Record 뼈대 코드

Posts

Records

```
@Getter
@NoArgsConstructor
@Entity
public class Records extends BaseTimeEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int record_id;

    @Column(length = 500, nullable = true)
    private int cost;
    //private Cost cost; - 다른 Cost클래스(테이블) 있을때

    @Column(length = 255, nullable = false)
    private String record_title;

    @Column(nullable=false)
    private String location;

    @Column(nullable = false)
    private LocalDate start_date;

    @Column(nullable = false)
    private LocalDate end_date;

    /*@Column(columnDefinition = "TEXT", nullable = false)
    private String content;*/


    @Builder
    public Records(int cost_id, String record_title, String location, Date start_date, Date end_date) {
        this.cost_id = cost_id;
        this.record_title = record_title;
        this.location = location;
        this.start_date = start_date;
        this.end_date = end_date;
    }

    public void update(Long cost_id, String record_title, String location, Date start_date, Date end_date) {
```

```
        this.cost_id = cost_id;
        this.record_title = record_title;
        this.location = location;
        this.start_date = start_date;
        this.end_date = end_date;
    }
}
```

RecordContent

```
@Getter
@NoArgsConstructor
@Entity
public class RecordContent extends BaseTimeEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer date;

    @ManyToOne
    @JoinColumn(name = "record_id")
    private Records record;

    @Column(columnDefinition = "TEXT")
    private String content;

    @Column(length = 255)
    private String hashtag;

    @OneToMany(mappedBy = "record", cascade = CascadeType.ALL)
    private List<RecordContent> recordContents = new ArrayList<>();
    //record_content 테이블과의 관계 나타내줘야한다

    @Builder
    public RecordContent(Integer date, Records record, String content, String hashtag) {
        this.date = date;
        this.record = record;
        this.content = content;
        this.hashtag = hashtag;
    }
}
```

RecordsRepository

```
public interface PostsRepository extends JpaRepository<Posts, Long> {

    @Query("SELECT p FROM Posts p ORDER BY p.id DESC")
    //어떤 쿼리가 들어가야할까
    List<Records> findAllDesc();


}
```

Service

RecordsService

```
@RequiredArgsConstructor
@Service
public class RecordsService {
    private final RecordsRepository recordsRepository;

    @Transactional
    public int save(PostsSaveRequestDto requestDto) {
        return postsRepository.save(requestDto.toEntity()).getId();
    }

    @Transactional
    public int update(int record_id, PostsUpdateRequestDto requestDto) {
        Posts posts = postsRepository.findById(id)
                .orElseThrow(() -> new IllegalArgumentException("해당 사용자가 없습니다. id=" + id));

        posts.update(requestDto.getTitle(), requestDto.getContent());

        return record_id;
    }

    @Transactional
    public void delete (int record_id) {
        Posts records = postsRepository.findById(record_id)
                .orElseThrow(() -> new IllegalArgumentException("해당 사용자(기록이?)가 없습니다. id=" + record_id));

        postsRepository.delete(records);
    }

    @Transactional(readOnly = true)
    public RecordsResponseDto findByRecordId(int record_id) {
        Records entity = recordsRepository.findByRecordId(record_id)
                .orElseThrow(() -> new IllegalArgumentException("해당 사용자가 없습니다. id=" + record_id));

        return new RecordsResponseDto(entity);
    }

    /*Post 리스트 받는것*/
    @Transactional(readOnly = true)
    public List<PostsListResponseDto> findAllDesc() {
        return postsRepository.findAllDesc().stream()
                .map(PostsListResponseDto::new)
                .collect(Collectors.toList());
    }
}
```

Web

RecordsPostApiController

```
import java.util.List;

@RequiredArgsConstructor
@RestController
public class RecordsApiController {

    private final RecordsService RecordsService;

    @PostMapping("/api/v1/Records")
    public Long save(@RequestBody RecordsSaveRequestDto requestDto) {
        return RecordsService.save(requestDto);
```

```
    }

    @PutMapping("/api/v1/Records/{record_record_id}")
    public Long update(@PathVariable Long record_id, @RequestBody RecordsUpdateRequestDto requestDto) {
        return RecordsService.update(record_id, requestDto);
    }

    @DeleteMapping("/api/v1/Records/{record_id}")
    public Long delete(@PathVariable Long record_id) {
        RecordsService.delete(record_id);
        return record_id;
    }

    @GetMapping("/api/v1/Records/{record_id}")
    public RecordsResponseDto findByRecordId(@PathVariable Long record_id) {
        return RecordsService.findByRecordId(record_id);
    }

    @GetMapping("/api/v1/Records/list")
    public List<RecordsListResponseDto> findAll() {
        return RecordsService.findAllDesc();
    }
}
```

dto

RecordsListResponseDto	record_id, cost_id, record_title, location, start_date, end_date	
RecordsResponseDto	“	
RecordsSaveRequestDto	"	
RecordsUpdateRequestDto	update는 ‘record_title, location, start_date, end_date'만 해당하지 않을까.	

```
//Dto들 구조
/*

@Getter
@NoArgsConstructor
public class RecordsSaveRequestDto {

    private 필요한 속성들

    @Builder
    public RecordsSaveRequestDto(타입 속성명){
       this.속성 = 속성

    }

    public Records toEntity(){
        //DTO 객체를 엔티티 객체로 변환
        return Records.builder()
                .record_title(record_title)
                .location(location)
                //.뭐가 더들어갈까. 위랑 연동
                .build();
    }

}

* RecordContent에 대해서도 똑같이.
*
@Getter
@NoArgsConstructor
```

```
public class RecordContentSaveRequestDto {

    private Integer date; // 우리 스키마에 int라 되어있음. 여행일자들 date로 하는거아님? 뭐더라
    private String content;
    private String hashtag;

    public RecordContent toEntity(){
        return RecordContent.builder()
                .date(date)
                .content(content)
                .hashtag(hashtag)
                .build();
    }


}

*
*
* */
```

RecordsSaveRequestDto

```
//여기 안에 Records와 RecordContent Save관련된걸 같이

@Getter
@NoArgsConstructor
public class RecordsSaveRequestDto {

    private Cost cost_id;
    private String record_title;
    private String location;
    private Date start_date;
    private Date end_date;
    private List<RecordContentSaveRequestDto> recordContents;

    @Builder
    public RecordsSaveRequestDto(Strint record_title, String location){
        //여기서 얼마나 만든단건지 모르겠네
        this.record_title = record_title;
        this.location = location;

    }

    public Records toEntity(){
        //DTO 객체를 엔티티 객체로 변환
        return Records.builder()
                .record_title(record_title)
                .location(location)
                //.뭐가 더들어갈까. 위랑 연동
                .build();
    }

}

@Getter
@NoArgsConstructor
public class RecordContentSaveRequestDto {

    private Integer date; // 우리 스키마에 int라 되어있음. 여행일자들 date로 하는거아님? 뭐더라
    private String content;
    private String hashtag;

    public RecordContent toEntity(){
        return RecordContent.builder()
                .date(date)
                .content(content)
                .hashtag(hashtag)
                .build();
```

```
    }

}
```

RecordsListResponseDto

RecordsResponseDto

RecordsUpdateRequestDto