



FPT POLYTECHNIC



www.poly.edu.vn



Conceive Design Implement Operate

SPRING BOOT SECURITY 3

GIẢNG VIÊN: NGUYỄN NGHIỆM

- ❑ 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



- ❑ **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.

A login form interface with a light blue background and a dark blue border. The form is divided into two main sections. The left section contains three input fields: 'Username?' and 'Password?' (both with orange borders), and a checkbox labeled 'Remember me?' (with an orange square). Below these is a red 'Login' button. The right section contains two social login buttons: 'Google' (with the multi-colored logo) and 'facebook' (with the blue logo and a registered trademark symbol).

Username?

Password?

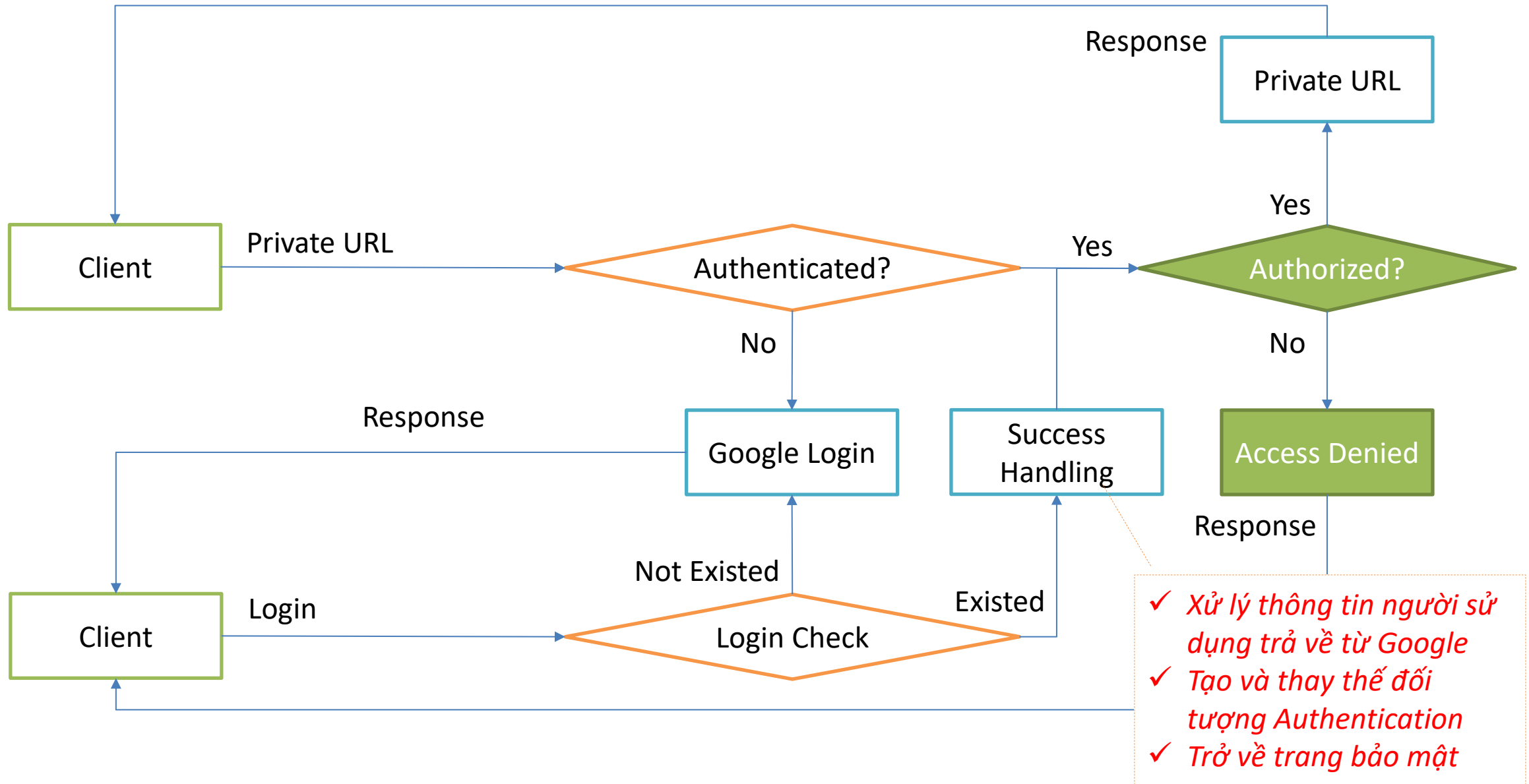
☐ Remember me?

