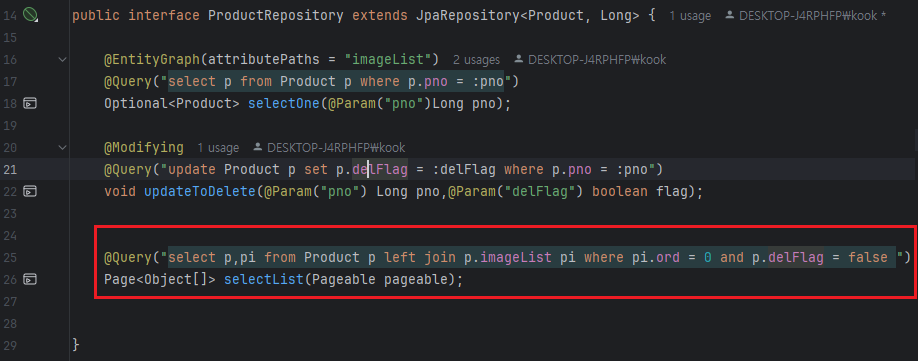
**#목록 가져오기**

**N+1문제**

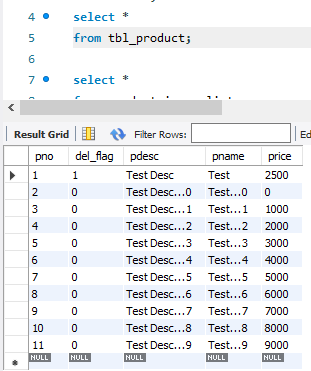
예) 상품 열개 리스트를 가져올 때 리스트를 가져오는 쿼리 뿐 아니라 상품에 대한 쿼리 10개가 추가적으로 날아감

