Spring Boot Application

스프링 부트 응용

OAuth2-Client를 이용한 인증 처리

한국기술교육대학교
온라인평생교육원

학습내용

- OAuth 2.0 개념과 동작 원리
- OAuth-Client를 활용한 구글 인증 적용

학습목표

- OAuth 2.0 개념과 동작 원리를 설명할 수 있다.
- OAuth-Client를 활용하여 구글 인증을 적용할 수 있다.

OAuth 2.0 개념과 동작 원리

1 OAuth 개념

〈1〉 OAuth란?

- 대부분의 웹 애플리케이션에서 사용하는 아이디/비밀번호 기반의 인증은 여러 시스템에 개인 정보가 중복되어 개인 정보 관리가 어려움

1 OAuth 개념

〈1〉 OAuth란?

- 애플리케이션마다 관리하는 회원 정보가 오용되는 경우
  - 이를 알 수 없을 뿐더러 이를 제한할 명확한 방법이 없는 것이 현실임

OAuth 2.0 개념과 동작 원리

1 OAuth 개념

1 OAuth란?

합성어 OAuth

- 사용자 인증을 다른 시스템에서 대신 처리해주는 것을 의미함

1 OAuth 개념

1 OAuth란?

- OAuth를 이용하면 사용자 정보를 중앙집중해서 관리함

➢ 회원 가입에서부터 로그인, 회원 탈퇴,
휴면 계정 전환 등의 관리 부담에서 벗어날 수 있음

OAuth 2.0 개념과 동작 원리

1 OAuth 개념

① OAuth란?

OAuth 1.0	• IETF에서 2010년에 RFC 5849로 발표하면서 표준화됨

1 OAuth 개념

② OAuth 2.0

OAuth 2.0	• OAuth 1.0을 개선하여 2012년에 만들었음

OAuth 2.0 개념과 동작 원리

1 OAuth 개념

〈2〉 OAuth 2.0

기존 1.0에서 추가된 OAuth 2.0의 기능	
1	웹 뿐만이 아닌 모바일 앱(App)도 지원함
2	HTTPS 지원을 통해 보안을 강화함
3	인증 절차 및 구현이 단순해짐
4	Access Token의 유효시간을 설정하여 보안을 강화함

1 OAuth 개념

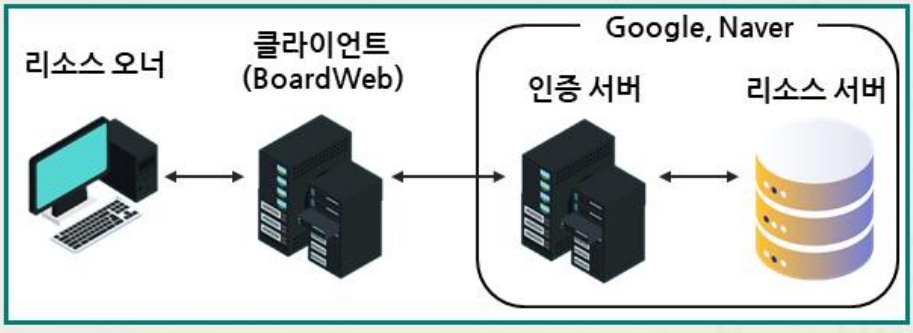
〈3〉 OAuth 2.0 용어 정리

- OAuth를 시스템에 적용하려면 OAuth에서 제공하는 용어들을 정확하게 이해해야 함

OAuth 2.0 개념과 동작 원리

1 OAuth 개념

〈3〉 OAuth 2.0 용어 정리



리소스 오너	리소스 서버에 저장된 사용자 정보의 소유자, 인증을 요청하는 주체
클라이언트	인증 서버 입장에서 인증을 요청하는 클라이언트
인증 서버	인증을 처리하는 서버
리소스 서버	사용자 정보가 저장된 자원 서버

