🏘️

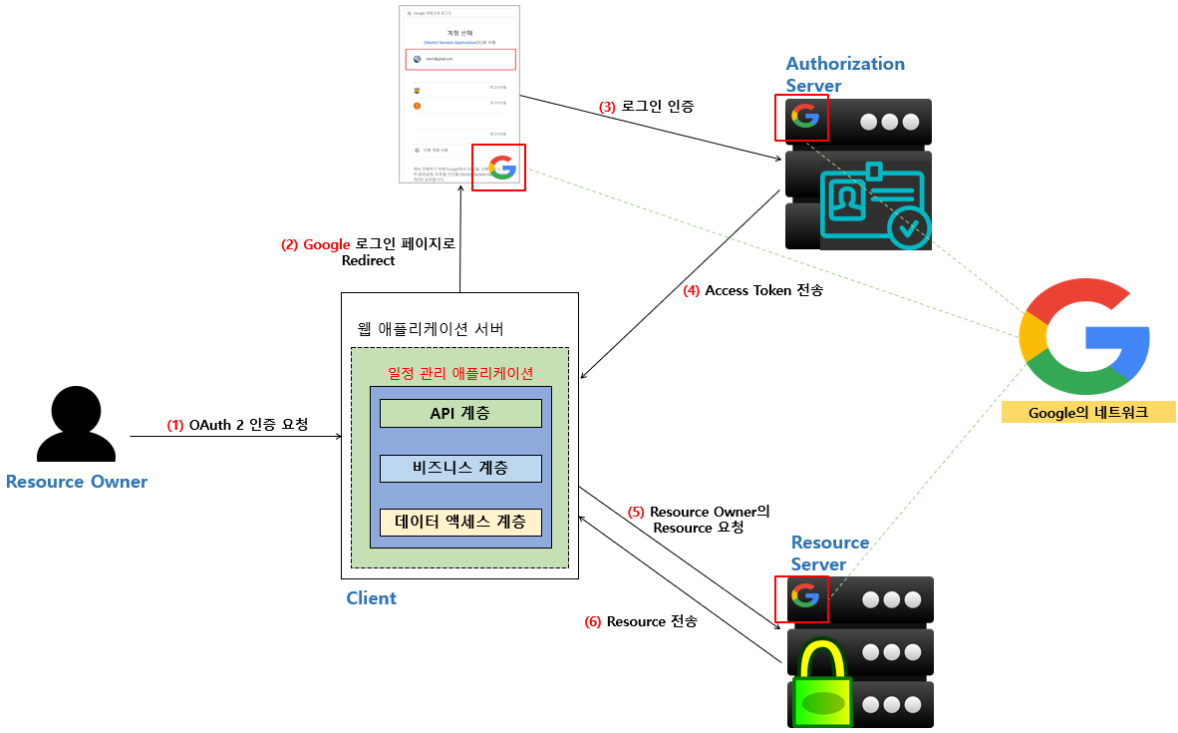
라이브 세션-2022.09.29(목)-Spring Security-OAuth 2

✅ OAuth 2란?