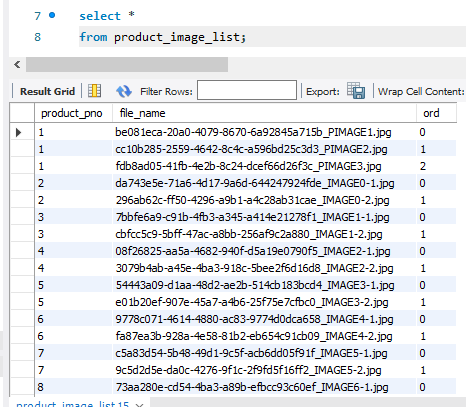
**상품 목록 가져오기**

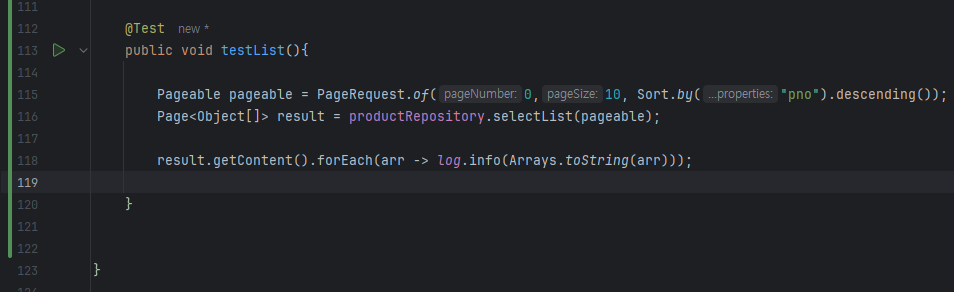
상품과 상품 이미지 하나(0번 이미지) 가져오기

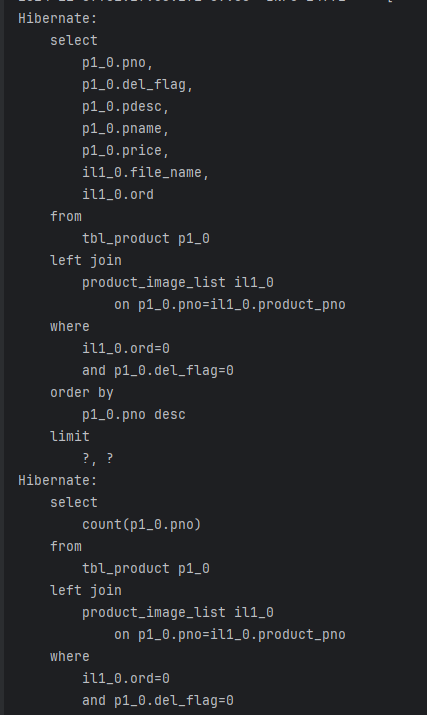


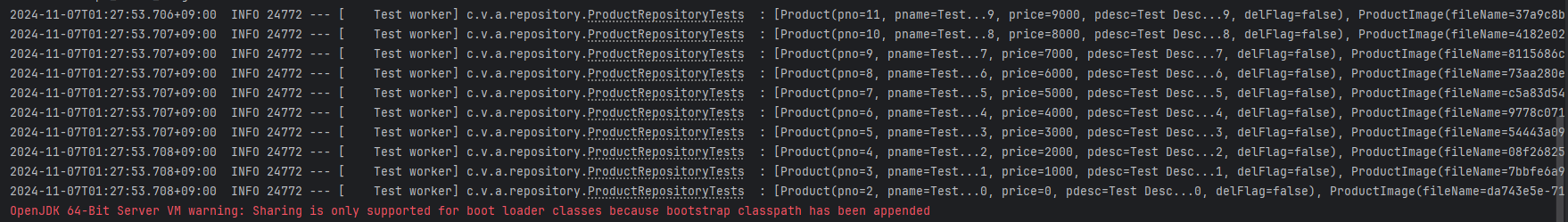


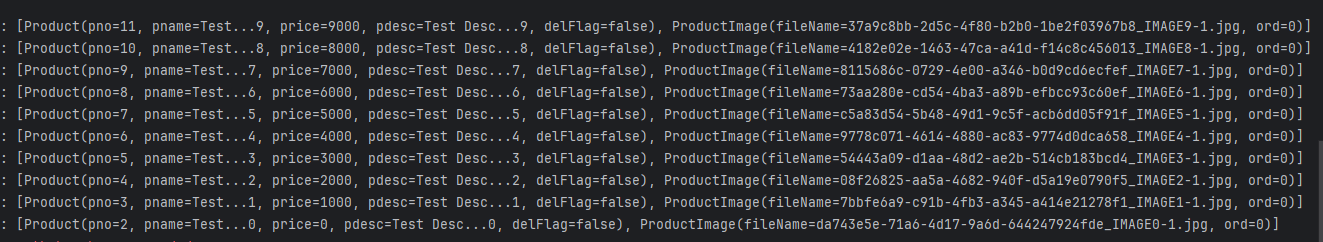








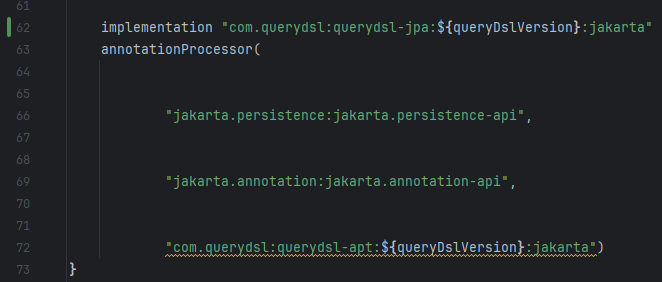