2 인증서버와 OAuth 클라이언트

- OAuth 인증을 구현하기 위해서는 인증 서버에서 발급한 클라이언트 아이디와 리디렉트 URI 정보가 필요함

OAuth 2.0 개념과 동작 원리

2 인증서버와 OAuth 클라이언트

〈1〉 클라이언트 아이디

1 리소스 오너가 구글이나 네이버에 로그인 요청을 전송

2 OAuth 클라이언트(BoardWeb)는 곧바로 인증 서버에 인증을 요청함

2 인증서버와 OAuth 클라이언트

〈1〉 클라이언트 아이디

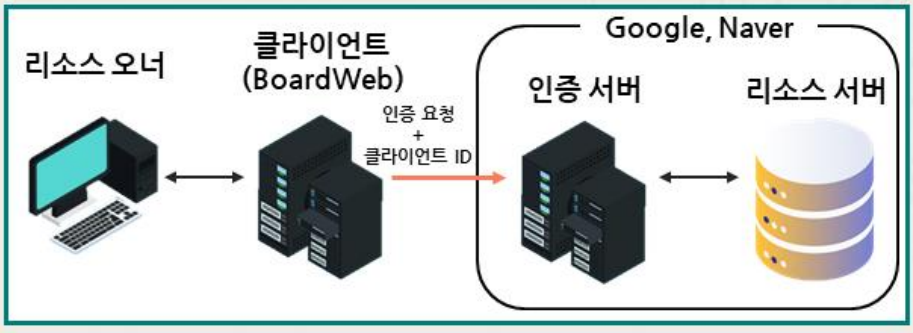
- 인증 서버는 OAuth 인증을 요청한 클라이언트를 식별할 수 없으면 인증을 처리할 수 없음

OAuth 2.0 개념과 동작 원리

2 인증서버와 OAuth 클라이언트

〈1〉 클라이언트 아이디

- 클라이언트 ID는 인증 서버가 OAuth 클라이언트를 식별할 수 있도록 OAuth 클라이언트가 인증 서버에게 전달해야 하는 필수 정보임



2 인증서버와 OAuth 클라이언트

〈2〉 리디렉트 URI

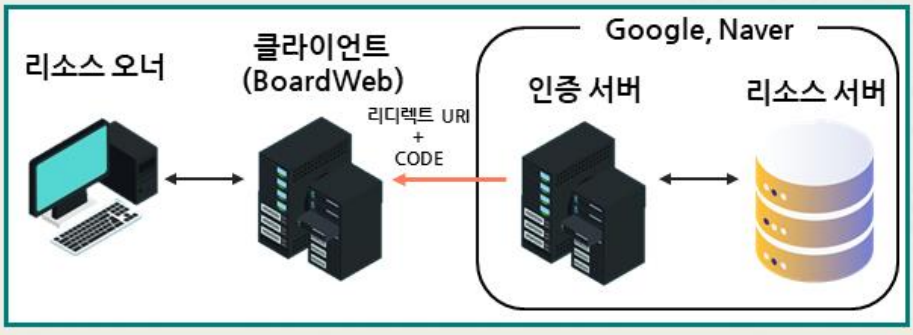
- 인증 서버는 인증에 성공한 OAuth 클라이언트에 대해 인증에 성공했다는 의미로 CODE 정보를 전달함