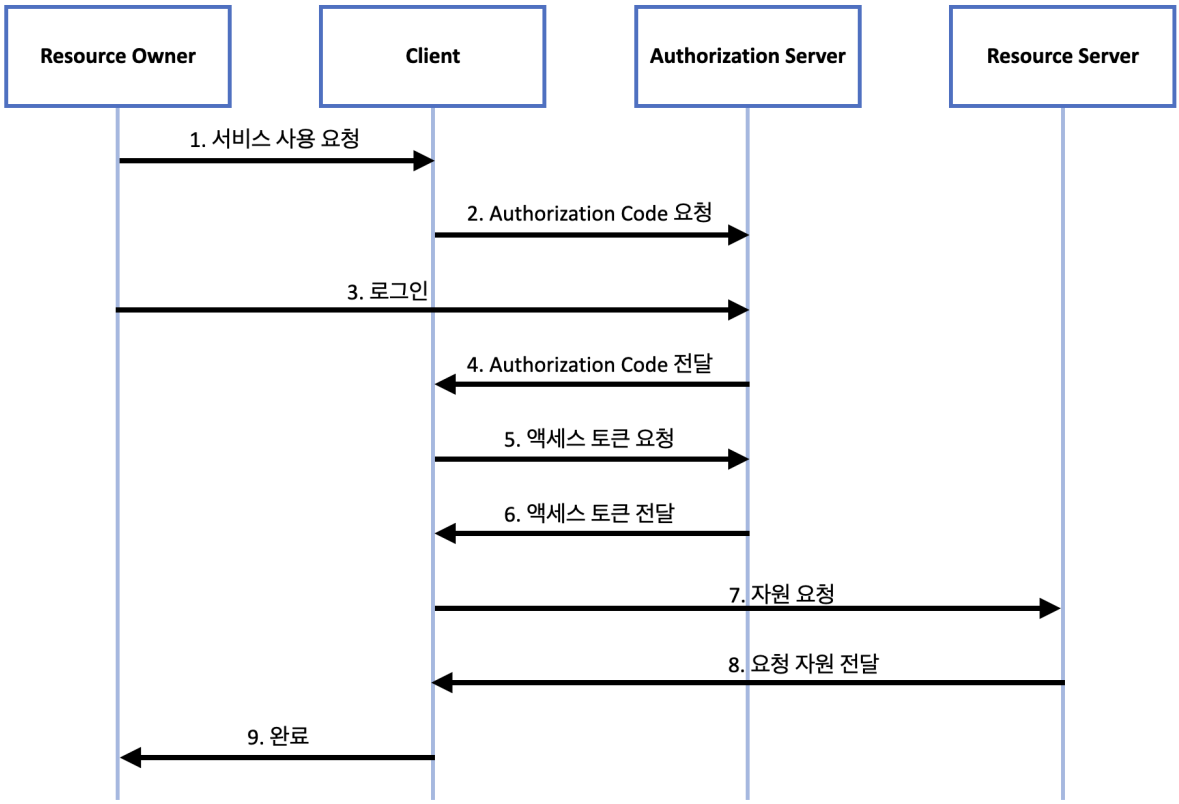
- 공식적인 정의
  - **OAuth 2**는 **사용자 정보를 보유하고 있는 신뢰할 만한 써드 파티 애플리케이션(GitHub, Google, Facebook 등)에서 사용자의 인증을 대신 처리**해 주고 Resource에 대한 자격 증명용 토큰을 발급한 후, Client가 해당 토큰을 이용해 써드 파티 애플리케이션의 서비스를 사용하게 해주는 방식

- 기억하기 쉬운 정의
  - **O(**pen**)**.. 권한(**Auth**orization) 좀 줘!
  - **O(**pen**)**.. 구글(ID Provider)님아 너희 서비스를 사용할 수 있는 권한(**Auth**orization) 좀 줘!
  - **O(**pen**)**.. 구글(ID Provider)님아 **너희 로그인 화면에서 인증**할테니 너희 서비스를 사용할 수 있는 권한(**Auth**orization) 좀 줘!
  - **O(**pen**)**.. 구글(ID Provider)님아 **너희 로그인 화면에서 인증**할테니 너희 서비스를 사용할 수 있는 권한(**Auth**orization)을 **Access Token**으로 좀 줘!

✅ OAuth 2 인증 아키텍처를 구성하는 컴포넌트



✅ OAuth 2 인증 처리 흐름



✅ Spring Security를 이용한 OAuth 2 기본 구현

1️⃣ OAuth 2 Provider에 OAuth 2 클라이언트 등록

- Google API Console 사이트에서 등록
  - https://console.cloud.google.com/apis

2️⃣ build.gradle에 의존성 추가

```
...
...

dependencies {
  ...
```

```
  implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'    // (1)
  implementation 'org.springframework.boot:spring-boot-starter-security'     // (2)
  implementation 'org.springframework.boot:spring-boot-starter-oauth2-client'  // (3)
  ...
}

...
...
```

