**#엔티티 처리**

**Many to One (게시글과 댓글의 관계)**

* **게시글과 댓글의 라이프 사이클이 다르다.**

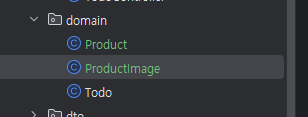
**ElementCollection (상품과 상품이미지)**

* **상품이 상품의 이미지를 갖는다. 별도의 엔티티라고 보기 어렵다.**

**#ElementCollection**

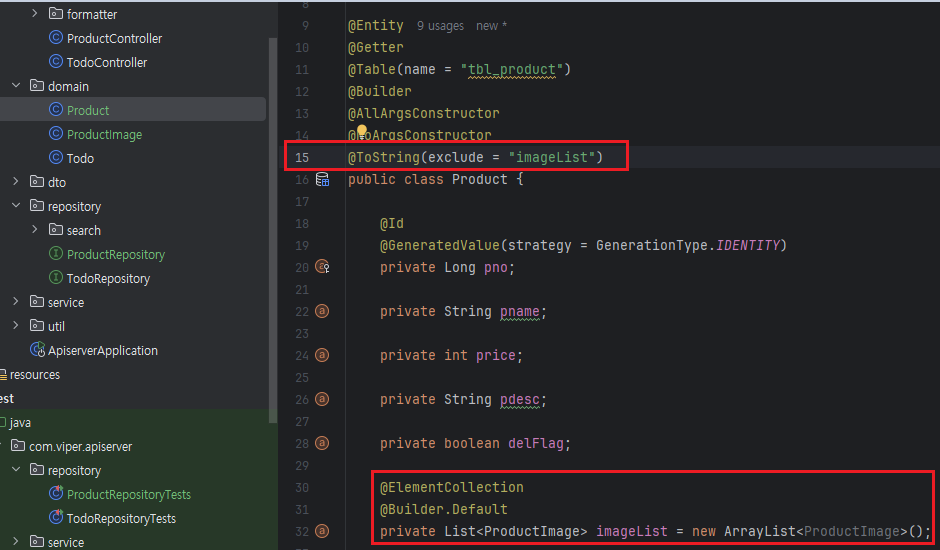
ElementCollection이 속해 있는 엔티티에서 관리하는 것이 좋다.

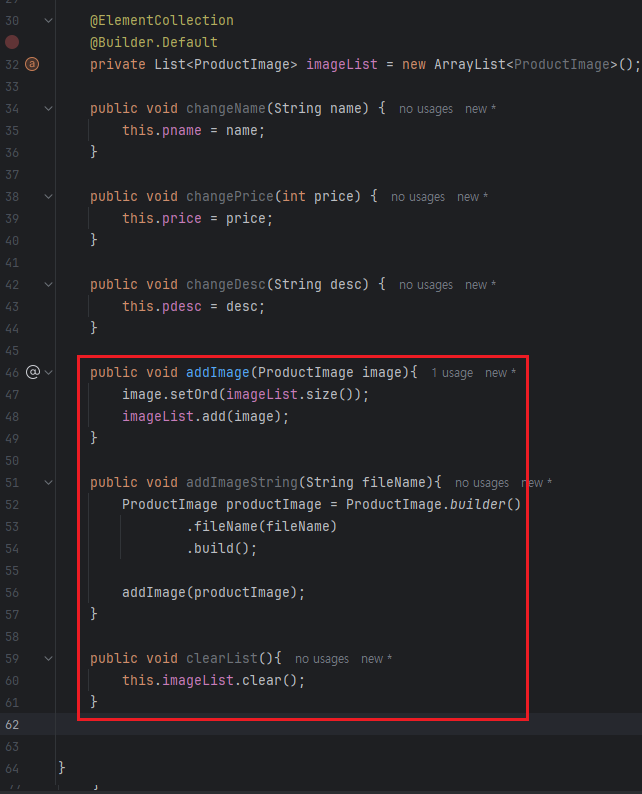
즉, 여기서는 상품에서 상품 이미지까지 같이 관리한다고 생각하면 된다.