Login

Google

facebook®

GOOGLE AUTHENTICATION FLOW



- ❑ 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 viện 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 dùng 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

application.properties

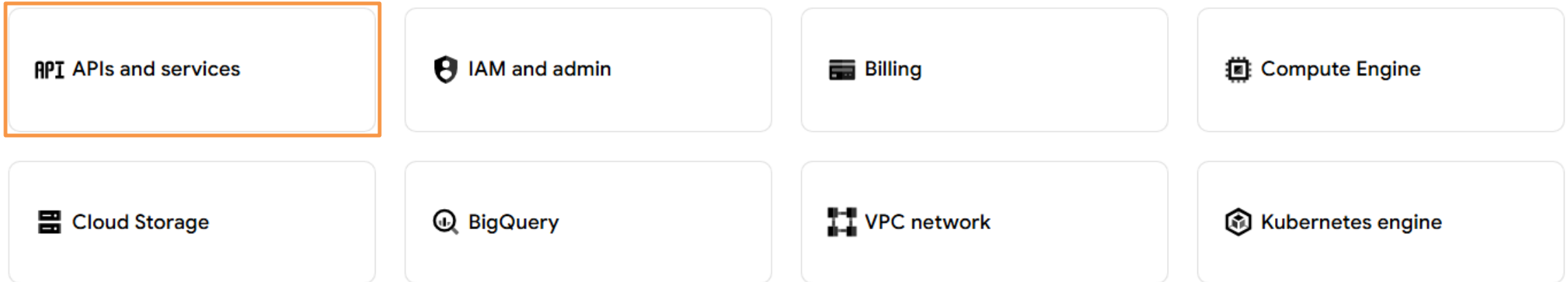
spring.security.oauth2.client.registration.google.client-id=<<client-id>>

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

- ❑ 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

Quick access



❑ Bước 4: [Credentials>Create Credentials>OAuth2 Client ID]

The screenshot shows the Google Cloud Platform interface for managing APIs and services. On the left sidebar, the 'Credentials' option is highlighted. The main panel shows the 'Credentials' section with a '+ Create credentials' button. A dropdown menu is open from this button, showing three options: 'API key', 'OAuth client ID', and 'Service account'. The 'OAuth client ID' option is highlighted with an orange box. Arrows indicate the navigation path: from the 'Credentials' menu item in the sidebar to the '+ Create credentials' button, and then to the 'OAuth client ID' option in the dropdown menu.

APIs and services

- Enabled APIs and services
- Library
- Credentials**
- OAuth consent screen
- Page usage agreements

Credentials

+ Create credentials ▾

Delete Restore deleted credential

Create credentials to a

API keys

- ☐ API key
Identifies your project using a simple API key to check quota and access
- ☒ **OAuth client ID**
Requests user consent so that your app can access the user's data.
- ☐ Service account
Enables server-to-server, app-level authentication using robot accounts

☐ Name

☐ API key

❑ Bước 5: Nhập 3 thông số:

❖ **Application type**

❖ **Name**

❖ **Authorised redirect URIs**

← Create OAuth client ID

Application type *
Web application

Name *
Web client 4

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

Authorised redirect URIs ⓘ

For use with requests from a web server

URIs 1 *
http://localhost:8080/login/oauth2/code/google

+ Add URI

Note: It may take five minutes to a few hours for settings to take effect

Create Cancel

TẠO VÀ SAO CHÉP CLIENT ID VÀ SECRET 4

OAuth client created

The client ID can always be accessed from Clients tab under Google Auth Platform.

i OAuth is limited to 100 [sensitive scope logins](#) until the [OAuth consent screen](#) is verified. This may require a verification process that can take several days.

Client ID	87218790348- akcstres2qgtdikvgfh3r0lmnimofaoas.ap ps.googleusercontent.com
-----------	--

⚠ Starting in June 2025, you will no longer be able to view or download the client secret once you close this dialog. Make sure you have copied or downloaded the information below and securely stored it.

