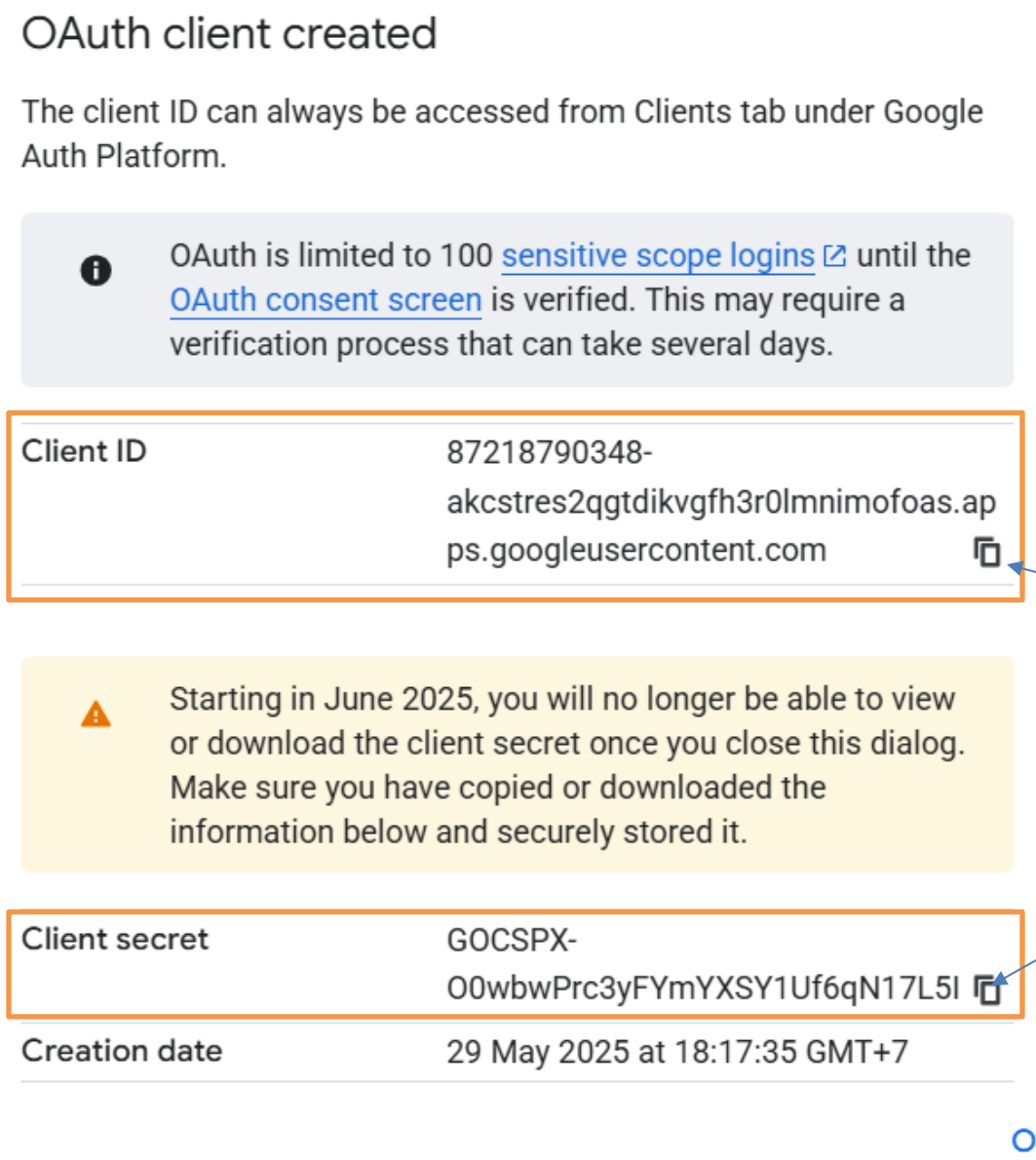
Tạo và sao chép Client ID và Secret 3

- Bước 5: Nhập 3 thông số:
  - **Application type**
  - **Name**
  - **Authorised redirect URIs**



