OAuth란?

OAuth(Open Authorization)는 토큰 기반의 인증 및 권한을 위한 표준 프로토콜이다. OAuth와 같은 인증 프로토콜을 통해 유저의 정보를 페이스북, 구글, 카카오 등의 서비스에서 제공받을 수 있고 이 정보를 기반으로 어플리케이션 사용자에게 로그인이나 다른 여러 기능들을 손쉽게 제공할 수 있다. 자세한 내용은 여기를 참조한다.

스프링 부트로 OAuth2를 통한 로그인 기능

스프링 부트로 OAuth2를 통한 로그인 기능을 제공할 수 있다.
여기서는 페이스북, 구글, 카카오, 네이버에서 제공하는 정보를 통해 사용자가 웹페이지에 손쉽게 로그인하는 기능을 구현할 것이다. 스프링부트로 해당 기능을 구현하기 전에 먼저 위 4가지 서비스의 개발자 사이트에 들어가서 OAuth 인증을 위한 설정을 해야한다.

구글 디벨로퍼 콘솔

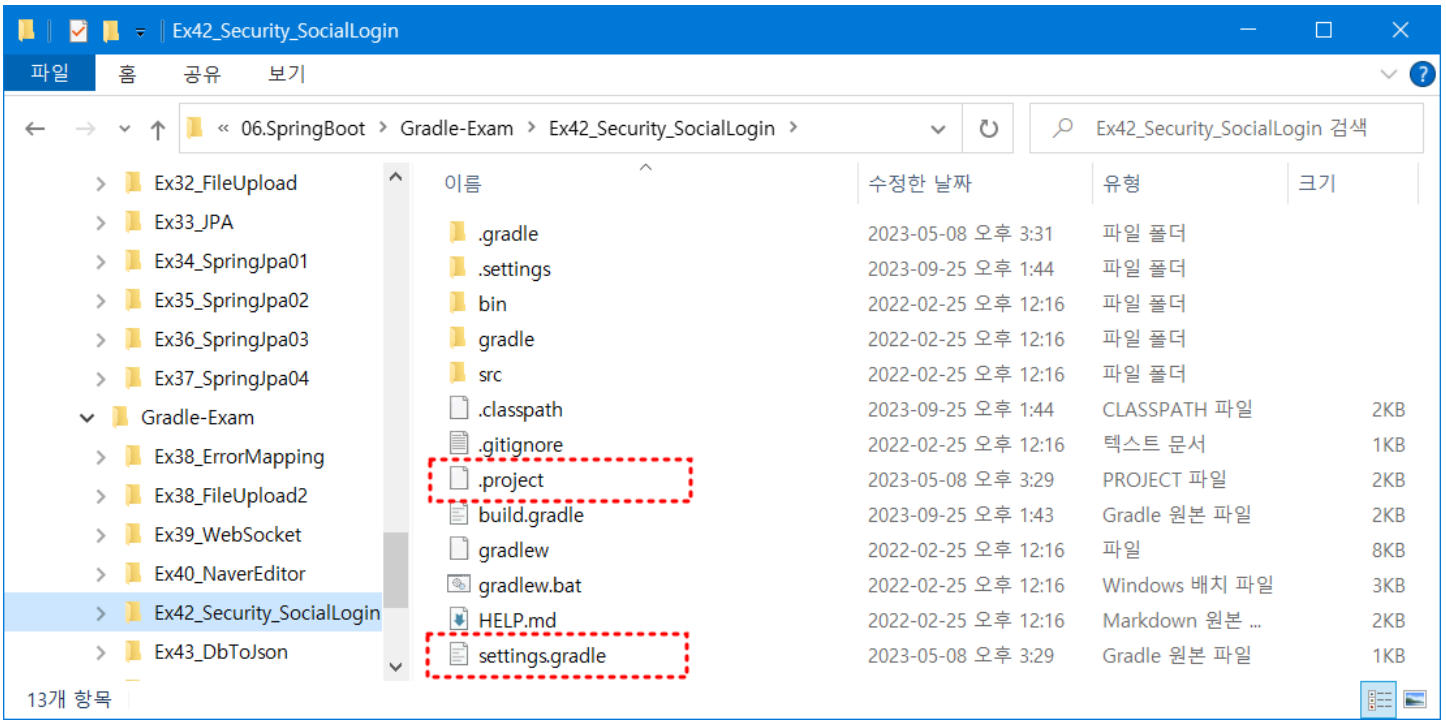
카카오 개발자 페이지

네이버 개발자 센터

프로젝트 구성

Ex29_Security_tablibs 프로젝트를 복사하여 이름을 Ex42_Security_SocialLogin 으로 수정한다.

