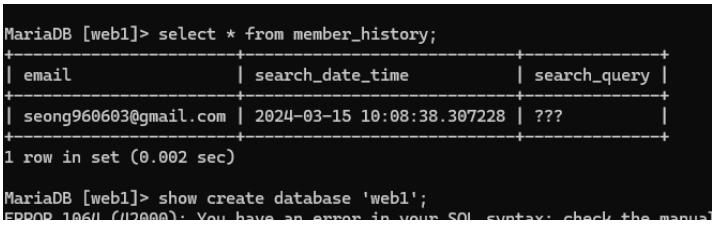
3/15 개발일지

포토패쓰 추가

문제 상황 및 해결 과정

쿼리 한국어 입력 문제

- **문제**: 한국어로 쿼리를 입력했을 때, 데이터베이스에 제대로 저장되지 않는 문제가 발생했다. 이 문제는 사용자의 입력 정보가 데이터베이스에 제대로 반영되지 않아서, 데이터 처리와 관리에 어려움을 겪게 만들었다.



- **원인**: `innodb_large_prefix` 설정이 비활성화되어 있어, `utf8mb4` 인코딩을 사용할 때 인덱스 키 길이 제한을 초과하는 문제 발생.
- **해결**: `innodb_large_prefix` 옵션을 활성화하여 인덱스 키의 최대 길이를 3072바이트로 확장. 이를 통해 한국어 입력 문제 해결.

```
[mysqld]
innodb_large_prefix=1
innodb_file_format=Barracuda
innodb_file_per_table=1
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
# Settings user and group are ignored when systemd is used.
# If you need to run mysqld under a different user or group,
# customize your systemd unit file for mariadb according to the
# instructions in http://fedoraproject.org/wiki/Systemd
bind-address = 0.0.0.0

[mysqld_safe]
log-error=/var/log/mariadb/mariadb.log
pid-file=/var/run/mariadb/mariadb.pid

#
# include all files from the config directory
#
!includedir /etc/my.cnf.d

"my.cnf" 23L, 668C
```

복합키 매핑 문제

- **문제**: 복합키를 사용하여 `MemberHistory`와 `Member` 엔티티를 매핑하려고 할 때, 매핑이 정상적으로 이루어지지 않는 문제 발생.
  - 아래 코드로 매핑하려하니 계속 문제 발생함.

```
    @EmbeddedId
    private MemberHistoryId id;

    private String searchQuery;

    @ManyToOne(fetch = FetchType.LAZY)
    @MapsId("email") // 복합 키의 일부를 외래 키로 매핑
    @JoinColumn(name = "email", referencedColumnName = "email") //
    private Member member;

    public MemberHistory(String searchQuery, Member member, LocalDa
        this.id = new MemberHistoryId(member.getEmail(), searchDate
        this.searchQuery = searchQuery;
        this.member = member;
    }
```

- **원인**: JPA에서 복합키 매핑 시 발생하는 제한으로 인해, 예상대로 매핑되지 않음. JPA의 한계라고 생각함.
- **해결**: 복합키 대신 `autoincrement`를 사용하여 문제 해결.

```
@Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id; // New auto-incrementing ID

    private String searchQuery;

    @Column(nullable = false, length = 191) // 동일하게 적용
    private String email; // 유저의 이메일을 기본키로 사용

    private LocalDateTime searchDateTime; // Directly included in t

    @ManyToOne
    @JoinColumn(name = "email", referencedColumnName = "email", ins
    private Member member;
```

Spring Security URL 필터링 문제

- **문제**: "Rejecting request due to: The request was rejected because the URL contained a potentially malicious String ";" 오류 발생.
- **원인**: Spring Security가 세미콜론(`;`)을 포함한 URL을 잠재적으로 악의적인 것으로 판단하고 요청을 거부함.
- **해결 과정**:
  - Spring Security 필터 체인 전에 커스텀 필터를 추가하여 실제로 받는 요청 URL 로깅.

```
public class CustomRequestLoggingFilter implements Filter {

    private static final Logger log = LoggerFactory.getLogger(Custo

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResp
        HttpServletRequest request = (HttpServletRequest) servletRe
        log.info("Request URL: {}, Method: {}", request.getRequestU

        // 요청 헤더 로깅 (예시)
        log.info("Authorization Header: {}", request.getHeader("Aut

        filterChain.doFilter(servletRequest, servletResponse);
    }

    // init 및 destroy 메소드는 필터 인터페이스를 구현하기 위해 필요하지만, 여
    @Override
    public void init(FilterConfig filterConfig) {
    }

    @Override
    public void destroy() {
    }
}
```

