1.개요

주차	10주차	날짜	11.18
주제	Quick Sort 정석 SpringBoot Test SpringBoot Data JPA		
목표			
실습	1. Controller Mock Test만들기 2. Service Mock Test만들기		
Links	java-algorithm Kyeongrok/java-algorithm (github.com) 코딩테스트 고득점 Kit | 프로그래머스 스쿨 (programmers.co.kr)		

10주차 알고리즘 재귀

https://docs.google.com/document/d/1WZUKse7K2uThJjo3vYUOgPs27-l9rGGWsR3UL0cjNUM/edit#

JPA Mapping

9.1 연관관계 매핑 종류와 방향

연관관계를 맺는 두 엔티티 간에 생성할 수 있는 연관관계의 종류는 다음과 같습니다.

- One To One: 일대일(1:1)
- One To Many: 일대다(1:N)
- Many To One: 다대일(N:1)
- Many To Many: 다대다(N:M)

실제로는 ManyToOne - One To Many를 90% 사용합니다. One To One가끔
ManyToMany는 OneToMany, ManyToOne으로 대체 해서 사용합니다.

여러 테이블끼리 연관관계가 설정돼 있어 여러 left outer join이 설정되는 것은 괜찮으나 위와 같이 양쪽에서 외래키를 가지고 left outer join이 두 번이나 수행되는 경우는 효율성이 떨어집니다. 실제 데이터베이스에서도 테이블 간 연관관계를 맺으면 한쪽 테이블이 외래키를 가지는 구조로 이뤄집니다. 바로 앞에서 언급한 '주인' 개념입니다.

JPA에서도 실제 데이터베이스의 연관관계를 반영해서 한쪽의 테이블에서만 외래키를 바꿀 수 있도록 정하는 것이 좋습니다. 이 경우 엔티티는 양방향으로 매핑하되 한쪽에게만 외래키를 줘야 하는데, 이때 사용되는 속성 값이 mappedBy입니다. mappedBy는 어떤 객체가 주인인지 표시하는 속성이라고 볼 수 있습니다. 예제 9.7과 같이 Product 객체에 mappedBy 속성을 추가해 보겠습니다.

```
01  @OneToOne(mappedBy = "product")
02  @ToString.Exclude
03  private ProductDetail productDetail;
```

book(책)

```
create table `db-exercies2`.book
(
   id           int auto_increment,
   name         varchar(45) not null comment '책 제목',
   publisher_id int         null comment '출판사 id',
   author_id    int         null comment '저자 id',
   constraint book_pk
       primary key (id)
);
```

BookEntity

```
@Entity
@Getter
@AllArgsConstructor
```

```java
@NoArgsConstructor
public class Book {
    @Id
    private Long id;
    private String name;
    private Long authorId;
}
```

author(저자)

```sql
create table `db-exercies2`.author
(
    id   int auto_increment,
    name varchar(45) not null,
    constraint author_pk
        primary key (id)
)
    comment '저자';
```

AuthorEntity

```java
@Entity
@Getter
@AllArgsConstructor
@NoArgsConstructor
public class Author {
    @Id
    private Long id;
    private String name;
}
```

@ManyToOne
Book(fk: author_id)

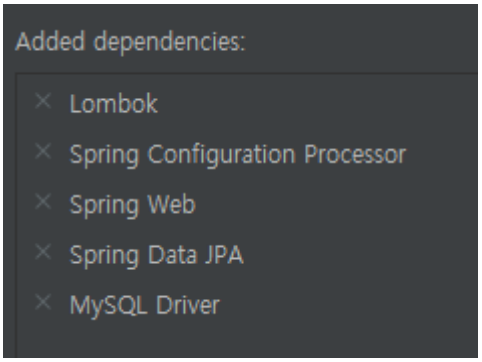
@OneToMany
Author

@OneToMany @ManyToOne
Hospital – Review(fk:hospital_id)