3️⃣ application.yml에 OAuth 2 클라이언트 정보 추가

```
spring:
  h2:
    console:
      enabled: true
      path: /h2
  datasource:
    url: jdbc:h2:mem:test
  jpa:
    hibernate:
      ddl-auto: create
    show-sql: true
    properties:
      hibernate:
        format_sql: true
  sql:
    init:
      data-locations: classpath*:db/h2/data.sql
  security:
    oauth2:
      client:
        registration:
          google:
            clientId: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx           # (1)
            clientSecret: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx  # (2)

...
...
```

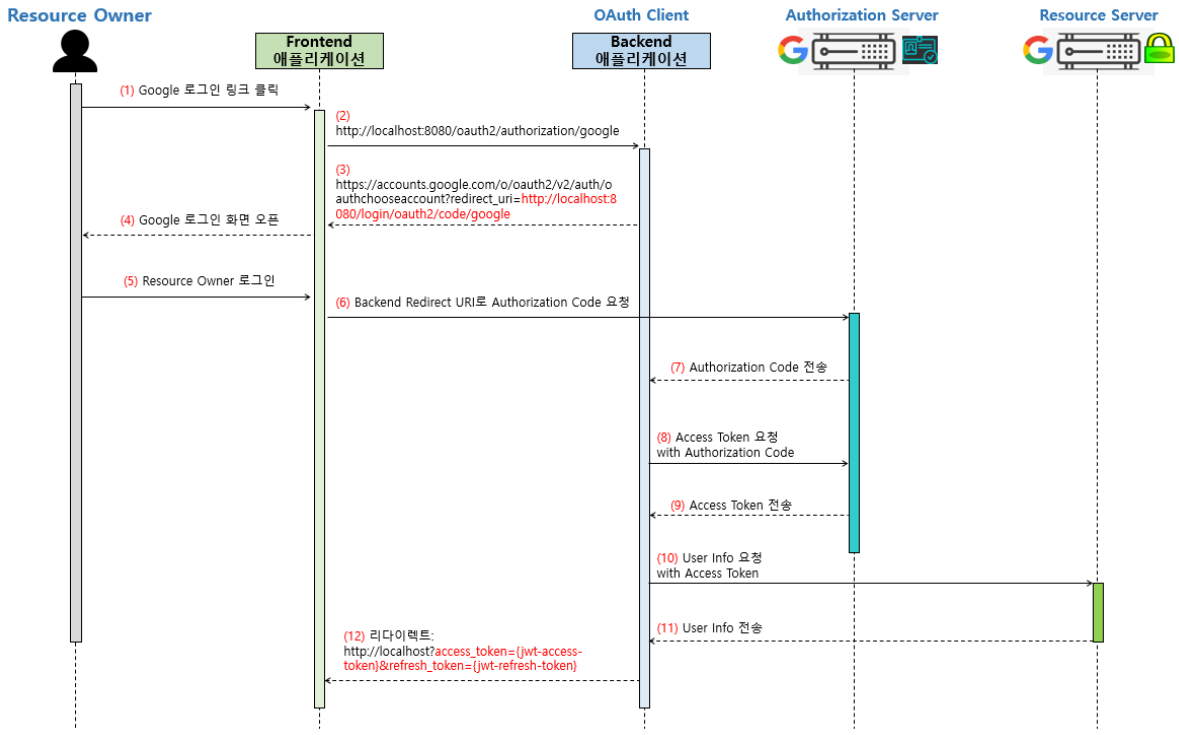
3️⃣ SecurityConfiguration 구성

- 기본적으로 Spring Boot 자동 구성 기능이 알아서 구성해주기 때문에 설정 필요없음

✅ Spring Security를 이용한 OAuth 2 + JWT 구현

- 사용자(Resource Owner)의 로그인 인증은 OAuth 2로 처리하고,
- 인증된 사용자에 대한 자격 증명 부여는 JWT로 처리한다.
- CSR(Client Side Rendering) 방식의 애플리케이션에 적합한 방식

