3/13 개발일지

1. JWT 완성 (쿠키, 응답요청)

문제 상황

- JWT 토큰을 `Token` 엔티티에 저장하려 할 때 오류 발생.
- `tokenRepository.save(token);` 코드 실행 시 동일한 오류 발생, 주석 처리 시 저장되지 않고 실행만 됨.

해결 과정

- `TokenRepository`의 기본 키 타입 불일치 문제 해결을 위해 `TokenRepository` 인터페이스 수정.
- `Token` 엔티티의 `@Id` 필드 타입을 `Long`으로 변경하여 기본 키 타입 일치.
- `Token` 엔티티에 `@Builder`와 `@AllArgsConstructor` 어노테이션 추가하여 문제 해결.

변경된 `Token` 엔티티 스키마

```
@Entity
@NoArgsConstructor
@Getter
@Setter
@Builder
@AllArgsConstructor
public class Token {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id; // 이 부분을 새롭게 추가

    @Column(nullable = false, unique = true)
    private String email; // 유저의 이메일

    private String tokenValue;
    private LocalDateTime createdTime;
    private LocalDateTime expiredTime;
```

```
    private String tokenType;

    @OneToOne
    @JoinColumn(name = "email", referencedColumnName = "ema
il", insertable = false, updatable = false)
    private Member member;
}
```

2. 로그아웃 기능 구현 상세

프론트엔드 처리

NavBar 컴포넌트

- 사용자가 로그아웃 버튼을 클릭하면, `handleLogout` 함수가 호출됩니다.
- 함수는 로컬 스토리지와 쿠키에서 토큰을 추출하여 서버에 로그아웃 요청을 보냅니다.
- 성공적으로 로그아웃하면, 클라이언트에서도 로컬 스토리지와 쿠키의 토큰을 제거하고, 메인 페이지로 리다이렉션됩니다.

```
const handleLogout = async (e) => {
    e.preventDefault();
    const tokenFromLocalStorage = localStorage.getItem('tok
en');
    const tokenFromCookies = cookies.token;

    let token, provider;
    if (tokenFromLocalStorage) {
        token = tokenFromLocalStorage;
        provider = "web1";
    } else if (tokenFromCookies) {
        token = tokenFromCookies;
        provider = "google";
    }

    if (token) {
        try {
            const response = await fetch('<http://localhos
```

```
t:8080/member/logouts>', {
                method: 'POST',
                headers: {
                    'Authorization': `Bearer ${token}`,
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ "provider" : provide
r })
            });

            if (response.ok) {
                setIsLoggedIn(false);
                localStorage.removeItem('token');
                removeCookie('token');
                navigate ('/');
            }
        } catch (error) {
            console.error('An error occurred:', error);
        }
    }
};
```

백엔드 처리

로그아웃 컨트롤러

- 클라이언트로부터 로그아웃 요청을 받으면, 요청 헤더에서 토큰을 추출합니다.
- 토큰의 유효성과 만료 여부를 검사합니다.
- 토큰이 유효하거나 만료와 관계없이 로그아웃 로직을 수행합니다. 이는 사용자가 로그아웃 요청을 명시적으로 한 경우, 세션을 종료시키기 위함입니다.
- `TokenFactory`의 `deleteToken` 메소드를 호출하여 해당 사용자의 토큰을 데이터베이스에서 삭제합니다.

```
javaCopy code
@PostMapping("/member/logouts")
public ResponseEntity<?> logout(HttpServletRequest request,
@RequestBody Map<String, String> body) {
```

```
    String authorizationHeader = request.getHeader("Authori
zation");
    String tokenValue = authorizationHeader != null ? autho
rizationHeader.replace("Bearer ", "") : null;
    String provider = body.get("provider");

    TokenFactory tokenFactory = applicationContext.getBean
(provider + "TokenFactory", TokenFactory.class);
    Map<String, Object> decodedToken = tokenFactory.decodeT
oken(tokenValue);

    Boolean isChecked = tokenFactory.checkToken(tokenValue,
Arrays.asList("USER", "ADMIN"));

    if (!isChecked) {
        tokenFactory.deleteToken(String.valueOf(decodedToke
n.get("oauth2Id")));
        return ResponseEntity.ok().body(Map.of("message",
"로그아웃 성공"));
    } else {
        tokenFactory.deleteToken(String.valueOf(decodedToke
n.get("oauth2Id")));
        return ResponseEntity.ok().body(Map.of("message",
"로그아웃 성공"));
    }
}
```

AbstractTokenFactory 클래스

- `deleteToken` 메소드는 JDBC 템플릿을 사용하여 데이터베이스에서 사용자의 토큰을 삭제합니다.
- `oauth2Id`를 기반으로 사용자의 이메일을 조회한 후, 이메일에 해당하는 토큰 레코드를 삭제합니다.

```
javaCopy code
@Transactional
public void deleteToken(String oauth2Id) {
```

```
    String email = jdbcTemplate.queryForObject("SELECT emai
l From member WHERE oauth2id = ?", String.class, oauth2Id);
    if (email != null) {
        jdbcTemplate.update("DELETE FROM token WHERE email
= ?", email);
    }
}
```

이 구현은 로그아웃 요청을 받았을 때, 서버 측에서 사용자의 인증 토큰을 삭제하여 세션을 종료시키는 로직을 포함합니다. 클라이언트와 서버 모두에서 필요한 처리를 수행하여 안전하게 사용자를 로그아웃 시키며, 이 과정은 사용자의 보안과 직결된 중요한 기능입니다.