OAuth 2.0 개념과 동작 원리

2 인증서버와 OAuth 클라이언트

② 리디렉트 URI

- 인증 서버가 CODE 정보를 OAuth 클라이언트에게 전달(Callback)할 때 사용하는 URI



2 인증서버와 OAuth 클라이언트

③ 인증 서버에 클라이언트 등록

- 인증 서버로부터 클라이언트 ID와 리디렉트 URI를 발급받기 위해서는 인증 서버에 클라이언트 애플리케이션을 등록해야 함

OAuth 2.0 개념과 동작 원리

2 인증서버와 OAuth 클라이언트

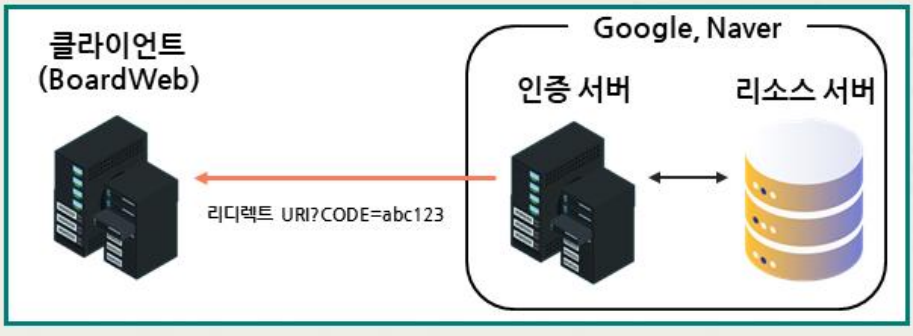
〈3〉 인증 서버에 클라이언트 등록

- Google API Console에 접속하여 구글로부터 클라이언트 ID와 리디렉트 URI를 발급받음

2 인증서버와 OAuth 클라이언트

〈4〉 인증 처리

- 사용자가 입력한 정보를 기반으로 인증에 성공한 인증 서버는 클라이언트에게 리디렉트 URI를 통해 CODE 정보를 되돌려 줌

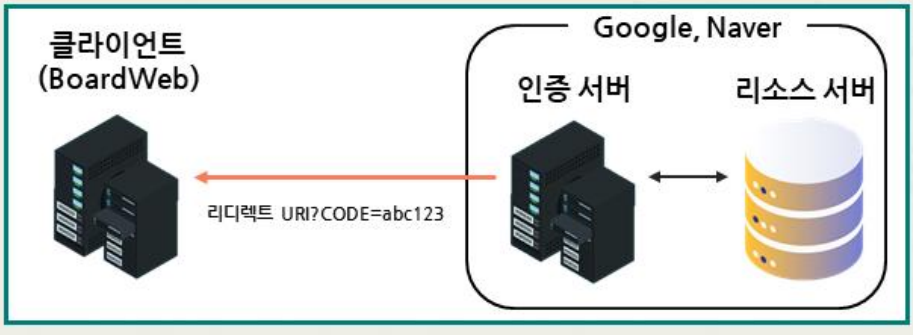


OAuth 2.0 개념과 동작 원리

2 인증서버와 OAuth 클라이언트

④ 인증 처리

- CODE를 받은 클라이언트는 인증을 요청한 리소스 오너가 구글에 정상적인 회원으로 등록되어 있음을 확인할 수 있음(여기 까지가 인증 성공)



2 인증서버와 OAuth 클라이언트

⑤ Access Token

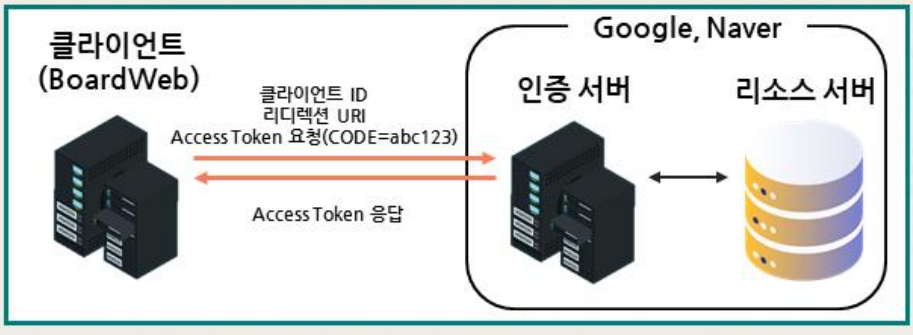
- 클라이언트가 리소스 서버에 저장된 자원에 접근할 수 있는 권한이 있음을 증명하는 증명서

OAuth 2.0 개념과 동작 원리

2 인증서버와 OAuth 클라이언트

5 Access Token

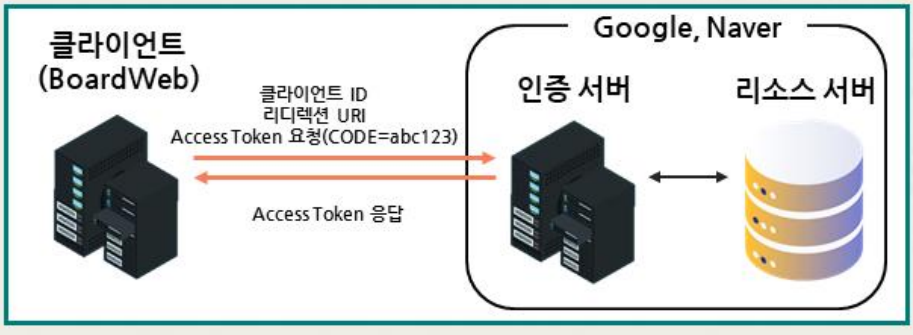
- 클라이언트는 인증 서버로부터 받은 CODE를 통해 Access Token을 요청할 수 있음