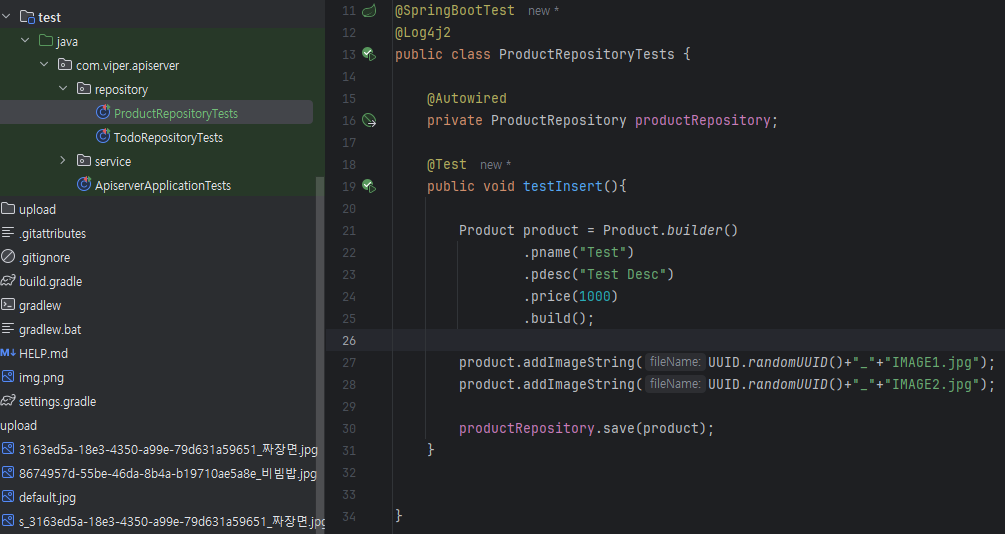


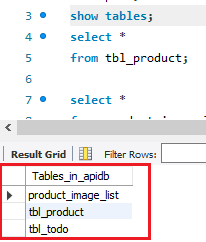
상품 이미지와 관련된 것들을 모두 엔티티에서 관리하도록 메서드 작성

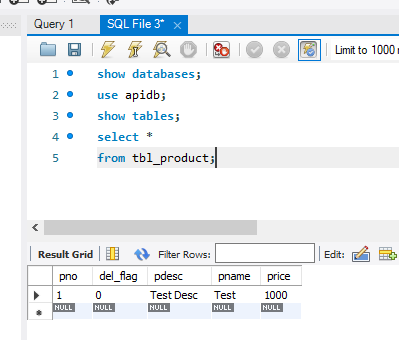


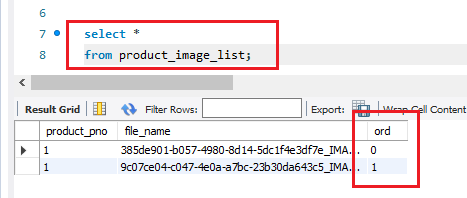


상품 이미지 관리 메서드 테스트 해보기









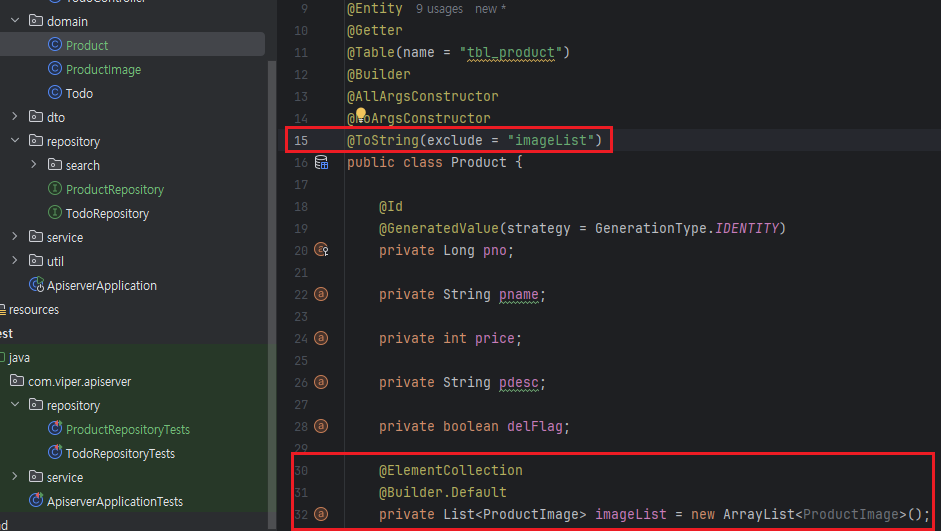
**상품 조회 테스트 해보기**

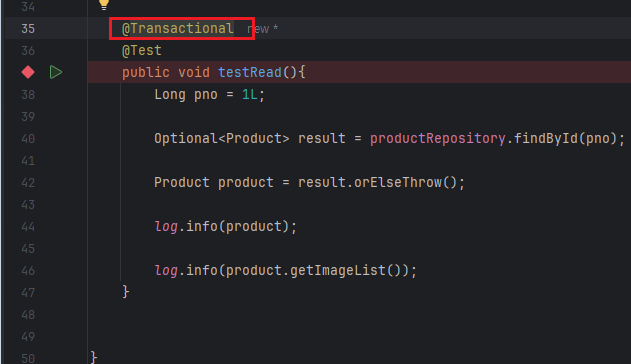
**testRead()**

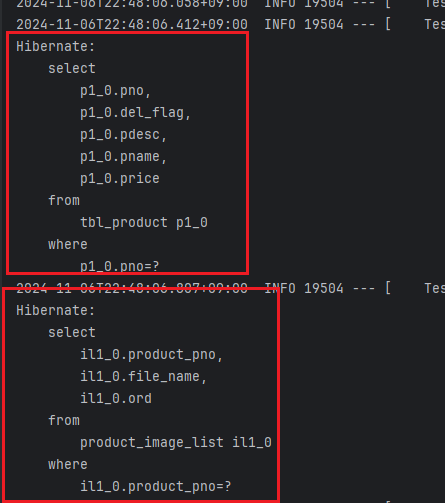
