[JPA] 더티 체킹 (Dirty Checking)

```
트랜잭션 안에서 Entity의 변경이 일어났을 때
변경한 내용을 자동으로 DB에 반영하는 것
```

ORM 구현체 개발 시 더티 체킹이라는 말을 자주 볼 수 있다.

더티 체킹이 어떤 것을 뜻하는 지 간단히 살펴보자.

JPA로 개발하는 경우 구현한 한 가지 기능을 예로 들어보자

ex) 주문 취소 기능

```java
@Transactional
public void cancelOrder(Long orderId) {
    //주문 엔티티 조회
    Order order = orderRepository.findOne(orderId);

    //주문 취소
    order.cancel();
}
```

`orderId`를 통해 주문을 취소하는 메소드다. 데이터베이스에 반영하기 위해선, `update`와 같은 쿼리가 있어야 할 것 같은데 존재하지 않는다.

하지만, 실제로 이 메소드를 실행하면 데이터베이스에 update가 잘 이루어진다.

- 트랜잭션 시작
- `orderId`로 주문 Entity 조회
- 해당 Entity 주문 취소 상태로 **Update**
- 트랜잭션 커밋

이를 가능하게 하는 것이 바로 '더티 체킹(Dirty Checking)'이라고 보면 된다.

그냥 더티 체킹의 단어만 간단히 해석하면 변경 감지로 볼 수 있다. 좀 더 자세히 말하면, Entity에서 변경이 일어난 걸 감지한 뒤, 데이터베이스에 반영시켜준다는 의미다. (변경은 최초 조회 상태가 기준이다)

> Dirty : 상태의 변화가 생김

> Checking : 검사

JPA에서는 트랜잭션이 끝나는 시점에 변화가 있던 모든 엔티티의 객체를 데이터베이스로 알아서 반영을 시켜준다. 즉, 트랜잭션의 마지막 시점에서 다른 점을 발견했을 때 데이터베이스로 update 쿼리를 날려주는 것이다.

- JPA에서 Entity를 조회
- 조회된 상태의 Entity에 대한 스냅샷 생성
- 트랜잭션 커밋 후 해당 스냅샷과 현재 Entity 상태의 다른 점을 체크
- 다른 점들을 update 쿼리로 데이터베이스에 전달

이때 더티 체킹을 검사하는 대상은 영속성 컨텍스트가 관리하는 Entity로만 대상으로 한다.

준영속, 비영속 Entity는 값을 변경할 지라도 데이터베이스에 반영시키지 않는다.

기본적으로 더티 체킹을 실행하면, SQL에서는 변경된 엔티티의 모든 내용을 update 쿼리로 만들어 전달하는데, 이때 필드가 많아지면 전체 필드를 update하는게 비효율적일 수도 있다.

이때는 `@DynamicUpdate`를 해당 Entity에 선언하여 변경 필드만 반영시키도록 만들어줄 수 있다.

```
@Getter
@NoArgsConstructor
@Entity
@DynamicUpdate
public class Order {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String product;
```

[참고 자료]

- 링크
- 링크