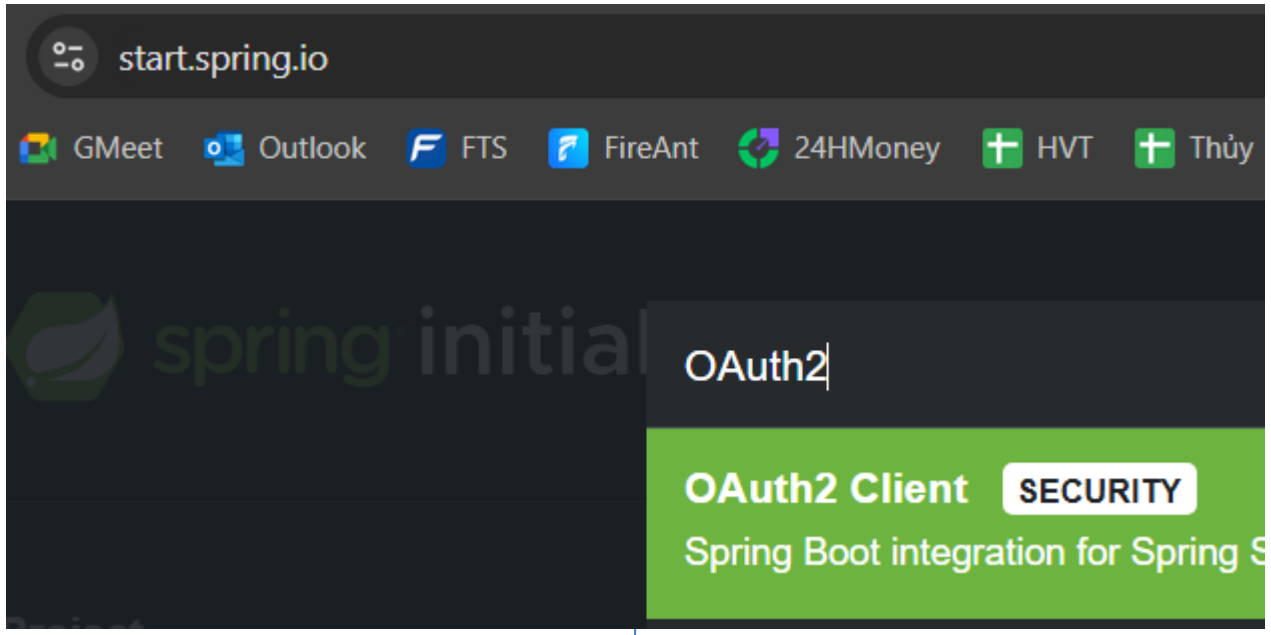
TẠO VÀ SAO CHÉP CLIENT ID VÀ SECRET 4

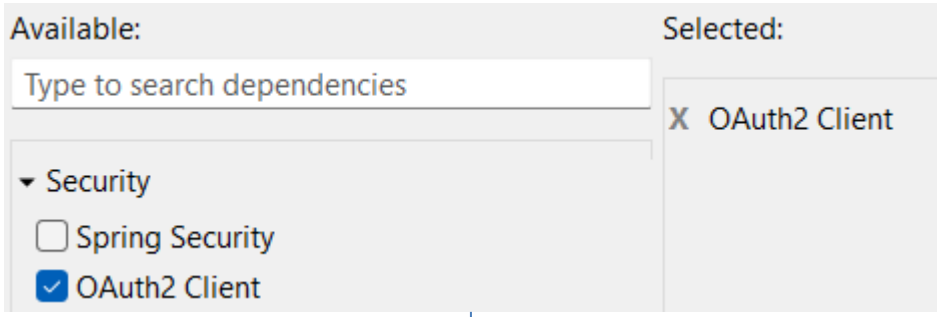
- Bước 6: Click [Create] chép Client ID, Client Secret và khai báo vào application.properties