실습
프로젝트 새로 만들어서 build.gradle
위 table을 Entity로 만들기

1. Book - Author

11:30까지



application.yml

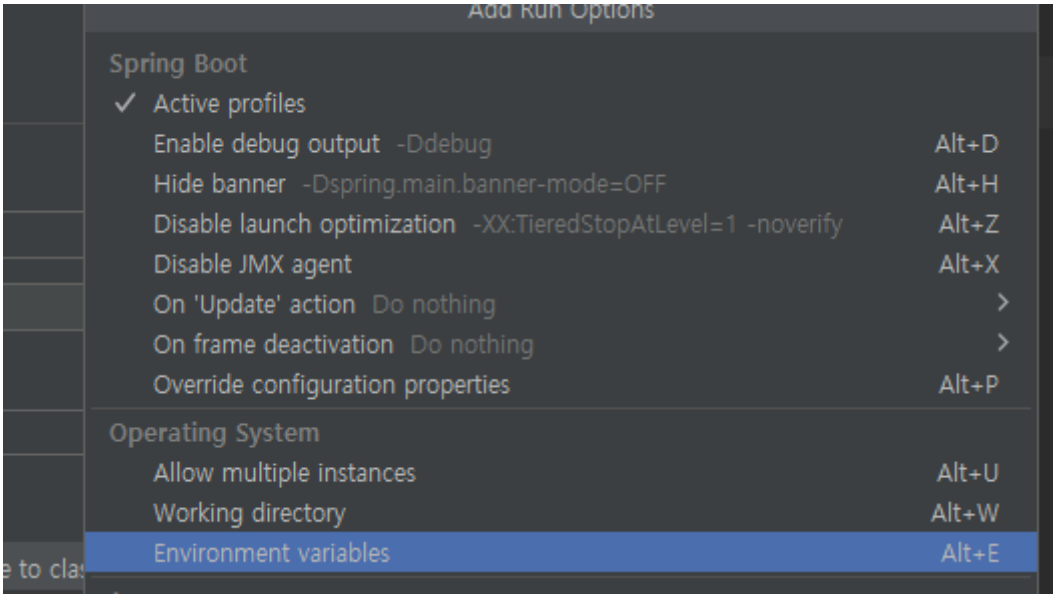
```yaml
server:
 servlet:
   encoding:
     force-response: true
spring:
 jpa:
   hibernate:
     ddl-auto: update
   show-sql: true
```