Client secret	GOCSPX- 00wbwPrc3yFYmYXSY1Uf6qN17L5I
---------------	---

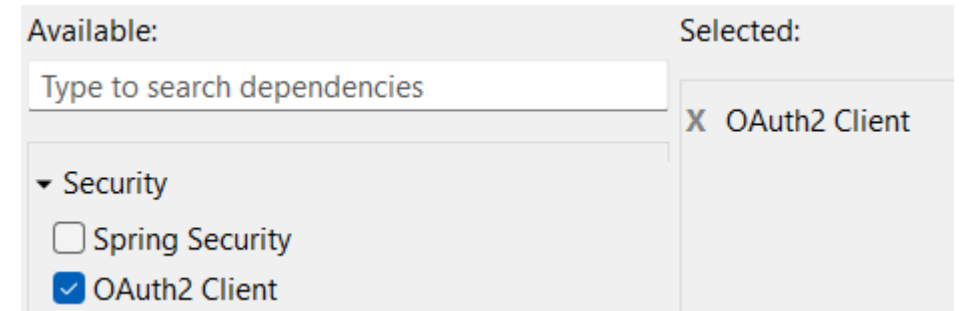
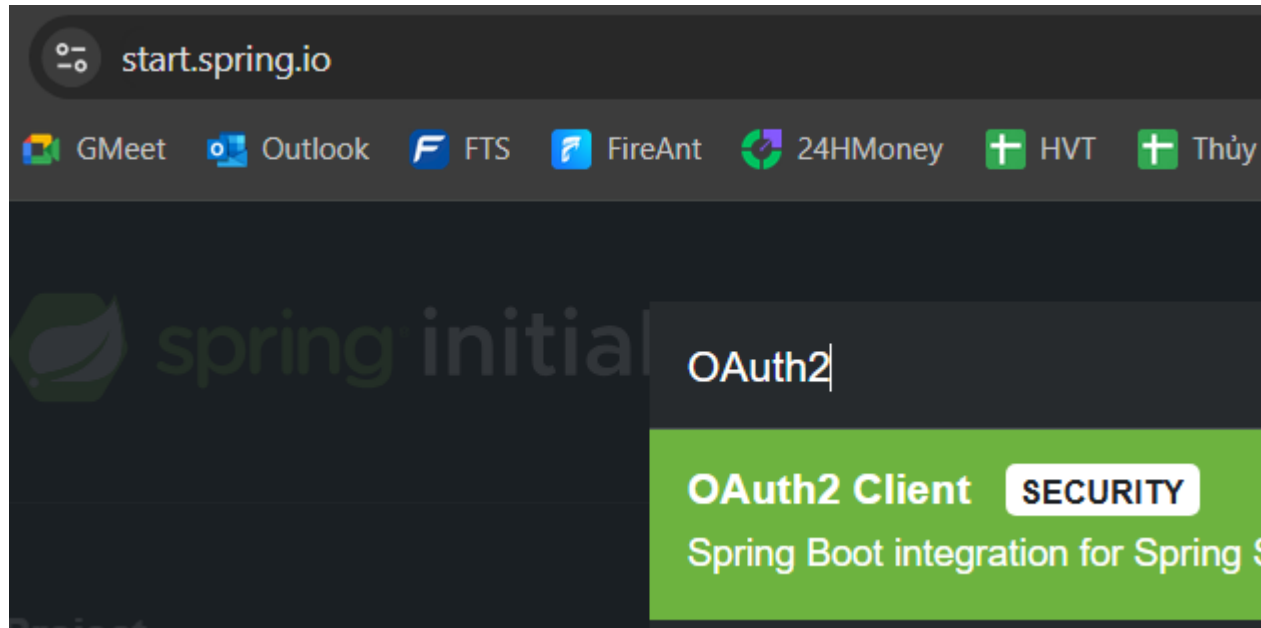
Creation date	29 May 2025 at 18:17:35 GMT+7
---------------	-------------------------------

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

Sao chép Client ID và Secret

OK

KHAI BÁO THƯ VIỆN CẦN THIẾT (OAUTH2)



pom.xml

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

```
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);
```


Please sign in

Sign in

Login with OAuth 2.0

Google

Liên kết dẫn đến trang đăng nhập của Google

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



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')")

@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 CSRF 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

@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) {...}  
}
```

- ❑ 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

```
<!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>
```

```
<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

- ❑ Bean AuthService được tạo ra để đơn giản hóa việc lập trình security trong cả Controller và Thymeleaf
- ❑ Trong Controller bạn chỉ cần thêm 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 thị 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 THYMELAF

```
<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()

- ✓ 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





Cảm ơn