폴더 이름 변경



settings.gradle 파일안의 내용 변경

settings.gradle

```
rootProject.name = 'Ex42_Security_SocialLogin'
```

.project 파일안의 내용 변경

```
<?xml version="1.0" encoding="UTF-8"?>
<projectDescription>
    <name>Ex42_Security_SocialLogin</name>
    <comment>Project Ex42_Security_SocialLogin created by Buildship.</comment>
    <projects>
    </projects>
```

.settings 폴더의 org.eclipse.wst.common.component 파일을 열어서 다음처럼 내용 수정

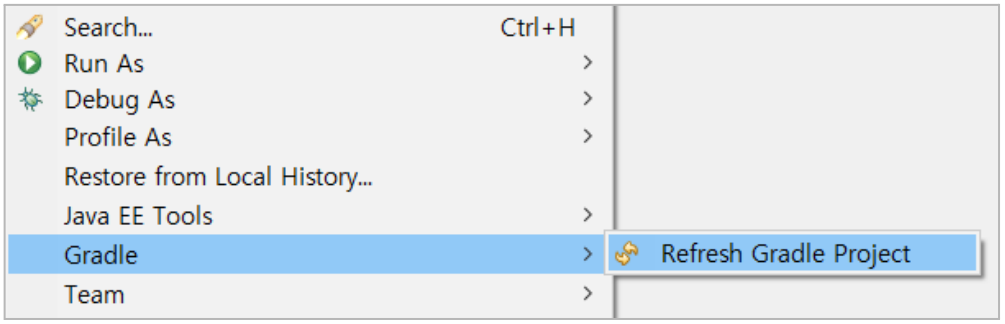
```
<?xml version="1.0" encoding="UTF-8"?>
<project-modules id="moduleCoreId" project-version="1.5.0">
    <wb-module deploy-name="Ex42_Security_SocialLogin">
        <property name="context-root" value="Ex42_Security_SocialLogin"/>
        <wb-resource deploy-path="/WEB-INF/classes" source-path="src/main/resources"/>
        <wb-resource deploy-path="/WEB-INF/classes" source-path="src/main/java"/>
        <wb-resource deploy-path="/" source-path="src/main/webapp"/>
    </wb-module>
</project-modules>
```

build.gradle 수정

- 다음 내용과 비교하여 빠진 부분이 있다면 입력하여 추가하도록 한다.

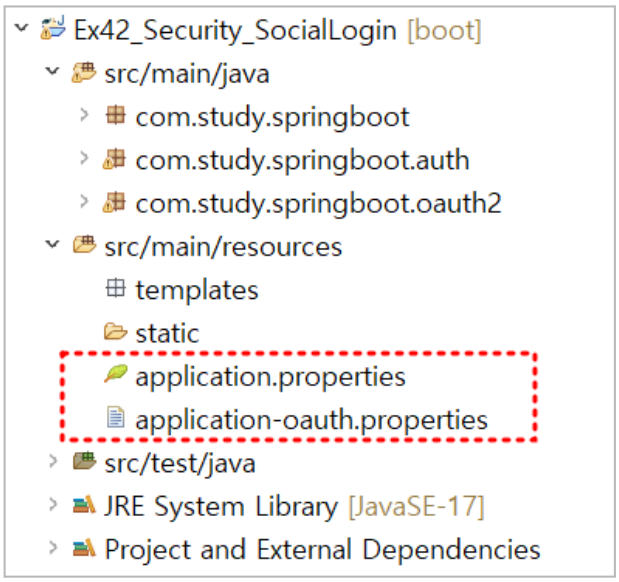
```
27 dependencies {
28     implementation 'org.springframework.boot:spring-boot-starter-security'
29     implementation 'org.springframework.boot:spring-boot-starter-web'
30     compileOnly 'org.projectlombok:lombok'
31     annotationProcessor 'org.projectlombok:lombok'
32     providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
33     testImplementation 'org.springframework.boot:spring-boot-starter-test'
34     testImplementation 'org.springframework.security:spring-security-test'
35     implementation 'jakarta.servlet:jakarta.servlet-api'
36     implementation 'jakarta.servlet.jsp.jstl:jakarta.servlet.jsp.jstl-api'
37     implementation 'org.glassfish.web:jakarta.servlet.jsp.jstl'
38     implementation 'org.apache.tomcat.embed:tomcat-embed-jasper'
39     implementation 'org.springframework.security:spring-security-taglibs'
40     implementation 'org.springframework.boot:spring-boot-starter-oauth2-client'
41 }
```