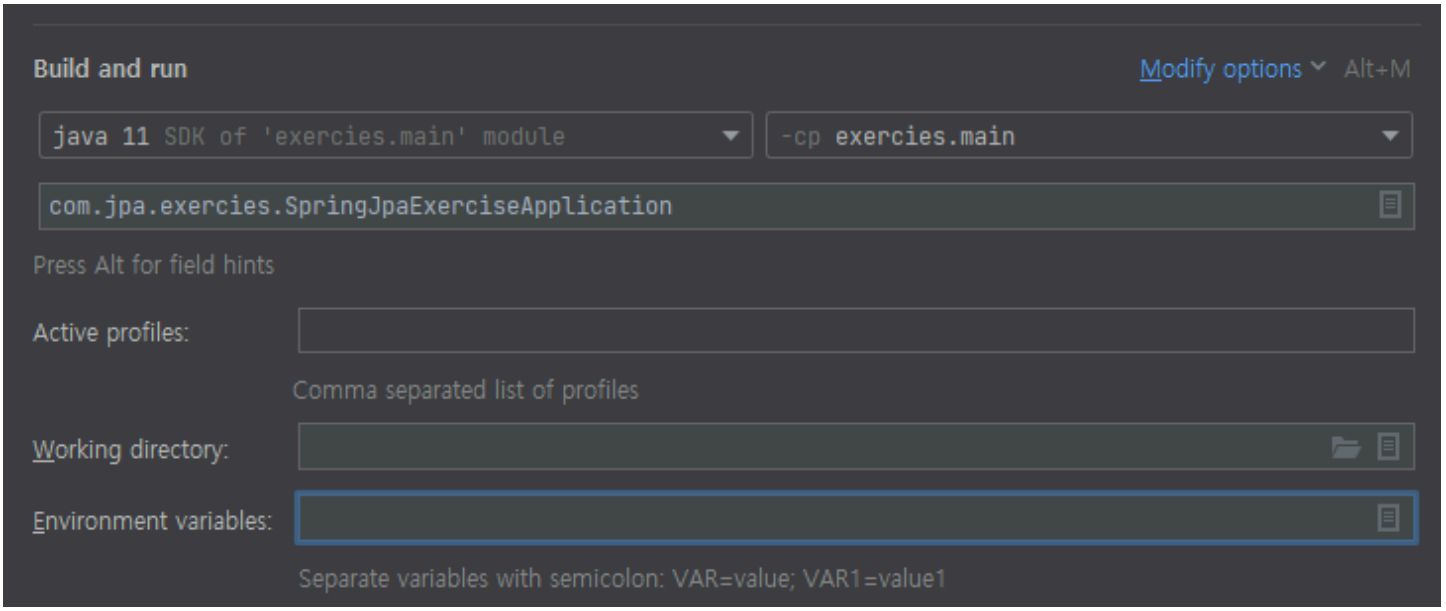
```
datasource:
  driver-class-name: com.mysql.cj.jdbc.Driver
  url: localhost
  username: root
  password: 1q2w3e4r
```

새로 만든 DB에 붙여야 합니다.

SPRING_DATASOURCE_URL=jdbc:mysql://ec2-3-38-111-117.ap-northeast-2.compute.amazonaws.com:3306/db-exercise2;SPRING_DATASOURCE_PASSWORD=1q2w3e4r

Environment variables안보이는 경우





실행 하기 전에 주의

`ddl-auto: update`
인 경우 데이터가 없어질 수 있습니다.

	id	name	publisher_id	author_id
1	1	토비의 스프링3	2	2
2	2	스프링부트 핵심가이드	1	3
3	3	말랑말랑 알고리즘	3	1
4	4	한입에 웹 크롤링	3	1
5	5	MySql로 배우는 데이터베이스 개론고	4	4

publisher_id가 사라질 수 있으니 주의*

1,토비의 스프링3,2,2
2,스프링부트 핵심가이드,1,3
3,말랑말랑 알고리즘,3,1

4,한입에 웹 크롤링,3,1
5,MySql로 배우는 데이터베이스 개론과 실습,4,4

@Table(name = "book2") 를 추가 하는 경우

book2테이블이 없기 때문에 새로 만듭니다.

```
@Entity
@Getter
@AllArgsConstructor
```

```java
@NoArgsConstructor
@Table(name = "book2")
public class Book {
    @Id
    private Long id;
    private String name;
    private Long authorId;
}
```

Hibernate: create table book2 (id bigint not null, author_id bigint, name varchar(255), primary key (id)) engine=InnoDB

BookRepository.java

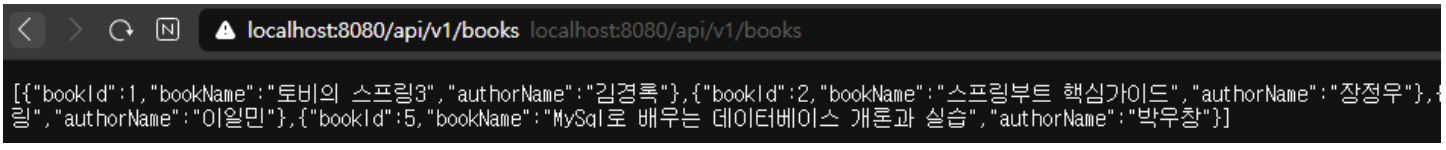
```java
public interface BookRepository extends
JpaRepository<Book, Long> {
}
```

AuthorRepository.java

```java
public interface AuthorRepository extends
JpaRepository<Author, Long> {
}
```

요구사항

1. 책 목록을 보여주는 API Endpoint
2. 책 목록을 보여줄 때 저자 이름이 보여야 합니다.



