3/11 개발일지

1. 소셜 로그인 개발작업 진행중

▼ 인증코드를 토큰화시켜 만드는 과정

받아온 인증코드를 토큰화(각 서비스에서) 하여 JWT 토큰을 받는것이 목적이다.

이를 위하여 아래와 같이 설정함.

- OAuth2TokenService.java
  - 인증코드를 받아와 각 서비스에 해당하는 엔드포인트로 요청 전송

```
package web.web1.Oauth.domain;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

import org.springframework.beans.factory.annotation.A
import org.springframework.context.annotation.Proper
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.util.UriComponentsBui

import org.springframework.beans.factory.annotation.A
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.util.UriComponentsBui

import com.google.api.client.util.Value;

import web.web1.Oauth.token.*;
```

```
// import java.math.BigInteger;
// import java.security.SecureRandom;

@Service
// @PropertySource("classpath:application.properties
public class OAuth2TokenService {

    private OAuth2Token tokenProvider;

    // @Value("${spring.security.oauth2.client.regis
    // private String clientID;

    // @Value("${spring.security.oauth2.client.regis
    // private String clientSecret;

    public String oauth2MakeServiceProviderGetReqest

        System.out.println("serviceType: " + service

        if(serviceType.equals("naver")) {
            this.tokenProvider = new NaverToken();
        } else if(serviceType.equals("google")) {
            this.tokenProvider = new GoogleToken();
        } else {
            throw new IllegalArgumentException("지원하
        }


        String clientID = this.tokenProvider.getClie
        String clientSecret = this.tokenProvider.get
        String code = this.tokenProvider.getCode();
        String redirectUri = this.tokenProvider.getR
        String grantType = this.tokenProvider.getGra


        String url = "";

        if(this.tokenProvider instanceof NaverToken)
```

```
            url = "https://nid.naver.com/oauth2.0/au
            + clientID + "&client_secret=" + clientS
            + redirectUri + "&grant_type=" + grantTy
        } else if(this.tokenProvider instanceof Goog
            url = "https://oauth2.googleapis.com/tok
            + code + "&client_id=" + clientID + "&cl
            + clientSecret + "&grant_type=" + "autho
        }
        System.out.println(url);
        return url;

    }

    @Autowired
    private RestTemplate restTemplate;

    public String exchangeCodeForAccessToken(String
        // Token 엔드포인트 URL

        System.out.println("---exchangeCodeForAccess
        String tokenEndpoint = "https://oauth2.googl

        // 요청 본문을 구성하기 위한 UriComponentsBuilder
        String requestBody = UriComponentsBuilder.ne
                .queryParam("code", decode)
                .queryParam("client_id", clientId)
                .queryParam("client_secret", clientS
                .queryParam("redirect_uri", redirect
                .queryParam("grant_type", "authoriza
                .toUriString().substring(1); // 맨 앞


        System.out.println("requestBody: " + request

        // HTTP 헤더 설정
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_
```

```
        // HttpEntity 객체 생성
        HttpEntity<String> requestEntity = new HttpE
        System.out.println(requestEntity);

        // POST 요청 보내기
        ResponseEntity<String> responseEntity = rest

        // 응답 본문 반환
        System.out.println(responseEntity.getBody())
        return responseEntity.getBody();
    }
    // 1. 네이버냐 구글이냐에 따라 네이버면 ../token/NaverT
    // 2. 해당 서비스에 대해 필요한 조건을 찾기. naver면
        // 2-1. client_id(),
        // 2-2.response_type(이건인증코드)
        // 2-3. redirect_uri(),
        // 2-4. state(NaverToken에 구현방식잇음),
    // 3. 해당 내용을 json화 시켜서
    // https://nid.naver.com/oauth2.0/token?client_i
    //로 보냄
}
```

- 이 클래스를 실행할 핸들러(SecurityConfig 진행사항에 붙을)를 만들었다.
    - 구글에 필요한 파라미터 (code, clientId, clientSecret, redirect_url, grant_type)을 정의함

```
package web.web1.Oauth.config;

import org.springframework.security.core.Authenticat
import org.springframework.security.web.authenticati
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;

import web.web1.Oauth.domain.OAuth2TokenService;
```

```
import web.web1.Oauth.domain.PrincipalDetails;

public class CustomAuthenticationSuccessHandler exte

    private final OAuth2TokenService oAuth2TokenServ

    public CustomAuthenticationSuccessHandler(OAuth2
        this.oAuth2TokenService = oAuth2TokenService
    }

    @Override
    public void onAuthenticationSuccess(HttpServletR
        if (authentication.getPrincipal() instanceof
            PrincipalDetails principalDetails = (Pri
            // Extract the service type (provider) f
            String serviceType = principalDetails.ge

            // Retrieve the authorization code from
            String code = request.getParameter("code
            String decode = URLDecoder.decode(code, S

            if (decode != null && !decode.isEmpty())
                // Assuming the clientId, clientSecr
                String clientId = "162422478014-k780
                String clientSecret = "GOCSPX-lMnXy7
                String redirectUri = "http://localho

                // Directly exchange the authorizati
                String accessToken = oAuth2TokenServ

                // Log or process the accessToken as
                System.out.println("Access Token: " +

                // Optionally, redirect the user to a
                getRedirectStrategy().sendRedirect(r
            } else {
                // Fallback to the default behavior i
                super.onAuthenticationSuccess(reques
```

```
            }
        } else {
            super.onAuthenticationSuccess(request, r
        }
    }
}
```