다음과 같이 메뉴를 선택하여 수정한 내용이 적용되도록 한다.



속성 파일 추가

- application.properties 를 복사하여 붙여넣고 이름을 application-oauth.properties 로 수정한다.



application.properties 내용을 다음과 같이 수정한다.

- application-oauth.properties 파일을 속성 파일로 추가한다고 지정

```
1 server.port=8081
2 # JSP 환경 설정
3 spring.mvc.view.prefix=/WEB-INF/views/
4 spring.mvc.view.suffix=.jsp
5
6 spring.profiles.include=oauth
7
```

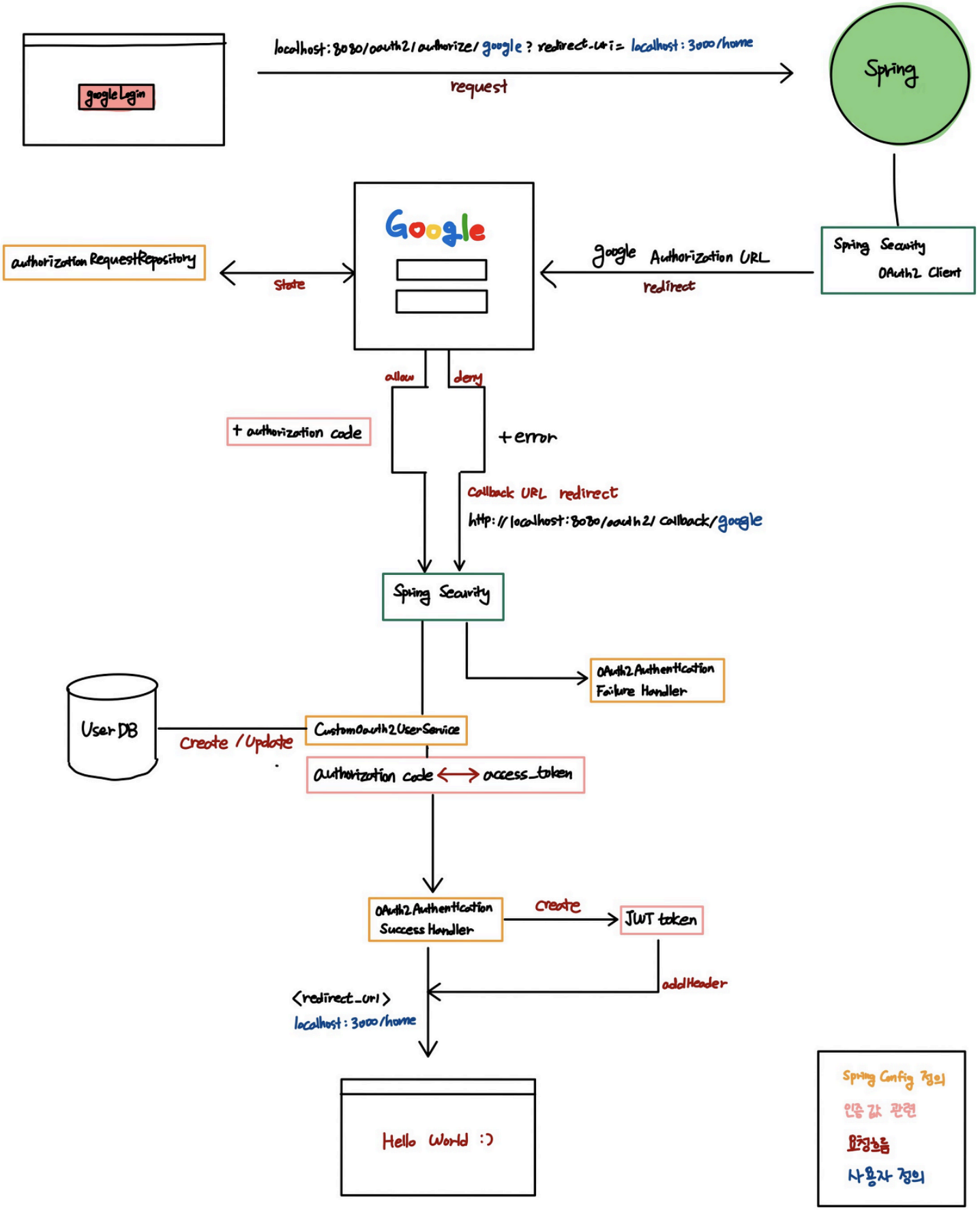
application-oauth.properties 내용 작성

다음의 내용을 복사하여 붙여 넣기 한다.

- 각 계정별 client-id, clietnt-secret 는 본인 계정의 것으로 입력한다.
- Google, Facebook 은 스프링에 기능이 내장되어 있기 때문에 이정도의 정보만 주면 된다.
- Facebook의 경우 provide로 추가 정보를 지정해야 사진을 가져올 수 있다.
- Kakao, Naver 는 기본으로 지원이 되지 않기에 모든 항목에 대한 정보를 제공해야 한다.
    - 관련 값들은 개발자 센터에서 모든 정보가 제공되고 있다.
    - 이런 식으로 어떤 사이트라도 OAuth 로그인을 구현할 수 있다.

```
# Google
spring.security.oauth2.client.registration.google.client-id=본인클라이언트아이디
spring.security.oauth2.client.registration.google.client-secret=본인시크릿키
spring.security.oauth2.client.registration.google.scope=profile,email

# Facebook
spring.security.oauth2.client.registration.facebook.client-id=본인클라이언트아이디
spring.security.oauth2.client.registration.facebook.client-secret=본인시크릿키
spring.security.oauth2.client.provider.facebook.user-info-uri=https://graph.facebook.com/me?fields=id,name,
email,picture

# Kakao
spring.security.oauth2.client.registration.kakao.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.kakao.client-id=본인REST API 키
spring.security.oauth2.client.registration.kakao.redirect-uri={baseUrl}/login/oauth2/code/{registrationId}