  - 문제의 원인이 되는 URL `http://localhost:8080/login;jsessionid=...`을 확인.
  - 2024-03-15T15:55:53.383+09:00 DEBUG 26552 --- [nio-8080-exec-7] o.s.security.web.FilterChainProxy : Securing OPTIONS /error
    2024-03-15T15:55:53.383+09:00 DEBUG 26552 --- [nio-8080-exec-7] o.s.s.w.a.AnonymousAuthenticationFilter : Set SecurityContextHolder to anonymous SecurityContext
    2024-03-15T15:55:53.383+09:00 DEBUG 26552 --- [nio-8080-exec-7] o.s.s.w.s.HttpSessionRequestCache : Saved request http://localhost:8080/error?continue to session

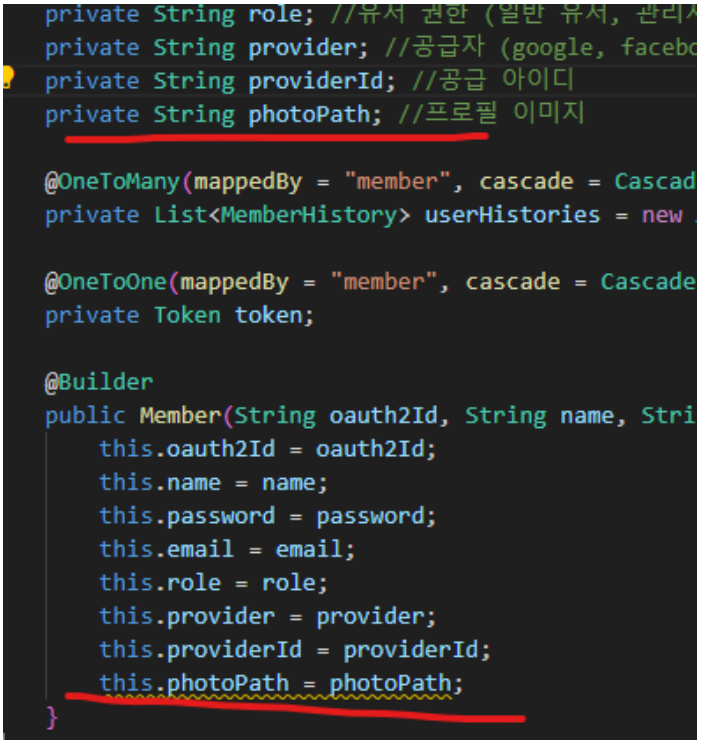
2024-03-15T15:55:53.383+09:00 DEBUG 26552 --- [nio-8080-exec-7] o.s.s.web.DefaultRedirectStrategy : Redirecting to **http://localhost:8080/login;jsessionid=1359EA9DD7CC29D112994313F5A74212**
2024-03-15T15:55:53.389+09:00 DEBUG 26552 --- [nio-8080-exec-3] s.s.w.f.HttpStatusRequestRejectedHandler : Rejecting request due to: The request was rejected because the URL contained a potentially malicious String ";"

→ 여기서 login;jsessionid로 리다이렉트하는 과정에서 문제가 생김

- URL의 잘못된 부분(`useprofile` 오타) 수정 후 문제 해결.
- 생각보다 오타로 오류가 생기는 경우가 정말 많다는것을 느낌

추가 작업: 포토패쓰 추가

- 포토패쓰를 프로젝트에 성공적으로 추가하여 사용자 프로필 이미지 기능 개선.



- Name, Photo에 대해 새롭게 변경사항이 추가되어 update로 U작업을 실시한다면

```
<button onClick={toggleProfileUpdateModal}>Edit Profile</button>
                            {showProfileUpdateModal && (
                                <div className="modal-backdrop">
                                    <div className="modal-content">
                                        <form onSubmit={handleProfileU
                                            <label htmlFor="name">Name
                                            <input type="text" id="nam
                                            <label htmlFor="photo">Pho
                                            <input type="file" id="pho
```

```
                                        <button type="submit">Upda
                                        <button type="button" onCl
                                    </form>
                                </div>
                            </div>
                        )}
```

```
const handleProfileUpdateSubmit = async (event) => {
        event.preventDefault();
        const formData = new FormData(event.target);

        const updateProfileUrl = 'http://localhost:8080/memberpage/use
```

```
당신의 토큰은 이 작업을 실행하기에 유효합니다.
directory경로 = C:\projects\web1\src\main\resources\static\user_profile_photos
save경로 = C:\projects\web1\src\main\resources\static\user_profile_photos\google_117558175824592101272_profile_photo.png
save경로 = C:\projects\web1\src\main\resources\static\user_profile_photos\google_117558175824592101272_profile_photo.png
2024-03-18T12:11:17.930+09:00 DEBUG 22992 --- [nio-8080-exec-7] o.s.s.w.a.AnonymousAuthenticationFilter  : Set SecurityContextHolder to anonymous SecurityContext
```

