**2022년도 졸업프로젝트**

주제: TLS를 구현을 통한 홈페이지 제작 및 보안 채널 로그인 기능 구현

|  |  |  |  |
| --- | --- | --- | --- |
| **작성자** | 김도현 | **작성일자** | 2022.9.07 |

* **금일 진행 상황**

|  |  |
| --- | --- |
| **진행 상황** | **비고** |
| **질문 및 진행방식에 대한 논의**  **진행 중**  **완료 1. Oauth2 기능 추가**  **web.ignoring().mvcMatchers("/login/\*\*"); // /login/\*\* 있는 모든 파일들은 시큐리티 적용을 무시한다.**  **web.ignoring().mvcMatchers("/users/oauthLogin/\*\*"); // /oauth/\*\* 있는 모든 파일들은 시큐리티 적용을 무시한다.**  **Jwt Manager :**  **public class JwtManager {**  **public String makeJwtToken(MemberLoginReq req) {**  **Date now = new Date();**  **return Jwts.builder()**  **.setHeaderParam(Header.TYPE, Header.JWT\_TYPE) // (1)**  **.setIssuer("fresh") // (2)**  **.setIssuedAt(now) // (3)**  **.setExpiration(new Date(now.getTime() + Duration.ofMinutes(30).toMillis())) // (4)**  **.claim("id", req.getLoginId()) // (5)**  **.claim("email", req.getLoginId() + "@gmail.com") // (6)**  **.signWith(SignatureAlgorithm.HS256, "secret") // (6)**  **.compact();**  **}**  **public String makeJwtOauthToken(OauthAttributes req) {**  **Date now = new Date();**  **return Jwts.builder()**  **.setHeaderParam(Header.TYPE, Header.JWT\_TYPE) // (1)**  **.setIssuer("fresh") // (2)**  **.setIssuedAt(now) // (3)**  **.setExpiration(new Date(now.getTime() + Duration.ofMinutes(30).toMillis())) // (4)**  **.claim("id", req.getMemberId()) // (5)**  **.claim("email", req.getEmail()) // (6)**  **.signWith(SignatureAlgorithm.HS256, "secret") // (6)**  **.compact();**  **}**  **}**  **OAuthLoginController :**  **@Api(tags = "01. 사용자")**  **@RequiredArgsConstructor**  **@RestController**  **public class OAuthLoginController {**  **@Value("${spring.security.oauth2.client.registration.google.client-id}")**  **private String clientId;**  **@Value("${spring.security.oauth2.client.registration.google.client-secret}")**  **private String clientSecret;**  **@Value("${spring.security.oauth2.client.registration.google.redirect-uri}")**  **private String redirectUri;**  **private final MemberRepository memberRepository;**  **private final MemberService memberService;**  **private final ResponseService responseService;**  **@GetMapping(path = "/login/oauth2/code/google")**  **public void GoogleSignCallback(HttpServletRequest request, HttpServletResponse response) {**  **try {**  **// TODO Auto-generated method stub**  **String code = request.getParameter("code");**  **HttpHeaders headers = new HttpHeaders();**  **RestTemplate restTemplate = new RestTemplate();**  **headers.setContentType(MediaType.APPLICATION\_FORM\_URLENCODED);**  **MultiValueMap<String, String> parameters = new LinkedMultiValueMap<>();**  **parameters.add("code", code);**  **parameters.add("client\_id", clientId);**  **parameters.add("client\_secret", clientSecret);**  **parameters.add("redirect\_uri", redirectUri);**  **parameters.add("grant\_type", "authorization\_code");**  **HttpEntity<MultiValueMap<String,String>> rest\_request = new HttpEntity<>(parameters,headers);**  **URI uri = URI.create("https://www.googleapis.com/oauth2/v4/token");**  **JwtService jwtService = new JwtService(memberRepository);**  **ResponseEntity<String> rest\_reponse;**  **rest\_reponse = restTemplate.postForEntity(uri, rest\_request, String.class);**  **String bodys = rest\_reponse.getBody();**  **jwtService.createOAuthAccessToken(bodys);**  **System.out.println(bodys);**  **response.sendRedirect("http://localhost:9000/success");**  **} catch (Exception e) {**  **// TODO Auto-generated catch block**  **e.printStackTrace();**  **}**  **}**  **public BaseResult authOAuthUser(@RequestBody OAuth2UserRequest oAuth2UserRequest){**  **// 이전 boolean 통해 오류를 검출하는 방식으로 작동**  **// refactoring 이후 Exception 통한 예외처리로 로직 변경**  **// 없는 아이디, 잘못된 비밀번호에서 오류 발생.