spring.security.oauth2.client.registration.kakao.scope=profile,account_email
spring.security.oauth2.client.registration.kakao.client-authentication-method=POST
spring.security.oauth2.client.registration.kakao.client-name=Kakao

spring.security.oauth2.client.provider.kakao.authorization-uri=https://kauth.kakao.com/oauth/authorize
spring.security.oauth2.client.provider.kakao.token-uri=https://kauth.kakao.com/oauth/token
spring.security.oauth2.client.provider.kakao.user-info-uri=https://kapi.kakao.com/v2/user/me
spring.security.oauth2.client.provider.kakao.user-name-attribute=kakao_account

# Naver
spring.security.oauth2.client.registration.naver.client-id=본인클라이언트아이디
spring.security.oauth2.client.registration.naver.client-secret=본인시크릿키
spring.security.oauth2.client.registration.naver.redirect-uri={baseUrl}/login/oauth2/code/{registrationId}
spring.security.oauth2.client.registration.naver.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.naver.scope=name,email,profile_image
spring.security.oauth2.client.registration.naver.client-name=Naver

spring.security.oauth2.client.provider.naver.authorization-uri=https://nid.naver.com/oauth2.0/authorize
spring.security.oauth2.client.provider.naver.token-uri=https://nid.naver.com/oauth2.0/token
spring.security.oauth2.client.provider.naver.user-info-uri=https://openapi.naver.com/v1/nid/me
spring.security.oauth2.client.provider.naver.user-name-attribute=response
```