ToString exclude로 imageList를 하지 않으면 두번 조회가 일어난다.

product에서 한번 productimage에서 한번.

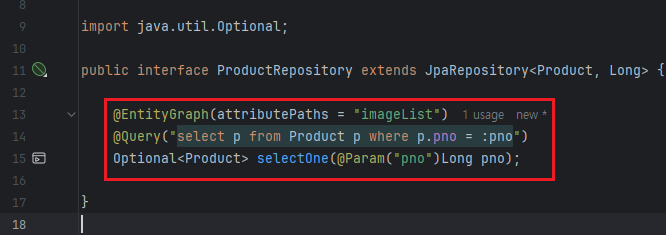
따라서 transactional 어노테이션 안 써주면 에러남





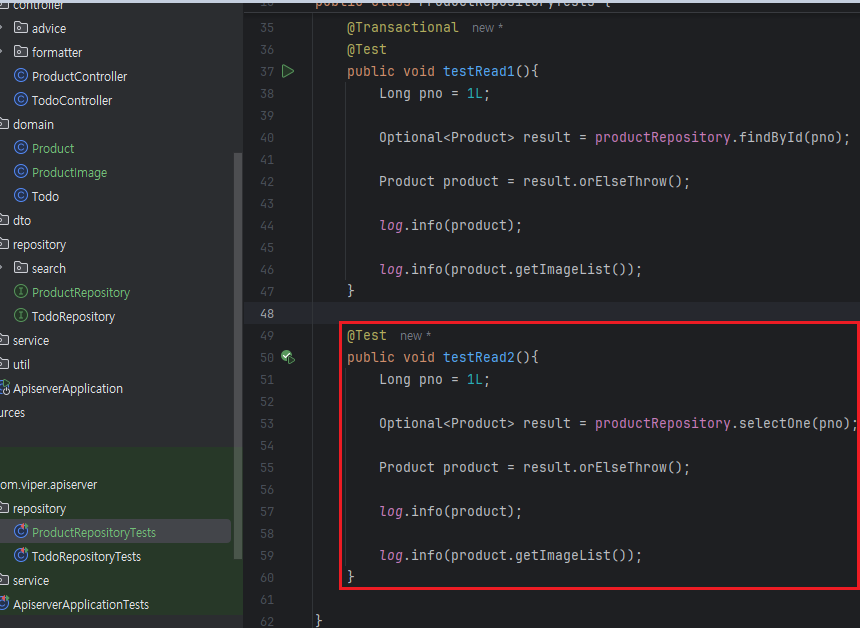


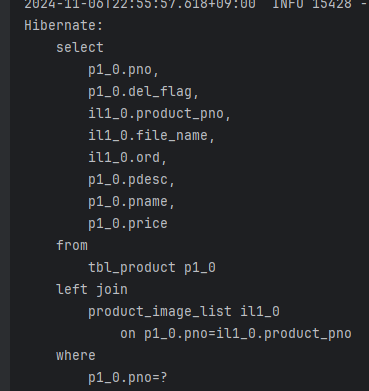
**testRead2()**

****

**@EntityGraph**

**엔터티를 조회할 때 특정 연관 속성을 로딩하는 데 사용**

****



left join을 통해 하나의 쿼리로 조회함.