2 인증서버와 OAuth 클라이언트

5 Access Token

- Access Token을 넘겨받은 클라이언트는 인증 서버를 통해 리소스 서버에 저장된 리소스 오너의 정보를 요청할 수 있음



OAuth-Client를 활용한 구글 인증 적용

1 OAuth2-Client 설정

⟨1⟩ OAuth2-Client 라이브러리 추가

- 스프링 프로젝트에서 OAuth 인증을 처리하려면 가장 먼저 OAuth2-Client 스타터를 추가해야 함

1 OAuth2-Client 설정

⟨1⟩ OAuth2-Client 라이브러리 추가

- 그러면 pom.xml 파일에 다음과 같이 라이브러리 의존성이 추가로 설정됨

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

OAuth-Client를 활용한 구글 인증 적용

1 OAuth2-Client 설정

⟨1⟩ OAuth2-Client 라이브러리 추가

- 참고로 스프링은 현재 Google과 Facebook에 대한 OAuth2-Client 라이브러리를 지원함

1 OAuth2-Client 설정

⟨2⟩ application.yml 파일 수정

- application.yml 파일에 Google API Console에서 설정한 클라이언트 ID와 클라이언트 보안 비밀 코드를 등록함

```
# OAuth2 설정(Google)
security:
  oauth2:
    client:
      registration:
        google:
          client-id: 700754933096-a1b5kj2cugu8ur43t3kqp37vffs44t71.apps.googleusercontent.com
          client-secret: 00GCvdrOQ3337LQKkazQgqY6
          scope: email, profile
```

OAuth-Client를 활용한 구글 인증 적용

1 OAuth2-Client 설정

〈2〉 application.yml 파일 수정

scope

- 인증에 성공했을 때 인증 서버로부터 받을 사용자 정보

2 OAuth2-Client를 이용한 구글 인증

〈1〉 구글 로그인 링크 추가

- 로그인 화면(login.html)에 구글 로그인을 위한 메뉴를 추가함

```
<tr>
<td colspan="2" align="center">
<input type="submit" value="로그인">
<a href="/oauth2/authorization/google">구글 로그인</a></td>
</tr>
</table>
```

OAuth-Client를 활용한 구글 인증 적용

2 OAuth2-Client를 이용한 구글 인증

① 구글 로그인 링크 추가

- 로그인 링크의 URI가 '/oauth2/authorization/google'로 고정되어 있다는 것

```
<tr>
<td colspan="2" align="center">
<input type="submit" value="로그인">
<a href="/oauth2/authorization/google">구글 로그인</a></td>
</tr>
</table>
```

2 OAuth2-Client를 이용한 구글 인증

① 구글 로그인 링크 추가

- 만약 구글이 아닌 페이스북으로 로그인 링크를 설정하려면 로그인 링크 URI를 '/oauth2/authorization/facebook'으로 수정해야 함

```
<tr>
<td colspan="2" align="center">
<input type="submit" value="로그인">
<a href="/oauth2/authorization/google">구글 로그인</a></td>
</tr>
</table>
```

OAuth-Client를 활용한 구글 인증 적용

2 OAuth2-Client를 이용한 구글 인증

〈2〉 시큐리티 설정 클래스 수정

- 시큐리티 설정 클래스(SecurityConfig)의 configure 메소드에 OAuth2를 이용한 인증 설정을 추가함

```
@Override
protected void configure(HttpSecurity security) throws Exception {
    // 접근 제어
    security.authorizeRequests().antMatchers("/", "/board/login").permitAll();
    security.authorizeRequests().antMatchers("/board/**").authenticated();
    security.authorizeRequests().antMatchers("/manager/**").hasRole("MANAGER");

    // CSRF 인증 비활성화
    security.csrf().disable();

    // 로그인 화면 제공
    security.formLogin().loginPage("/board/login").defaultSuccessUrl("/board/getBoardList");
    security.oauth2Login().defaultSuccessUrl("/board/getBoardList");
}
```