1.요구사항 분석
2.DB or API설계
3.테스트 코드 개발
3.기능 구현

4.배포

1:10까지 - git commit
1:20까지 배포

2:20까지 배포

Mapping안쓰고 Join구현

https://youtu.be/luk5YIVIBLE

빌드 하고 배포하기

https://youtu.be/p-_joOa0lH0

1시간

2. Hospital - Review

[도전]
Swagger추가
출판사 추가
공저자가 n명 있다면?

1

ManyToOne Mapping추가

ManyToOne 단방향 Mapping

Book에만 Mapping이 있고 Author에는 없기 때문

```java
@Entity
@Getter
@AllArgsConstructor
@NoArgsConstructor
public class Book {
    @Id
    private Long id;
    private String name;
    // private Long authorId; 이 줄 삭제

    @ManyToOne
    @JoinColumn(name = "author_id")
    private Author author;
}
```

JPA가 Foreign Key를 걸어줍니다.
Hibernate: alter table book add constraint FKklnrv3weler2ftkweewlky958 foreign key (author_id) references author (id)
//book에 author를 가져와서 authorId 필요없음 이제

BookService.java

```java
@Service
public class BookService {

    private final BookRepository bookRepository;

    public BookService(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public List<BookResponse> findBooks(Pageable pageable) {
        Page<Book> books = bookRepository.findAll(pageable);
        List<BookResponse> bookResponses = books.stream()
                .map(book ->
BookResponse.of(book)).collect(Collectors.toList());
        return bookResponses;
    }
}
```

```java
@Builder
@Getter
@NoArgsConstructor
@AllArgsConstructor
public class BookResponse {
    private Long bookId;
    private String bookName;
    private String authorName;

    public static BookResponse of(Book book) {
        return BookResponse.builder()
                .bookId(book.getId())
                .bookName(book.getName())
                .authorName(book.getAuthor().getName())
                .build();
    }
}
```

readme없으신 분들 꼭 추가 해주세요.
readme에 ec2접속 주소를 꼭 넣어주세요.

2:43까지

Publisher추가


```
[{"bookId":1,"bookName":"토비의 스프링3","authorName":"김경록","publisherName":"에이콘"},{"bookId":2,"bookName
{"bookId":3,"bookName":"말랑말랑 알고리즘","authorName":"이일민","publisherName":"비제이퍼블릭"},{"bookId":4,"
릭"},{"bookId":5,"bookName":"MySql로 배우는 데이터베이스 개론과 실습","authorName":"박우창","publisherName":"
```


```java
@Entity
@Getter
@AllArgsConstructor
@NoArgsConstructor
public class Publisher {
    @Id
    private Long id;
    public String name;
```

```
    public String address;
}
```