**  **MemberLoginReq memberLoginReq = new MemberLoginReq();**  **OauthService oauthService = new OauthService();**  **DefaultOAuth2User dor = oauthService.loadUser(oAuth2UserRequest);**  **memberLoginReq.setLoginId(dor.getAttribute("email"));**  **try{**  **return responseService.singleResult(memberService.oauthLogin(memberLoginReq).toLoginDto());**  **}catch (Exception e){**  **return responseService.failResult(**  **e.getMessage()**  **);**  **}**  **}**  **}**  **MemberService :**  **@Autowired**  **private final JwtService jwtService;**  **jwtService.createAccessToken(System.currentTimeMillis(), member.getLoginId());**  **// 임시적으로 세션을 등록**  **httpSession.setAttribute("user", new MemberSessionDto(member));**  **return memberRepository**  **.findByLoginId(memberLoginReq.getLoginId())**  **.orElseThrow(() -> new MemberException(ExceptionMessages.ERROR\_UNDEFINED));**  **}**  **@Transactional**  **public Member oauthLogin(MemberLoginReq memberLoginReq)**  **{**  **Member member = getMember(memberLoginReq.getLoginId());**  **memberRepository.updateLoginDate(new DateConfig().getDateTime(), memberLoginReq.getLoginId());**  **// HttpSession Or JWT 도입예정.**  **jwtService.createAccessToken(System.currentTimeMillis(), member.getLoginId());**  **OauthService :**  **@Autowired**  **private MemberRepository memberRepository;**  **@Autowired**  **private HttpSession httpSession;**  **@Autowired**  **private JwtService jwtService;**  **jwtService.createAccessToken(System.currentTimeMillis(), member.getLoginId());**  **2. JwtService 수정**  **JwtService :**  **@Slf4j**  **@Service**  **@Getter**  **@RequiredArgsConstructor**  **public class JwtService {**  **private final MemberRepository memberRepository;**  **public String createAccessToken(Long id, String username) {**  **return JWT.create()**  **.withSubject(JwtProperties.ACCESS\_TOKEN)**  **.withExpiresAt(new Date(System.currentTimeMillis() + JwtProperties.EXPIRATION\_TIME)) // 만료시간 10m**  **.withHeader(createHeader())**  **.withClaim(JwtProperties.ID, id)**  **.withClaim(JwtProperties.USERID, username)**  **.sign(Algorithm.HMAC512(JwtProperties.SECRET));**  **}**  **public String createOAuthAccessToken(String token){**  **return JWT.create()**  **.withSubject(JwtProperties.ACCESS\_TOKEN)**  **.withExpiresAt(new Date(System.currentTimeMillis() + JwtProperties.EXPIRATION\_TIME)) // 만료시간 10m**  **.withHeader(createHeader())**  **.withClaim(JwtProperties.ID, token)**  **.withClaim(JwtProperties.USERID, token)**  **.sign(Algorithm.HMAC512(JwtProperties.SECRET));**  **}**  **private Map<String, Object> createHeader() {**  **return Map.of(JwtProperties.HEADER\_PREFIX, "JWT" + JwtProperties.ACCESS\_TOKEN);**  **}**  **public String createRefreshToken() {**  **return JWT.create()**  **.withSubject(JwtProperties.REFRESH\_TOKEN)**  **.withHeader(createRefreshHeader())**  **.withExpiresAt(new Date(System.currentTimeMillis() + JwtProperties.REFRESH\_EXPIRATION\_TIME))**  **.sign(Algorithm.HMAC512(JwtProperties.SECRET));**  **}**  **private Map<String, Object> createRefreshHeader() {**  **return Map.of(JwtProperties.HEADER\_PREFIX, "JWT" + JwtProperties.REFRESH\_TOKEN);**  **}**  **진행 예정** | re |
| * **특이사항 / 협업 사항** | |