ChatGPT

2번(PostPermission 테이블 기반) 방식 전환 보고서

변경 개요

기존 4번 방식(JPA 조건 필터링 방식)은 `Post` 엔티티 내부에 `readableEmployees`, `readableTeams` 필드를 포함시켜 사용자/팀 단위로 열람 가능 여부를 판단했음. 그러나 수정/삭제 권한과 같은 세부 제어가 어려워 **2번 방식(PostPermission 테이블 방식)** 으로 구조 전환.

⚙ 변경 전 구조 (4번 방식)

- Post 엔티티 내부에 다음 필드 존재:

```
List<Employee> readableEmployees;
List<Team> readableTeams;
```

- 권한 체크는 `PostService`에서 필터 조건으로 직접 수행하거나, `PostRepository`에서 JPA 쿼리 조건을 이용해 필터링.
- 문제점:
- 열람만 가능하고, 수정/삭제 권한은 별도 구조가 없음
- 수정/삭제까지 고려하면 구조가 매우 복잡해짐
- 권한의 종류를 동적으로 추가/관리하기 어려움

변경 후 구조 (2번 방식)

1. `PostPermission` 엔티티 신설

```
@Entity
public class PostPermission {
  Post post;
  Employee employee; // 또는 Team team;
  PermissionType permissionType; // 열람/수정/삭제
}
```

- 권한 종류를 테이블에서 직접 관리하므로 확장성이 뛰어남 - VIEW / EDIT / DELETE 권한을 구분하여 저장 가능

2. `PermissionType` 열거형 생성

```
public enum PermissionType {
  VIEW, EDIT, DELETE
}
```

3. 권한 쿼리 로직 전환

`PostRepository` → 더 이상 필터링용 쿼리 사용 안 함

```
// PostPermissionRepository
List<PostPermission> findByEmpOrTeamWithPermission(...);
```

4. 서비스 계층 변경

- 기존 `post.getReadableEmployees().contains()` 등 직접 접근 방식 제거
- `PostPermission` 테이블 기반 권한 조회
- 예: 열람 가능한 게시물 목록

```
List<PostPermission> = repo.findByEmpOrTeamWithPermission(emp, team, VIEW);
```

변경 효과

항목	4번 방식	2번 방식
열람권한 설정	O (단순)	O (세밀)
수정/삭제 권한	X (불편)	O (권한 분리 가능)
권한 종류 확장	어려움	용이 (enum + 테이블 기반)
구조 이해도	직관적이나 기능 제한	처음엔 낯설 수 있으나 명확한 구조
유지보수	중장기적 복잡도 증가	명확한 책임 분리로 용이함

권장 사항

- `Post` 엔티티의 `readableEmployees`, `readableTeams` 필드는 제거 권장
- 향후에는 글쓰기, 수정, 삭제 기능을 위한 권한 검증 메서드를 `PostService`에 확장 예정
- 관리자는 PostPermission 테이블만으로 전체 권한 현황을 파악 가능

결론

초기 구현된 4번 방식은 단순하지만 확장성 및 유지보수 측면에서 한계를 가짐.

2번 방식은 구조가 명확하고, VIEW / EDIT / DELETE 등을 구분할 수 있어 실무에서 발생할 다양한 요구사항에 유연하게 대응 가능함.

따라서 현재 프로젝트의 권한 설계는 2번 방식으로 통일하고, 이에 따라 전체 코드를 구조적으로 리팩토링함.