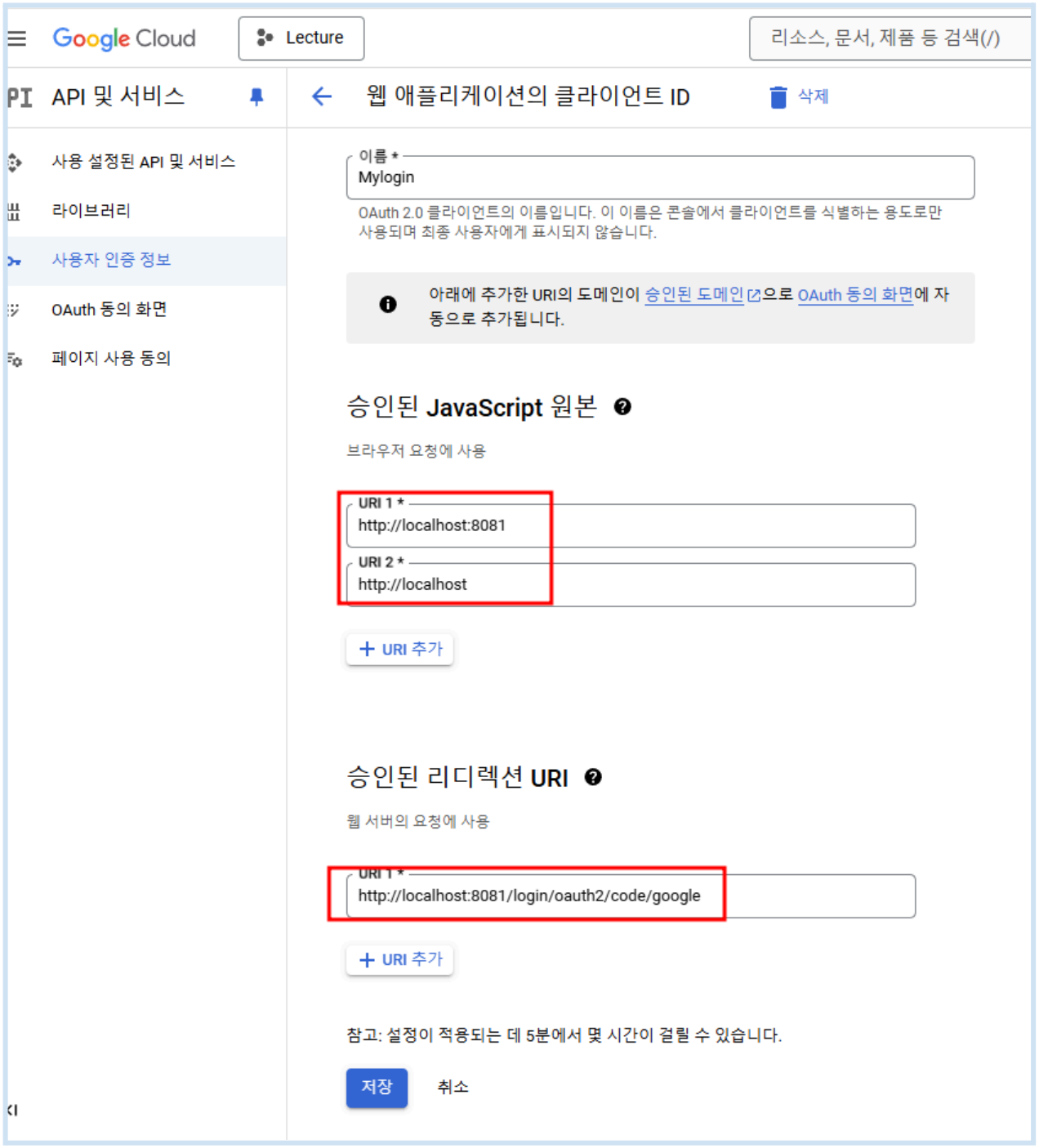
앞에서 작성한 정보들은 다음 플로우에서 사용된다. (참고)



이제 OAuth 로그인을 지원하기 위해 다음과 같이 패키지를 추가하고 클래스들을 추가한다.

- 한 번 만들면 그대로 사용하는 것들이기 때문에 수업용 자료실의 자료를 이용한다.
- 이 부분은 Social Login 이 사이트가 추가될 때 수정이 필요하다. 그 외에는 수정없이 사용한다.

구글 설정에서는 다음을 추가한다.



카카오 설정에서는 다음을 확인한다.