3 구글 정보로 회원 가입

〈1〉 회원 가입 절차

- 인증 서버로부터 CODE를 받으면 로그인 인증이 성공했다는 의미

OAuth-Client를 활용한 구글 인증 적용

3 구글 정보로 회원 가입

① 회원 가입 절차

1. CODE 정보를 이용하여 Access Token을 받음
2. Access Token을 이용함
3. 인증 서버에 리소스 오너의 정보를 요청할 수 있음

3 구글 정보로 회원 가입

① 회원 가입 절차

- 이렇게 얻어진 리소스 오너의 정보를 통해 클라이언트에 없는 회원 정보를 등록하면 됨

OAuth-Client를 활용한 구글 인증 적용

3 구글 정보로 회원 가입

〈2〉 UserDetails 클래스 수정

- UserDetails 인터페이스를 구현한 PrincipalDetails 클래스에 OAuth2User 인터페이스를 추가함

```
public class PrincipalDetails implements UserDetails, OAuth2User {
    private static final long serialVersionUID = 1L;
    private Member member;
    // 구글에서 조회한 사용자 정보들을 담을 컬렉션
    private Map<String, Object> attributes;

    public PrincipalDetails(Member member) {
        this.member = member;
    }

    // OAuth 로그인시 사용할 생성자
    public PrincipalDetails(Member member, Map<String, Object> attributes) {
        this.member = member;
        this.attributes = attributes;
    }

    @Override
    public Map<String, Object> getAttributes() {
        return attributes;
    }

    @Override
    public String getName() {
        return member.getName();
    }
```

3 구글 정보로 회원 가입

```
public class PrincipalDetails implements UserDetails, OAuth2User {
    private static final long serialVersionUID = 1L;
    private Member member;
    // 구글에서 조회한 사용자 정보들을 담을 컬렉션
    private Map<String, Object> attributes;

    public PrincipalDetails(Member member) {
        this.member = member;
    }

    // OAuth 로그인시 사용할 생성자
    public PrincipalDetails(Member member, Map<String, Object> attributes) {
        this.member = member;
        this.attributes = attributes;
    }

    @Override
    public Map<String, Object> getAttributes() {
        return attributes;
    }

    @Override
    public String getName() {
        return member.getName();
    }
```

OAuth-Client를 활용한 구글 인증 적용

3 구글 정보로 회원 가입

③ 임시 비밀번호 등록

- 만약 인증 서버를 통해 구글 인증이 통과한 리소스 오너의 정보가 OAuth 클라이언트에 없는 경우

  - Access Token을 이용하여 얻은 사용자 정보를 토대로 회원 가입을 처리해야 함

3 구글 정보로 회원 가입

③ 임시 비밀번호 등록

- 구글 회원의 정보를 OAuth 클라이언트에 저장하기 위해서 application.yml 파일에 구글 회원 가입 시 사용할 임시 비밀번호를 등록함

```
# 구글 회원가입 시 사용할 임시 비밀번호
google.default.password: google123
```

OAuth-Client를 활용한 구글 인증 적용

3 구글 정보로 회원 가입

④ DefaultOAuth2UserService 구현 클래스

- DefaultOAuth2UserService 인터페이스를 구현한 PrincipalOauth2DetailsService 클래스를 작성함

```
package com.mycompany.security.config;

import org.springframework.beans.factory.annotation.Autowired;

@Service
public class PrincipalOauth2DetailsService extends DefaultOAuth2UserService {
    @Autowired
    private PasswordEncoder passwordEncoder;

    @Autowired
    private MemberRepository memberRepository;

    @Value("${google.default.password}")
    private String googlePassword;

    @Override
    public OAuth2User loadUser(OAuth2UserRequest userRequest) throws OAuth2AuthenticationException {
        return new PrincipalDetails(null, null);
    }
}
```

3 구글 정보로 회원 가입

④ DefaultOAuth2UserService 구현 클래스

- DefaultOAuth2UserService 클래스로부터 재정의한 loadUser 메소드에서는 Access Token이 저장된 OAuthUserRequest 객체를 통해 인증 서버에게 사용자 정보를 요청할 수 있음

