**2022년도 졸업프로젝트**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **작성자** | 김도현 | **작성일자** | 2022.07.07 |

* **금일 진행 상황**

|  |  |
| --- | --- |
| **진행 상황** | **비고** |
| **질문 및 진행방식에 대한 논의**  **진행 중**  **완료 1. JWT Refresh Token기능 추가**  **JwtAuthenticationFilter :**  **//RSA가 아닌 Hash암호방식**  **Member member = principalDetails.getMember();**  **String accessJwt = jwtService.createAccessToken(member.getMemberId(), member.getLoginId());**  **String refreshJwt = jwtService.createRefreshToken();**  **// login 성공 -> Refresh 토큰 재발급**  **jwtService.setRefreshToken(member.getLoginId(), refreshJwt);**  **response.addHeader(JwtProperties.HEADER\_PREFIX,JwtProperties.TOKEN\_PREFIX+ accessJwt);**  **response.addHeader(JwtProperties.REFRESH\_HEADER\_PREFIX,JwtProperties.TOKEN\_PREFIX+ refreshJwt);**  **System.out.println("인증완료");**  **JwtAuthorizationFilter :**  **public JwtAuthorizationFilter(AuthenticationManager authenticationManager, JwtService jwtService) {**  **super(authenticationManager);**  **this.memberRepository = memberRepository;**  **this.jwtService = jwtService;**  **}**  **// 인증이나 권한이 필요한 주소요청이 있을 때 해당 필터를 거친다.**  **@Override**  **protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain chain) throws IOException, ServletException {**  **System.out.println("인증이나 권한이 필요한 주소 요청이 됨");**  **String jwtHeader = request.getHeader(JwtProperties.HEADER\_PREFIX);**  **System.out.println("jwtHeader:"+jwtHeader);**  **try {**  **jwtService.checkHeaderValid(request);**  **String accessJwtToken = request**  **.getHeader(JwtProperties.HEADER\_PREFIX)**  **.replace(JwtProperties.TOKEN\_PREFIX, "");**  **String refreshJwtToken = request**  **.getHeader(JwtProperties.REFRESH\_HEADER\_PREFIX)**  **.replace(JwtProperties.TOKEN\_PREFIX, "");**  **jwtService.checkTokenValid(refreshJwtToken);**  **System.out.println("리프레쉬 토큰 회원 조회");**  **Member memberByRefreshToken = jwtService.getMemberByRefreshToken(refreshJwtToken);**  **String loginId = memberByRefreshToken.getLoginId();**  **Long id = memberByRefreshToken.getMemberId();**  **// JWT 토큰 서명을 통해서 서명이 정상이면 Authentication 객체를 만들어준다.**  **Authentication authentication =**  **new UsernamePasswordAuthenticationToken(principalDetails, null, principalDetails.getAuthorities());**  **// 강제로 시큐리티 세션에 접근해서 Authentication 객체를 저장.**  **SecurityContextHolder.getContext().setAuthentication(authentication);**  **try {**  **System.out.println("액세스 토큰 검증");**  **jwtService.checkTokenValid(accessJwtToken);**  **} catch (TokenExpiredException expired) {**  **System.out.println("ACCESS TOKEN REISSUE : " + JwtErrorCode.JWT\_ACCESS\_EXPIRED);**  **accessJwtToken = jwtService.createAccessToken(id, loginId);**  **response.addHeader(JwtProperties.HEADER\_PREFIX, JwtProperties.TOKEN\_PREFIX + accessJwtToken);**  **}**  **chain.doFilter(request, response);**  **PrincipalDetails principalDetails = new PrincipalDetails(memberByRefreshToken);**  **Authentication auth = new UsernamePasswordAuthenticationToken**  **(principalDetails, null, principalDetails.getAuthorities());**  **SecurityContextHolder.getContext().setAuthentication(auth);**  **} catch (CustomJwtException cusJwtExc) {**  **request.setAttribute(JwtProperties.EXCEPTION, cusJwtExc.getMessage());**  **} catch (TokenExpiredException ee) {**  **request.setAttribute(JwtProperties.EXCEPTION, JwtErrorCode.JWT\_REFRESH\_EXPIRED);**  **} catch (MalformedJwtException | UnsupportedJwtException mj) {**  **request.setAttribute(JwtProperties.EXCEPTION, JwtErrorCode.JWT\_NOT\_VALID);**  **} catch (Exception e) {**  **System.out.println("미정의 에러 : " + e);**  **System.out.println(e.getMessage());**  **request.setAttribute(JwtProperties.EXCEPTION, JwtErrorCode.JWT\_NOT\_VALID);**  **}**  **chain.doFilter(request, response);**  **JWT Service:**  **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**  **.withClaim(JwtProperties.ID, id)**  **.withClaim(JwtProperties.USERID, username)**  **.sign(Algorithm.HMAC512(JwtProperties.SECRET));**  **}**  **public String createRefreshToken() {**  **return JWT.create()**  **.withSubject(JwtProperties.REFRESH\_TOKEN)**  **.withExpiresAt(new Date(System.currentTimeMillis() + JwtProperties.REFRESH\_EXPIRATION\_TIME))**  **.sign(Algorithm.HMAC512(JwtProperties.SECRET));**  **}**  **@Transactional(readOnly = true)**  **public Member getMemberByRefreshToken(String token) {**  **return memberRepository.findByRefreshToken(token)**  **.orElseThrow(() -> new CustomJwtException(JwtErrorCode.JWT\_REFRESH\_EXPIRED.getMessage()));**  **}**  **@Transactional**  **public void setRefreshToken(String username, String refreshJwt) {**  **memberRepository.findByLoginId(username)**  **.ifPresent(member -> member.setRefreshToken(refreshJwt));**  **}**  **@Transactional**  **public void removeRefreshToken(String token) {**  **memberRepository.findByRefreshToken(token)**  **.ifPresent(member -> member.setRefreshToken(null));**  **}**  **@Transactional**  **public void logout(HttpServletRequest request) {**  **try {**  **checkHeaderValid(request);**  **String refreshJwtToken = request**  **.getHeader(JwtProperties.REFRESH\_HEADER\_PREFIX)**  **.replace(JwtProperties.TOKEN\_PREFIX, "");**  **removeRefreshToken(refreshJwtToken);**  **} catch (Exception e) {**  **throw new CustomJwtException(JwtErrorCode.JWT\_REFRESH\_NOT\_VALID.name());**  **}**  **}**  **public void checkHeaderValid(HttpServletRequest request) {**  **String accessJwt = request.getHeader(JwtProperties.HEADER\_PREFIX);**  **String refreshJwt = request.getHeader(JwtProperties.REFRESH\_HEADER\_PREFIX);**  **if (accessJwt == null) {**  **throw new CustomJwtException(JwtErrorCode.JWT\_ACCESS\_NOT\_VALID.getMessage());**  **} else if (refreshJwt == null) {**  **throw new CustomJwtException(JwtErrorCode.JWT\_REFRESH\_NOT\_VALID.getMessage());**  **}**  **}**  **public void checkTokenValid(String token) {**  **JWT.require(Algorithm.HMAC512(JwtProperties.SECRET))**  **.build()**  **.verify(token);**  **}**  **}**  **진행 예정** | re |
| * **특이사항 / 협업 사항** | |