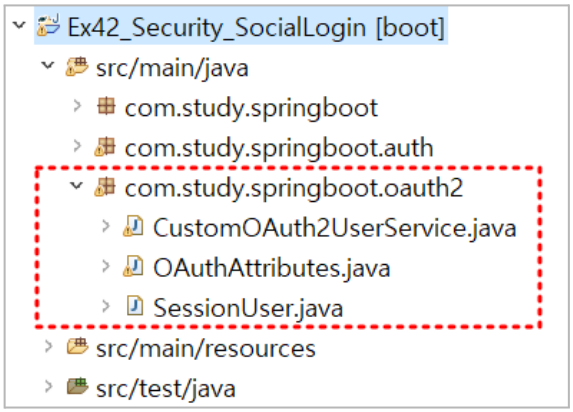

```
# Kakao
spring.security.oauth2.client.registration.kakao.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.kakao.client-id=10d9ec301e33b6cf5732db01e8554a86
spring.security.oauth2.client.registration.kakao.redirect-uri={baseUrl}/login/oauth2/code/{regis
spring.security.oauth2.client.registration.kakao.scope=profile_image,account_email
spring.security.oauth2.client.registration.kakao.client-authentication-method=POST
spring.security.oauth2.client.registration.kakao.client-name=Kakao
```

Naver 세팅에서 다음을 확인한다.

중간 부분에 [프로필 사진] 체크 확인 ...

하단 부분에 다음을 확인


PC 웹
서비스 URL
http://localhost:8081
서비스 URL예시: (O) http://naver.com (X) http://www.naver.com
서비스 URL값이 잘못 입력되어 있으면 정확한 값으로 수정하실 때 까지 네이버 로그인 사용이 일시적으로 제한됩니다.
불법/음란성 사이트 등 이용약관에 위배되는 사이트의 경우, 이용이 제한될 수 있습니다.
서비스하려는 사이트 URL과 동일한 사이트 URL로 해주셔야 네이버 로그인 뱃지가 노출됩니다.
로그인 오픈 API
서비스 환경
네이버 로그인
Callback URL (최대 5개)
http://localhost:8081/login/oauth2/code/naver
http://localhost:8081/MustHaveJSP/NaverLogin/login.jsp
텍스트 폼 우측 끝의 '+' 버튼을 누르면 행이 추가되며, '−' 버튼을 누르면 행이 삭제됩니다.
Callback URL은 네이버 로그인 후 이동할 페이지 URL입니다. Callback URL값이 잘못 입력되어 있으면 정확한 값으로 수정하실 때 까지 네이버 로그인 사용이 일시적으로 제한됩니다.
입력한 주소와 다른 Callback URL로 리다이렉트 될 경우, 이용이 제한될 수 있습니다.



SessionUser.java	세션 정보를 객체로 저장하기 위한 bean
OAuthAttributes	각 SocialLogin 의 응답 데이터를 파싱하여 공통 요소로 만들어 반환
CustomOAuth2UserService	실제 SocialLogin 을 수행하는 클래스 로그인에 성공하면 OAuthAttributes로 부터 받은 값으로 우리 프로그램상에서 사용할 데이터를 세션에 추가한다.

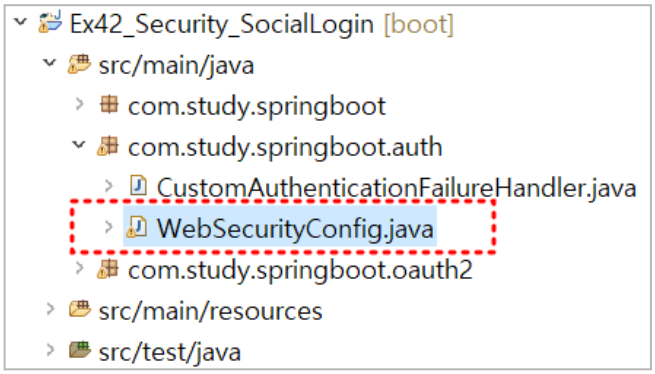
OAuthAttributes 에서 각 Social Login 성공으로 온 데이터 파싱하여 정리

```
    public static OAuthAttributes of(String registrationId, String userNameAttributeName,
                                     Map<String, Object> attributes)
    {
//      System.out.println(registrationId);
//      System.out.println(userNameAttributeName);
        if (registrationId.equals("google")) {
            return ofGoogle(userNameAttributeName, attributes);
        } else if  (registrationId.equals("facebook")) {
            return ofFacebook(userNameAttributeName, attributes);
        } else if  (registrationId.equals("kakao")) {
            return ofKakao(userNameAttributeName, attributes);
        } else if  (registrationId.equals("naver")) {
            return ofNaver(userNameAttributeName, attributes);
        }
        return ofGoogle(userNameAttributeName, attributes);
    }
```