OAuth-Client를 활용한 구글 인증 적용

3 구글 정보로 회원 가입

④ DefaultOAuth2UserService 구현 클래스

- OAuth2User 객체를 리턴하는 loadUser 메소드 구현

```
@Override
public OAuth2User loadUser(OAuth2UserRequest userRequest) throws OAuth2AuthenticationException {
    // 구글로부터 받은 프로필을 이용하여 회원가입 정보를 설정한다.
    OAuth2User oauth2User = super.loadUser(userRequest);
    String providerId = oauth2User.getAttribute("sub");
    String email = oauth2User.getAttribute("email");
    String username = email + "_" + providerId;
    String password = passwordEncoder.encode(googlePassword);

    // 회원가입이 되어있는 사용자인지 확인한다.
    Member member = memberRepository.findById(username).orElseGet(()->{
        return new Member();
    });

    // 회원가입이 되어있지 않은 사용자라면 회원가입을 처리한다.
    if(member.getUsername() == null) {
        // 신규 가입 처리
        member = new Member();
        member.setUsername(username);
        member.setPassword(password);
        member.setName(email);
        member.setRole(Role.ROLE_USER);
        member.setEnabled(true);
        memberRepository.save(member);
    }

    // OAuth-Client는 리턴된 PrincipalDetails로 세션에 저장된 사용자 정보를 갱신한다.
    return new PrincipalDetails(member, oauth2User.getAttributes());
}
```

3 구글 정보로 회원 가입

④ DefaultOAuth2UserService 구현 클래스

```
@Override
public OAuth2User loadUser(OAuth2UserRequest userRequest) throws OAuth2AuthenticationException {
    // 구글로부터 받은 프로필을 이용하여 회원가입 정보를 설정한다.
    OAuth2User oauth2User = super.loadUser(userRequest);
    String providerId = oauth2User.getAttribute("sub");
    String email = oauth2User.getAttribute("email");
    String username = email + "_" + providerId;
    String password = passwordEncoder.encode(googlePassword);

    // 회원가입이 되어있는 사용자인지 확인한다.
    Member member = memberRepository.findById(username).orElseGet(()->{
        return new Member();
    });

    // 회원가입이 되어있지 않은 사용자라면 회원가입을 처리한다.
    if(member.getUsername() == null) {
        // 신규 가입 처리
        member = new Member();
        member.setUsername(username);
        member.setPassword(password);
        member.setName(email);
        member.setRole(Role.ROLE_USER);
        member.setEnabled(true);
        memberRepository.save(member);
    }

    // OAuth-Client는 리턴된 PrincipalDetails로 세션에 저장된 사용자 정보를 갱신한다.
    return new PrincipalDetails(member, oauth2User.getAttributes());
}
```