1️⃣ Frontend와 Backend 간의 OAuth 2 + JWT 인증 흐름



2️⃣ Frontend 측 환경 준비

- 아파치 또는 NGINX 웹서버 설치 및 설정
- HTML 소스 코드 작성
- 웹서버에 HTMI 배포
- 웹서버 실행

3️⃣ Backend 애플리케이션에 OAuth 2 인증 기능 적용

- JwtTokenizer 추가
- application.yml 설정
- JwtVerificationFilter 추가
- AuthenticationSuccessHandler 구현
- SecurityConfigiguration 설정
  - 회원 등록 및 수정에 대한 접근 권한 설정 제거
  - oauth2Login()에 AuthenticationSuccessHandler 추가
  - OAuth2LoginAuthenticationFilter 뒤에 JwtVerificationFilter 추가
  - 나머지는 JWT 인증 때 설정과 동일
- 기존의 회원 정보 등록 및 수정에 대한 Controller, DTO, Service 기능 제거

✅ Spring Security의 OAuth 2 컴포넌트 조금 더 알아보기

1️⃣ OAuth2LoginAuthenticationFilter

- OAuth 2 인증을 처리하기 위한 Filter

2️⃣ CommonOAuth2Provider

- 신뢰할 만한 OAuth 2 Provider 목록 제공
- 실질적으로 각 Provider에게 넘겨줘야 되는 Client 정보를 포함한 `ClientRegistration` 객체를 생성한다.

3️⃣ ClientRegistration

- 각 OAuth 2 Provider 사용을 위해 Client에게 필요한 등록 정보

```
public enum CommonOAuth2Provider {

  GOOGLE {

    @Override
    public Builder getBuilder(String registrationId) {
      ClientRegistration.Builder builder = getBuilder(registrationId,
          ClientAuthenticationMethod.CLIENT_SECRET_BASIC, DEFAULT_REDIRECT_URL);
      // Provider가 제공할 Resource 범위
      builder.scope("openid", "profile", "email");

      // Client가 OAuth 2 인증을 위해 Redirect하는 URI(구글의 로그인 인증 화면)
      builder.authorizationUri("https://accounts.google.com/o/oauth2/v2/auth");

      // Client가 Access Token과 Refresh Token을 얻기 위해 호출하는 URI
      builder.tokenUri("https://www.googleapis.com/oauth2/v4/token");

      // JWT 검증용 Key Set을 조회할 수 있는 URI
      builder.jwkSetUri("https://www.googleapis.com/oauth2/v3/certs");

      // Access Token 발행자 정보 조회 URI
      builder.issuerUri("https://accounts.google.com");

      // Resource Owner의 User Info를 조회하기 위한 URI
      builder.userInfoUri("https://www.googleapis.com/oauth2/v3/userinfo");

      // Resource Owner의 이름에 접근하기 위한 Attribute Name
      builder.userNameAttributeName(IdTokenClaimNames.SUB);

      // Client Name. Provider를 구분하는 용도로 사용
      builder.clientName("Google");
      return builder;
    }

  },

  ...
  ...

}
```

4️⃣ OAuth2AuthorizedClientService

- OAuth 2로 인증된 Client(`OAuth2AuthorizedClient`)를 관리하는 서비스 클래스
- OAuth2AuthorizedClientService 인터페이스의 구현 클래스
  - `InMemoryOAuth2AuthorizedClientService`
  - `JdbcOAuth2AuthorizedClientService`

5️⃣ ClientRegistrationRepository

- OAuth 2로 인증된 Client(`OAuth2AuthorizedClient`)를 저장하는 역할을 한다.
  - ClientRegistrationRepository 인터페이스의 구현 클래스
    - `InMemoryClientRegistrationRepository`
- OAuth2AuthorizedClientService가 ClientRegistrationRepository를 사용해 `OAuth2AuthorizedClient` 를 저장한다.

6️⃣ OAuth2AuthorizedClientManager

- `OAuth2AuthorizedClient` 를 관리하는 관리자 역할을 한다.

7️⃣ OAuth2AuthorizedClientProvider

- Authorization Grant 유형을 설정할 수 있는 인터페이스
  - Authorization Code
  - Implicit Grant Type
  - Client Credentials
  - Resource Owner Password Credentials

Spring Security 학습하느라 수고하셨습니다!

Spring WebFlux 유닛에서 다시 만나요!

항상 응원할게요!!!