CustomOAuth2UserService 에서 세션에 필요한 데이터 세팅

```
        OAuthAttributes attributes = OAuthAttributes.of(registrationId,
                                                        userNameAttributeName,
                                                        oAuth2User.getAttributes());

        SessionUser user = new SessionUser(attributes.getName(),
                                           attributes.getEmail(),
                                           attributes.getPicture());
//      System.out.println("Picture:"+attributes.getPicture());
        httpSession.setAttribute("user", user);
```

기존 WebSecurityConfig 에 OAuth 로그인을 사용하기 위한 설정 추가




```
21 @Configuration
22 public class WebSecurityConfig{
23 
24     @Autowired
25     public AuthenticationFailureHandler authenticationFailureHandler;
26     @Autowired
27     private CustomOAuth2UserService customOAuth2UserService;
28 
29     @Bean
30     public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
...
52         http.logout((logout) -> logout
53             .logoutUrl("/logout") // default
54             .logoutSuccessUrl("/")
55             .deleteCookies("JSESSIONID")
56             .invalidateHttpSession(true)
57             .permitAll());
58 
59         // header 충돌 방지
60         http.headers((headers) -> headers
61                 .frameOptions(frameOptions -> frameOptions.disable())
62             );
63         http.oauth2Login((oauth) -> oauth
64                 .userInfoEndpoint(endPoint -> endPoint
65                     .userService(customOAuth2UserService)
66                 )
67             );
68 
69         return http.build();
70     }
```

loginForm.jsp 수정

Social Login 을 위한 링크를 다음과 같이 추가한다.

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>LoginForm</title>
</head>
<body>
<h1>loginForm.jsp</h1>

<c:url value="j_spring_security_check" var="loginUrl"/>
<form action="<c:url value="j_spring_security_check" />" method="post">
    <c:if test="${param.error != null}">
    <p>
        Login Error! <br />
        ${error_message}
    </p>
    </c:if>
    ID : <input type="text" name="j_username"> <br />
    PW : <input type="text" name="j_password"> <br />
    <input type="submit" value="LOGIN"> <br />
</form>

<a href="/oauth2/authorization/google">Google Login</a>   
<a href="/oauth2/authorization/facebook">Facebook Login</a>   
<a href="/oauth2/authorization/kakao">Kakao Login</a>   
<a href="/oauth2/authorization/naver">Naver Login</a>

</body>
</html>
```

application-oauth.properties 에서
spring.security.oauth2.client.registration.google.xxx 같이 등록한 내용에서 google 이 부분을
<a href="/oauth2/authorization/google> 로 지정하면 된다.

welcome2.jsp 수정

페이스북의 경우는 세션 악용을 보호하기 위해 redirection url 에 몇 가지 데이터를 추가하는데 다음과 같이 자바스크립트를 이용하여 제거할 수 있다.

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="jakarta.tags.core" prefix="c" %>
<%@ taglib uri="http://www.springframework.org/security/tags" prefix="sec" %>
<%@ page import="com.study.springboot.oauth2.SessionUser"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Welcome</title>
</head>
<body>
welcome : Member

<hr>

<sec:authorize access="isAuthenticated()">
<p> Log-In</p>
</sec:authorize>

    ID : <sec:authentication property="name"/> <br>
    소유 권한 : <sec:authentication property="authorities"/> <br>

<%
    request.setCharacterEncoding("UTF-8");

    SessionUser obj = (SessionUser) session.getAttribute("user");
    String sName = (String)obj.getName();
    String sEmail = (String)obj.getEmail();
    String sPicture = (String)obj.getPicture();
%>

<%= sName %> <br>
<%= sEmail %> <br>
<img src=<%= sPicture %>> <br>

<a href="/logout">Log Out</a> <br />

<script>
// '#_=_'은 보안 세션 악용을 보호하기 위해 Facebook에 의해 추가되었다.
// Facebook에 따르면 이를 처리하는 것은 개발자의 책임이다.
console.log("aaaa" + window.location.hash);
if (window.location.hash == '#_=_'){
    console.log("bbbb");

    history.replaceState
        ? history.replaceState(null, null, window.location.href.split('#')[0])
        : window.location.hash = '';
}
</script>

</body>
</html>
```