OAuth-Client를 활용한 구글 인증 적용

3 구글 정보로 회원 가입

⑤ DefaultOAuth2UserService 적용

- 스프링 시큐리티 설정 클래스(SecurityConfig)에 DefaultOAuth2UserService 객체를 등록하여 사용자 인증을 처리함

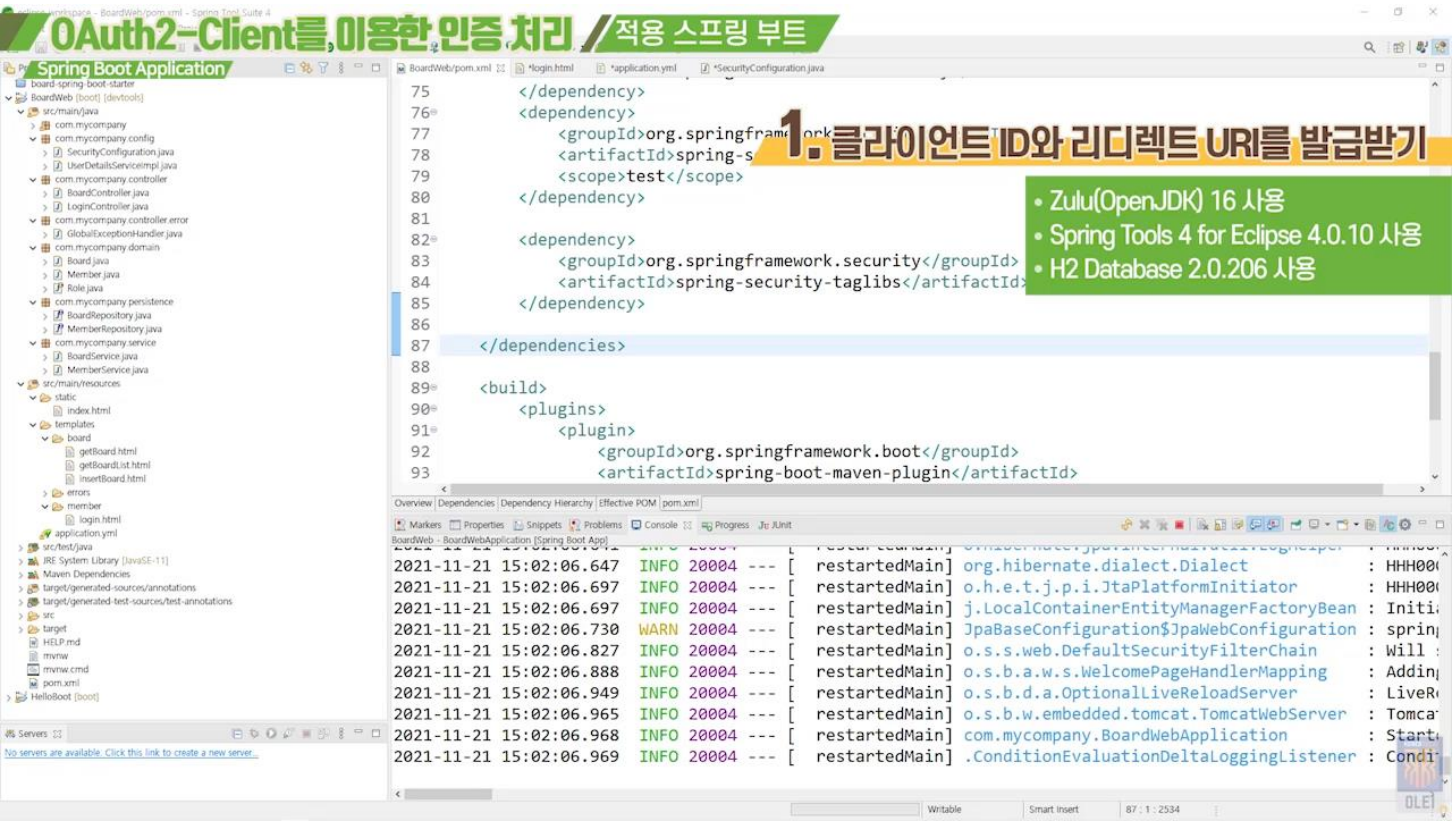
```
@Autowired
private PrincipalOauth2DetailsService oauth2DetailsService;

@Override
protected void configure(HttpSecurity security) throws Exception {
    // 접근 제어
    security.authorizeRequests().antMatchers("/", "/board/login").permitAll();
    security.authorizeRequests().antMatchers("/board/**").authenticated();
    security.authorizeRequests().antMatchers("/manager/**").hasRole("MANAGER");

    // CSRF 인증 비활성화
    security.csrf().disable();

    // 로그인 화면 제공
    security.formLogin().loginPage("/board/login").defaultSuccessUrl("/board/getBoardList");
    security.oauth2Login().defaultSuccessUrl("/board/getBoardList")
    .userInfoEndpoint()
    .userService(oauth2DetailsService);
}
```