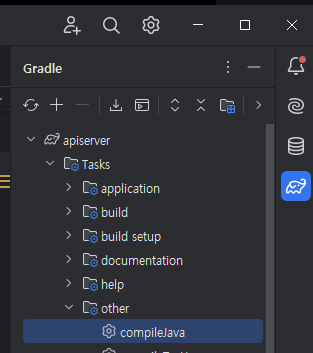


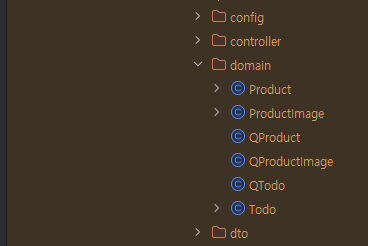
**#ElementCollection을 QueryDSL에서 사용하기**

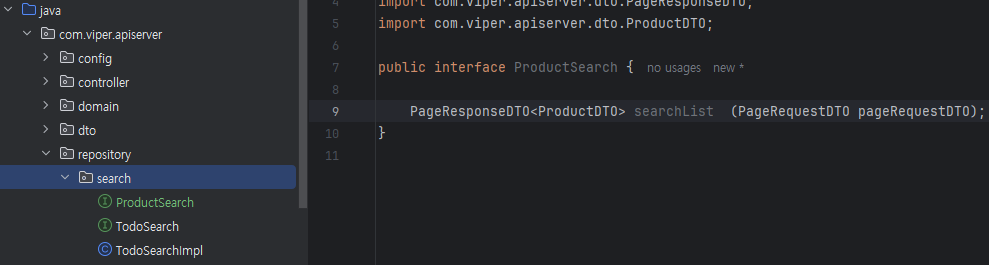
검색이 없는 경우에는 위와 같이 사용하면 되지만 검색이 필요하여 동적 쿼리를 사용해야하는 경우라면 QueryDSL을 사용해야함으로 달라진다.

라이브러리 추가 후 Q도메인 객체를 만들어준다.