Sao chép Client ID và Secret

KHAI BÁO THƯ VIÊN CẦN THIẾT (OAUTH2)





pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-oauth2-client</artifactId>
</dependency>
```

CẤU HÌNH ĐĂNG NHẬP TỪ GOOGLE

```
http.oauth2Login(login -> {
        login.permitAll();
        login.successHandler((request, response, authentication) -> {
                // Đọc UserDetails chứa thông tin người dùng từ Google
                ...
                // Tạo đối tượng Authentication mới và thay thế đối tượng Authentication hiện tại
                ...
                // Chuyển về trang yêu cầu bảo mật hoặc trang chủ
                ...
        });
});
```

- Bổ sung đoạn mã trên vào file cấu hình để đăng nhập từ Google
- Phương thức xử lý sau khi xác thực thành công nhận 3 đối số request, response và authentication.

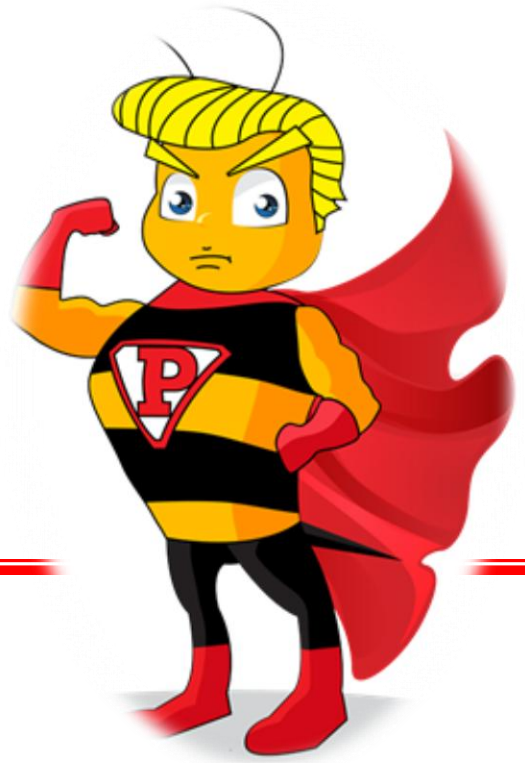
VIẾT MÃ XỬ LÝ SAU KHI XÁC THỰC THÀNH CÔNG

```
// Đọc UserDetails chứa thông tin người dùng từ Google
DefaultOidcUser user = (DefaultOidcUser) authentication.getPrincipal();
String username = user.getEmail();
String pass = pe.encode("");
String role = "OAUTH";

// Tạo đối tượng Authentication mới và thay thế đối tượng Authentication hiện tại
UserDetails u = User.withUsername(username).password(password).roles(role).build();
Authentication newauth = new UsernamePasswordAuthenticationToken(u, pass, u.getAuthorities());
SecurityContextHolder.getContext().setAuthentication(newauth);

// Chuyển về trang yêu cầu bảo mật hoặc trang chủ
HttpSession session = request.getSession();
String attr = "SPRING_SECURITY_SAVED_REQUEST";
DefaultSavedRequest req = (DefaultSavedRequest) session.getAttribute(attr);
String redirectUrl = (req == null) ? "/" : req.getRedirectUrl();
response.sendRedirect(redirectUrl);
```

LINK ĐĂNG NHẬP

Liên kết dẫn đến trang đăng nhập của Google

```
<a href="/oauth2/authorization/google">Google</a>
```



Authorizing Annotation

AUTHORIZING ANNOTATION

- Spring Security cho phép sử dụng các Annotation để phân quyền truy xuất trực tiếp trên các phương thức điều khiển của các Controller.
- @PreAuthorize() được chú thích trực tiếp trên các phương thức điều khiển dưới 3 dạng sau
  - @PreAuthorize("isAuthenticated()")
  - @PreAuthorize("hasRole('USER')")
  - @PreAuthorize("hasAnyRole('USER', 'ADMIN')")
  - @PreAuthorize("hasAuthority('ROLE_USER')")
  - @PreAuthorize("hasAnyAuthority('ROLE_USER', 'ROLE_ADMIN')")

PHÂN QUYỀN TRUY XUẤT

