**2022년도 졸업프로젝트**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **작성자** | 김도현 | **작성일자** | 2022.07.11 |

* **금일 진행 상황**

|  |  |
| --- | --- |
| **진행 상황** | **비고** |
| **질문 및 진행방식에 대한 논의**  **진행 중**  **완료 1. Oauth2 로그인 기능 추가 (미완성)**  **Security Config :**  **.antMatchers("/api/v1/guest/\*\*")**  **.access("hasRole('ROLE\_GUEST') or hasRole('ROLE\_ADMIN')")**  **// GUEST**  **.access("hasRole('ROLE\_/USER')**  **or hasRole('ROLE\_ADMIN')").accessDeniedHandler(customAccessDeniedHandler)**  **.and() //추가**  **.oauth2Login()**  **// OAuth2기반의 로그인인 경우**  **.userInfoEndpoint()**  **// 로그인 성공 후 사용자정보를 가져온다**  **.userService(oauthService);**  **//사용자정보를 처리할 때 사용한다**  **OauthAttributes :**  **@Getter**  **public class OauthAttributes {**  **private Map<String, Object> attributes; // Oauth2 반환하는 유저 정보 Map**  **private String nameAttributeKey;**  **private String name;**  **private String email;**  **private String memberId;**  **@Builder**  **public OauthAttributes(Map<String, Object> attributes,**  **String nameAttributeKey, String name,**  **String email, String memberId)**  **{**  **this.attributes = attributes;**  **this.nameAttributeKey= nameAttributeKey;**  **this.name = name;**  **this.email = email;**  **this.memberId = memberId;**  **}**  **public static OauthAttributes of(String registrationId,**  **String userNameAttributeName,**  **Map<String, Object> attributes) {**  **// 여기서 네이버와 카카오 등 구분(ofNaver, ofKakao).**  **return ofGoogle(userNameAttributeName, attributes);**  **}**  **private static OauthAttributes ofGoogle(String userNameAttributeName,**  **Map<String, Object> attributes) {**  **return OauthAttributes.builder()**  **.name((String) attributes.get("name"))**  **.email((String) attributes.get("email"))**  **.attributes(attributes)**  **.nameAttributeKey(userNameAttributeName)**  **.build();**  **}**  **public Member toEntity() {**  **return Member.builder()**  **.name(name)**  **.email(email)**  **.role(Role.GUEST)**  **.build();**  **}**  **}**  **SessionUser :**  **@Getter**  **public class SessionUser implements Serializable {**  **private String name;**  **private String email;**  **public SessionUser(Member member) {**  **this.name = member.getName();**  **this.email = member.getEmail();**  **}**  **}**  **OauthService :**  **@Service**  **@RequiredArgsConstructor**  **public class OauthService extends DefaultOAuth2UserService {**  **private final MemberRepository memberRepository;**  **private final HttpSession httpSession;**  **@Transactional**  **public DefaultOAuth2User loadUser(OAuth2UserRequest userRequest) throws OAuth2AuthenticationException {**  **OAuth2UserService<OAuth2UserRequest, OAuth2User> delegate = new DefaultOAuth2UserService();**  **OAuth2User oAuth2User = delegate.loadUser(userRequest);**  **// OAuth2 서비스 id (구글, 카카오, 네이버)**  **String registrationId = userRequest**  **.getClientRegistration()**  **.getRegistrationId();**  **// OAuth2 로그인 진행 시 키가 되는 필드 값(PK)**  **String userNameAttributeName = userRequest**  **.getClientRegistration()**  **.getProviderDetails()**  **.getUserInfoEndpoint()**  **.getUserNameAttributeName();**  **// OAuth2UserService**  **OauthAttributes attributes = OauthAttributes.of(registrationId, userNameAttributeName, oAuth2User.getAttributes());**  **Member member = saveOrUpdate(attributes);**  **httpSession.setAttribute("Oauth2User", new SessionUser(member)); // SessionUser (직렬화된 dto 클래스 사용)**  **return new DefaultOAuth2User(Collections.singleton(new SimpleGrantedAuthority(member.getRoleKey())),**  **attributes.getAttributes(),**  **attributes.getNameAttributeKey());**  **}**  **// 유저 생성 및 수정 서비스 로직**  **private Member saveOrUpdate(OauthAttributes attributes){**  **Member member = memberRepository.findByEmail(attributes.getEmail())**  **.map(entity -> entity.update(attributes.getName()))**  **.orElse(attributes.toEntity());**  **return memberRepository.save(member);**  **}**  **}**  **진행 예정** | re |
| * **특이사항 / 협업 사항** | |