적용 스프링 부트

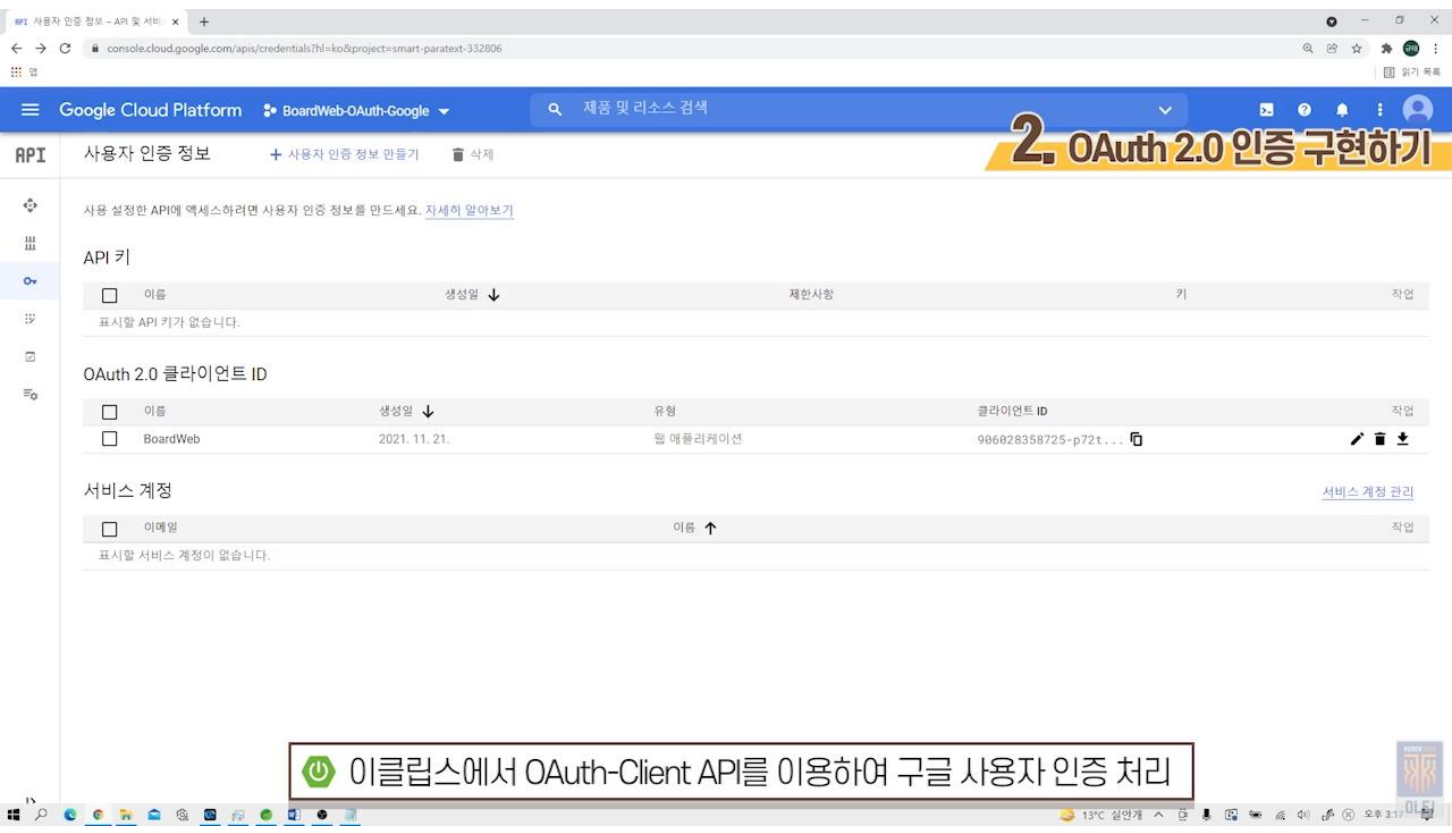


실습단계
Google OAuth API Console에 접속하여 클라이언트 등록, 클라이언트 ID와 보안 비밀번호를 발급받음
구글 계정 필요
Google API Console 검색
프로젝트 생성
새 프로젝트 → 프로젝트 이름 작성
BoardWeb-OAuth-Google 지정, 다른 이름으로 해도 상관없음
프로젝트 변경
Oauth 동의 화면 → 외부 → 만들기
계정 생성 시 등록한 이메일 지정
사용자 인증 정보 → 사용자 인증 정보 만들기 → OAuth 클라이언트 ID
BoardWeb이라는 애플리케이션을 웹으로 만들었기 때문에 유형은 웹 애플리케이션 선택!
코드 정보를 callback 해서 받을 리디렉션 URI 등록
login/oauth2/code 부분은 고정되어 있음

적용 스프링 부트

실습단계
OAuth2에서는 구글과 페이스북 기반의 OAuth 인증 지원
클라이언트 ID와 클라이언트 보안 비밀번호 확인 방법
Tip! 활용을 위해 필요한 정보 메모 필요

적용 스프링 부트



실습단계
이클립스에서 OAuth-Client API를 이용하여 구글 사용자 인증 처리
라이브러리 추가
<Ctrl> + <Space> → Add Starters
실행 중인 애플리케이션 종료 후 다시 실행
로그인 화면 수정
구글 로그인 하이퍼링크 추가
중요! application.yml 파일을 열어서 OAuth2 client 설정 추가
들여쓰기 주의
앞서 메모로 저장한 정보 등록
인증 성공 시 사용자 정보를 얻을 수 있음
에러 알림의 경우 실제 실행 시 에러가 아닐 수 있음! 확인 필요
SecurityConfiguration.java 열어서 OAuth2 클라이언트를 이용한 Google 인증 처리
모든 설정이 마무리되면 애플리케이션 종료 후 다시 구동
브라우저에서 실행
구글 인증 시도 가능