```
@Configuration
@EnableWebSecurity
@EnableMethodSecurity(prePostEnabled = true)
public class SecurityConfig {
    @Bean public PasswordEncoder passwordEncoder() {...}
    @Bean public UserDetailsService userDetailsService(DataSource dataSource) {...}
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        // Bỏ cấu hình mặc định CRSF và CORS
        http.csrf(csrf -> csrf.disable()).cors(cors -> cors.disable());
        // Phân quyền sử dụng
        http.authorizeHttpRequests(req -> {
            req.anyRequest().permitAll();
        });
        // Từ chối truy xuất nếu vai trò ko phù hợp
        http.exceptionHandling(denied -> denied.accessDeniedPage("/unauthorized.html"));
        // Form đăng nhập mặc định /login
        http.formLogin(login -> login.permitAll());
        return http.build();
    }
}
```

Kích hoạt chế độ phân quyền trên phương thức điều khiển

Cho phép truy xuất tất cả các tài nguyên

PHÂN QUYỀN TRUY XUẤT TRÊN PHƯƠNG THỨC ĐIỀU KHIỂN

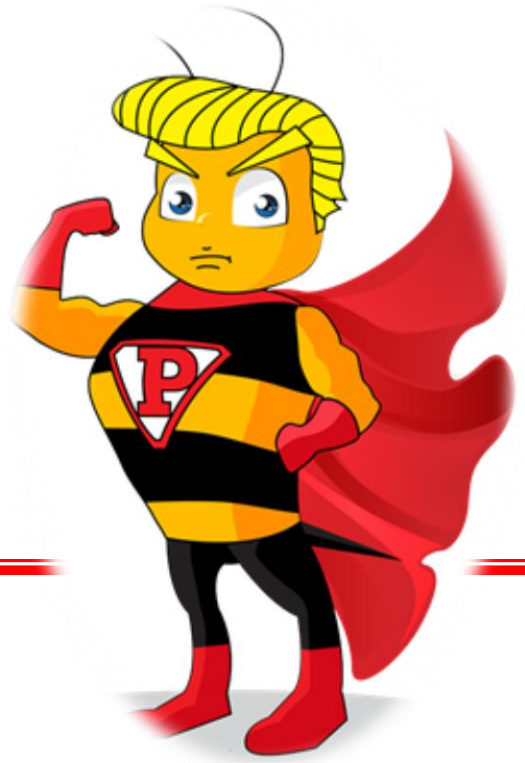
```
@Controller
public class MyController {
	@GetMapping({"/", "/poly/url0"})
	public String method0(Model model) {
		model.addAttribute("message", "/poly/url0 => method0()");
		return "page";
	}
	@PreAuthorize("isAuthenticated()") @GetMapping("/poly/url1")
	public String method1(Model model) {
		model.addAttribute("message", "/poly/url1 => method1()");
		return "page";
	}
	@PreAuthorize("hasRole('USER')") @GetMapping("/poly/url2")
	public String method2(Model model) {...}
	@PreAuthorize("hasRole('ADMIN')") @GetMapping("/poly/url3")
	public String method3(Model model) {...}
	@PreAuthorize("hasAnyRole('USER', 'ADMIN')") @GetMapping("/poly/url4")
	public String method4(Model model) {...}
}
```

PROGRAMMATICALLY SECURITY

- Cấu hình phân quyền truy xuất hoặc sử dụng @PreAuthorize() được sử dụng để bảo vệ tài nguyên URL hoặc toàn bộ phương thức điều khiển
- Một vài tình huống cần bảo vệ một cách mềm dẻo hơn là từng đoạn mã trong các phương thức điều khiển. Để làm được điều này chúng ta chỉ có thể dung phương pháp lập trình dựa vào thông tin được cung cấp từ đối tượng Authentication.
- Bạn có thể tham chiếu đến đối tượng này trong Controller theo 3 cách:
  - Tiêm (inject) Authentication
  - SecurityContextHolder.getContext().getAuthentication()
  - Tiêm HttpServletRequest và sử dụng phương thức getUserPrincipal()

THAM CHIẾU ĐẾN ĐỐI TƯỢNG AUTHENTICATION