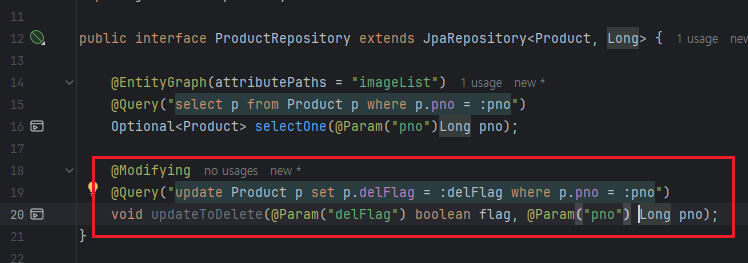
**상품 삭제 테스트 해보기**

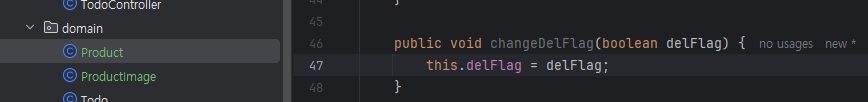
**@Modifying**

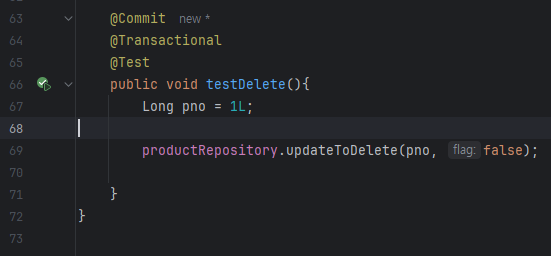
@Query 어노테이션(JPQL Query, Native Query)을 통해 작성된 INSERT, UPDATE, DELETE (SELECT 제외) 쿼리에서 사용되는 어노테이션이다.

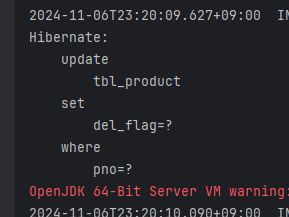
**주로 벌크 연산과 같이 이용**

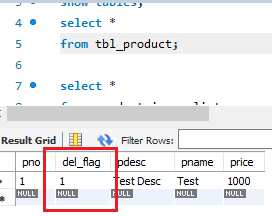
Entity Life-cycle을 무시하고 쿼리가 실행되기 때문에 해당 어노테이션을 사용할 때는 영속성 컨텍스트 관리에 주의해야 한다. (clearAutomatically, flushAutomatically를 통해 간단하게 해결할 수 있다.)







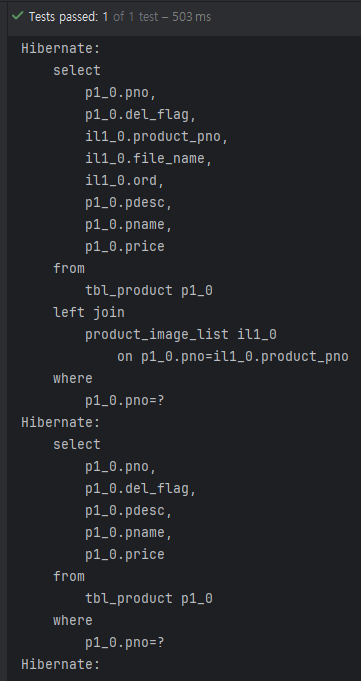


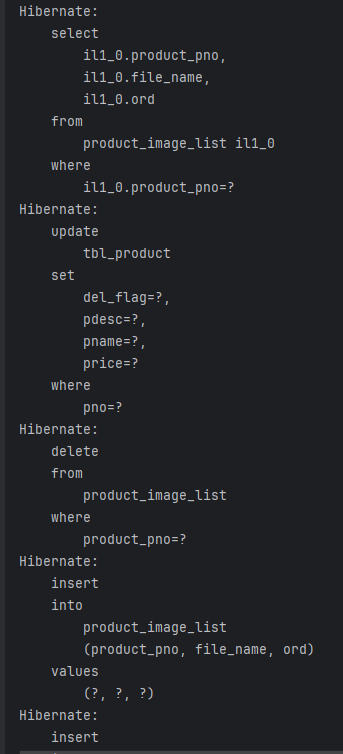


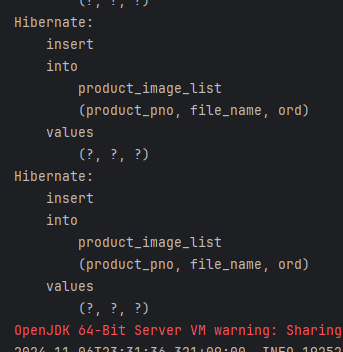
**상품 수정 테스트 해보기**

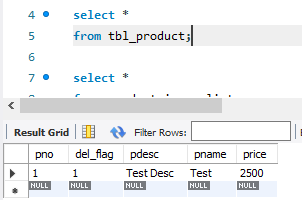
기존 이미지 삭제 후 새로운 이미지를 넣어 저장하도록

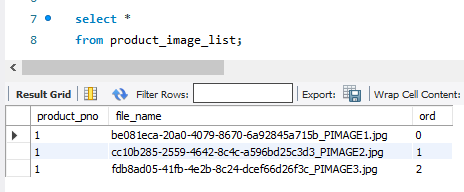












**@ElementCollection 사용의 장점**

* 다뤄하는 엔티티가 줄어든다.
* 한번에 다뤄야하는 것들에 적용하면 편리
* Join역시 한번에 된다.
* 라이프 사이클이 Entity에 맞춰진다.