위 터미널의 경로로 이미지를 새롭게 업데이트함

Controller

```
@PostMapping("/memberpage/userprofile/update")
    public ResponseEntity<?> updateUserProfile(
            HttpServletRequest request,
            @RequestParam("photo") MultipartFile photo,
            @RequestParam("name") String name) {
        String authorizationHeader = request.getHeader("Authorization"
        String tokenValue = authorizationHeader != null ? authorizatio

        if (tokenValue == null) {
            return ResponseEntity.badRequest().body("Token is missing.
        }

        userProfile.updateUserProfile(tokenValue, photo, name);

        return ResponseEntity.ok().body("User profile updated successf
    }
```

UserProfile 클래스의 updateUserProfile()

```
    public void updateUserProfile(String tokenValue, MultipartFile pho
        if (tokenValue == null) {
            System.out.println("토큰값이 없습니다. 이 작업을 실행할 수 없습니다
```

```
            return; // 조기 반환
        }

        TokenFactory tokenFactory = applicationContext.getBean("google

        // 이 메소드는 user와 admin인 경우 실행 가능함
        List<String> allowedRoles = Arrays.asList("USER", "ADMIN");
        Map<String, Object> decodedToken = null;

        try{
            decodedToken = tokenFactory.decodeToken(tokenValue);
            if (decodedToken.isEmpty()) {
                System.out.println("토큰 디코드 중 오류가 발생했습니다.");
                return;
            } else {
                System.out.println("토큰 디코드 성공");
            }
        } catch (Exception e) {
            System.out.println("토큰 디코드 중 오류가 발생했습니다.");
            return;
        }


        String oauth2Id = (String) decodedToken.get("oauth2Id");

        Boolean isChecked = tokenFactory.checkToken(tokenValue, allowe
        Boolean isExpired = tokenFactory.isExpired(String.valueOf(deco

        if (isChecked && !isExpired) {
            System.out.println("당신의 토큰은 이 작업을 실행하기에 유효합니다.'

            try {
                String sql = "UPDATE member SET name = ?, photo_path =
                String photoPath = null;
                if (photo != null && !photo.isEmpty()) {
                    photoPath = savePhotoAndGetPath(photo, oauth2Id);
                }

                jdbcTemplate.update(sql, newName, photoPath, oauth2Id)
            } catch (Exception e) {
                System.out.println("프로필을 업데이트중 오류발생함 :  " + e.
            }
        } else {
            System.out.println("당신의 토큰은 이 작업을 실행하기에 유효하지 않
```

```
        }
    }



    private String savePhotoAndGetPath(MultipartFile photo, String oau
        try {
            // 변경된 절대 경로
            String directoryPath = "C:\\projects\\web1\\src\\main\\res
            String photoFileName = oauth2Id + "_profile_photo" + getEx
            Path directory = Paths.get(directoryPath);

            System.out.println("directory경로 = " + directory);

            // Check if the directory exists, create it if it doesn't
            if (!Files.exists(directory)) {
                Files.createDirectories(directory);
            }

            Path savePath = directory.resolve(photoFileName);
            System.out.println("save경로 = " + savePath);
            System.out.println("save경로 = " + savePath.toString());

            // Save the photo
            Files.copy(photo.getInputStream(), savePath, StandardCopyO

            // 절대 경로 대신 웹 접근 가능한 상대 경로 반환
            String accessiblePath = "/user_profile_photos/" + photoFil
            return accessiblePath;
        } catch (Exception e) {
            System.out.println("Failed to save photo: " + e.getMessage
            return null;
        }
    }


    private String getExtension(String fileName) {
        return fileName.substring(fileName.lastIndexOf("."));
```

배운 점

- `innodb_large_prefix` 옵션의 중요성과 활성화 방법에 대해 학습.
- 복합키 사용 시 발생할 수 있는 문제와 대안적인 접근 방법에 대해 이해.

- Spring Security의 URL 필터링 기능과 커스텀 필터를 통한 문제 진단 방법에 대해 학습.
- 실제 개발 과정에서 발생할 수 있는 다양한 문제 상황에 대한 해결 방안 모색 경험 향상.