```
@GetMapping("/poly/url5")
public String method5() {
        var authentication = SecurityContextHolder.getContext().getAuthentication();
        // Lập trình security dựa trên biến authentication ở đây
        return "page";
}
@GetMapping("/poly/url6")
public String method6(Authentication authentication) {...}
@GetMapping("/poly/url7")
public String method7(HttpServletRequest request) {
        var authentication = (Authentication)request.getUserPrincipal();
        ...
}
```



TÙY BIẾN GIAO DIỆN

SECURITY TRONG THYMELEAF

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<body>
        <ul>
                <li th:text="${#authentication.name}"></li>
                <li th:text="${#authentication.authorities}"></li>
        </ul>
        <ul th:if="${#authentication.authenticated}">
                <li th:if="${#request.isUserInRole('ADMIN')}">Admin</li>
                <li th:if="${#request.isUserInRole('ADMIN')} OR ${#request.isUserInRole('USER')}">
                        Admin or User
                </li>
                <li th:if="${#request.isUserInRole('GUEST')}">Guest</li>
        </ul>
</body>
</html>
```

SECURITY VỚI THYMELEAF-EXTRAS

```
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity5">
<body>
    <ul>
        <li sec:authentication="name"></li>
        <li sec:authentication="authorities"></li>
    </ul>
    <ul sec:authorize="isAuthenticated()">
        <li sec:authorize="hasRole('ADMIN')">Admin</li>
        <li sec:authorize="hasAnyRole('ADMIN', 'USER')">Admin or User</li>
        <li sec:authorize="hasRole('GUEST')">Guest</li>
    </ul>
</body>
</html>
```

```
<dependency>
    <groupId>org.thymeleaf.extras</groupId>
    <artifactId>thymeleaf-extras-springsecurity6</artifactId>
</dependency>
```

Sử dụng Spring bean AuthService

Sử dụng bean AuthService

- Bean AuthService được tạo ra để đơn giản hóa việc lập trình security trong cả Controller và Thymeleaf
- Trong Controller bạn chỉ cần tiêm vào và sử dụng các phương thức đã viết sẵn để bảo vệ các đoạn mã bạn cần
- Trong Thymeleaf bạn sử dụng biểu thức ${@auth.<<phương thức>>} để thực hiện việc ẩn hay hiển thiện giao diện một cách tùy biến

SỬ DỤNG BEAN AUTHSERVICE TRONG CONTROLLER

```
@Autowired
AuthService authService;
```

Tiêm bean vào Controller

```
@GetMapping("/poly/url5")
public String method5() {
        var username = authService.getUsername();
        var roles = authService.getRoles();
        //...
        if(authService.isAuthenticated()) {
                //...
        }
        if(authService.hasAnyRole("USER", "ADMIN")) {
                //...
        }
        return "page";
}
```

Gọi các phương thức để xử lý công việc

Sử dụng bean AuthService trong Thymeleaf

```
<div th:switch="${@auth.authenticated}">
    <ul th:case="true">
        <li>Username: <b th:text="${@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>
```

isAuthenticated()

getUsername()

getRoles()

SUMMARY

- ☑ Xác thực tự mạng xã hội
- ☑ Sử dụng @PreAuthorize
- ☑ Lập trình bảo vệ tùy biến
- ☑ Tùy biến giao diện với #authentication và Thymeleaf-extras
- ☑ Sử dụng AuthService


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