OneToOne 단방향 매핑

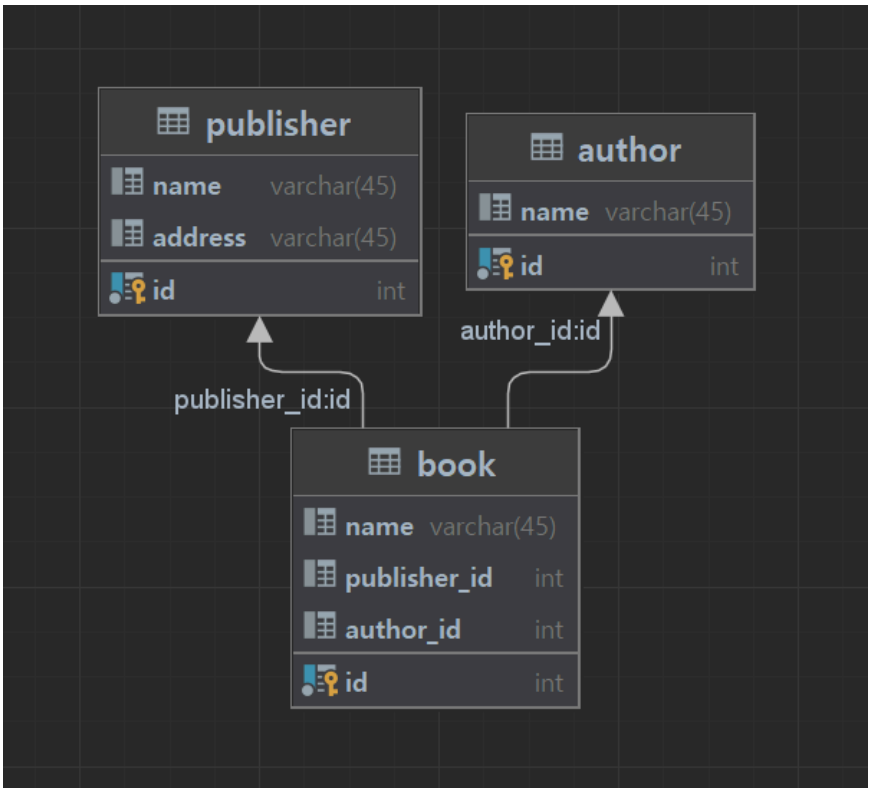
```
@OneToOne
@JoinColumn(name = "publisher_id")
private Publisher publisher;
```

Builder Pattern으로 해놓았기 때문에 수정할게 적습니다.

```
public class BookResponse {
    private Long bookId;
    private String bookName;
    private String authorName;
    private String publisherName;

    public static BookResponse of(Book book) {
        return BookResponse.builder()
                .bookId(book.getId())
                .bookName(book.getName())
                .authorName(book.getAuthor().getName())

.publisherName(book.getPublisher().getName())
                .build();
    }
}
```



3:15까지 실습

실습
요구사항
1. 병원 목록(5개 이상) 이 보이는 API Endpoint
2. 특정 병원의 정보(주소)와 리뷰들(s)을 보여주는 기능

Hospital – Review

Book의 경우 우리는 Book(Many)을 조회 GET /api/v1/books

Hospital의 경우 Hospital(One)을 조회 합니다. GET /api/v1/hospitals
foreign_key가 Review쪽에 있기 때문에 Hospital쪽에서는 Review를 조회할 수 없습니다.

OneToMany, ManyToOne 양방향 매핑

```
public class Hospital {
    @Id
    private Integer id;
    private String roadNameAddress;
    private String hospitalName;
```

```
    @OneToMany(mappedBy = "hospital", fetch = FetchType.LAZY)
    private List<Review> reviews;
```

```
public class Review {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String title;
    private String content;
    private String userName;

    @ManyToOne
    @JoinColumn(name = "hospital_id")
    private Hospital hospital;
```

Hibernate: alter table hospital add column hospital_name varchar(255)
Hibernate: alter table review add constraint FK49momwbwjdvu64cevtop3n4va foreign key (hospital_id) references hospital (id)

GET /api/v1/hospitals
GET /api/v1/hospitals/{id}

id로 조회 했을때 review가 보이면 됩니다.

3:37까지 실습

10주차	1. 댓글, 리뷰 기능 – Many To One 2. 누가 어떤 병원에 방문 했는지 Many To Many 3. 누가 어떤 병원에 방문해서 어떤 진료를 몇번 받았는지 Many To Many 4. 누가 어떤 병원에 방문해서 어떤 진료를 몇번 받았는데 진료비는 얼마인지 Aggregate

14주차	종합프로젝트 1주차 종합프로젝트 14주 부터(이번주가 10주) - 점수 - Docker로 안띄우시면 점수 x - readme.md 채점기준에 포함예정