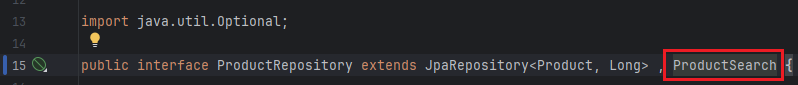




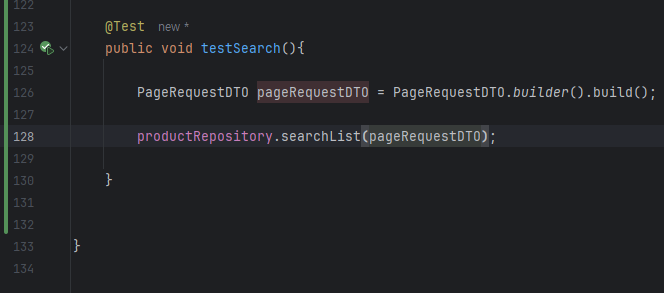


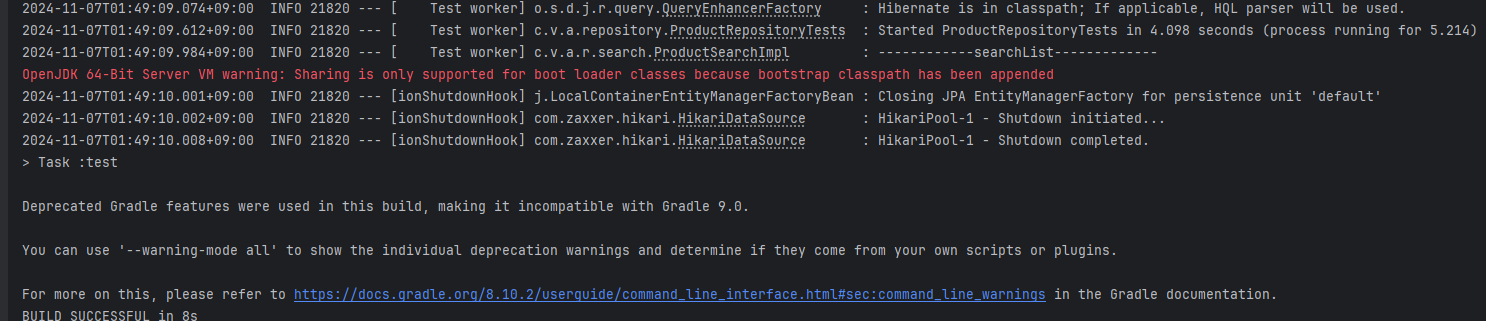




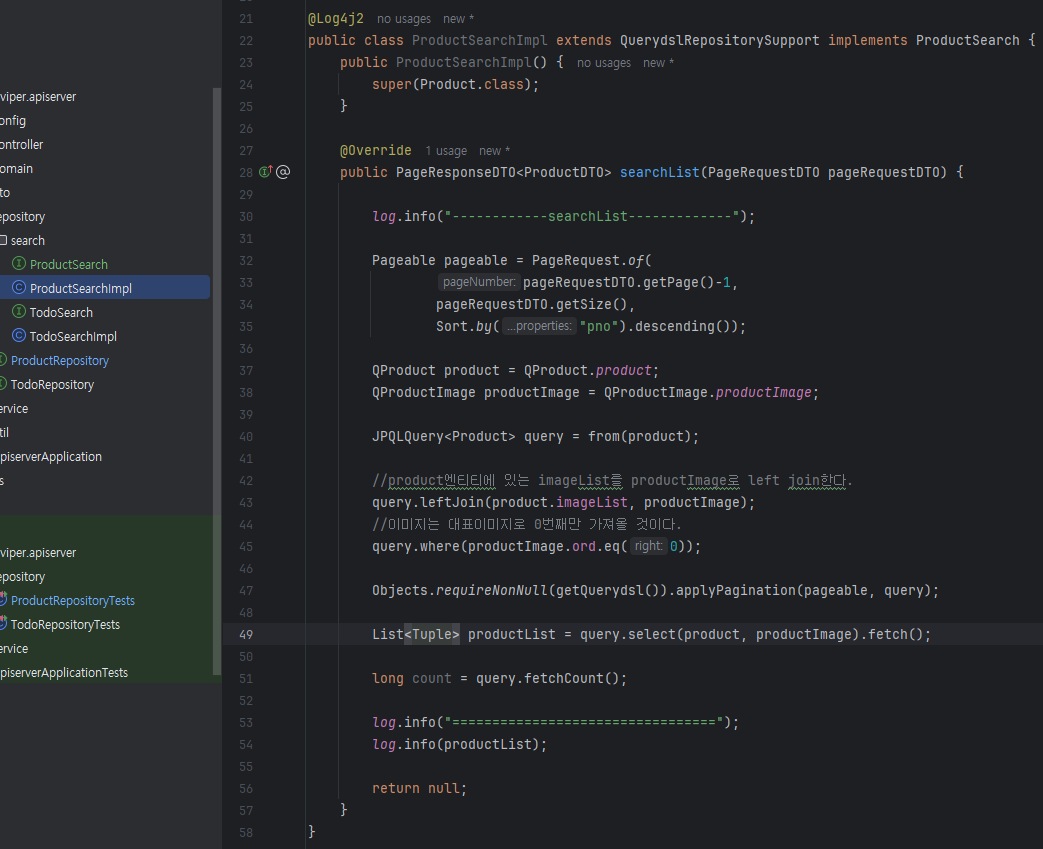


**searchList 로그로 찍어보기**





**searchList 기능 구현하기**



페이징 처리된 결과를 반환 받기 위해 pageable 객체를 구성해주고

쿼리를 날리기 위한 Q도메인 객체 생성 후.

product엔티티에 대해 쿼리를 날릴 준비

elementCollectoin을 이용해서 QueryDSL을 처리하는 방식.

이미지는 0번째만 가져오도록 where절 구성 후 페이칭 처리 반영하고

product, productImage를 프로젝션하여 쿼리를 날림.

프로젝션 결과는 튜플이기 때문에 튜플로 받아온다.

로그로 결과 찍기.

프로젝션 사용 시 결과값은 Tuple 타입으로 받아야한다.

다시 testSearch를 통해 실행 결과를 살펴보면 아래와 같다.

