컬렉션 페이징

문제 상황

- 주문을 기준으로 페이징을 해야 하는데 주문 아이템을 기준으로 페이징이 되고 있는 상황
- 주문 아이템이 N에 해당하기 때문에 주문이 N만큼 뻥튀기가 되고 있다.
- 주문은 한 번 했을 뿐인데 주문 아이템이 `2`개라는 이유로 `orderId 1`번이 `2`번 출력 되고 있다.

```
{
    "code": 1,
    "msg": "주문 이력 조회",
    "data": {
        "userId": 2,
        "username": "griotold",
        "orderItemDtos": {
            "content": [
                {
                    "orderId": 1,
                    "itemId": 3,
                    "itemName": "물티슈",
                    "count": 2,
                    "totalPrice": 20000
                },
                {
                    "orderId": 1,
                    "itemId": 2,
                    "itemName": "안경닦이",
                    "count": 2,
                    "totalPrice": 20000
                }
            ],
            "pageable": {
                "sort": {
                    "empty": true,
                    "unsorted": true,
                    "sorted": false
```


컬렉션 조회할 때 원칙 3가지

1. ToOne 관계는 모두 fetch join한다

- `ToOne` 연관 관계를 `fetch join`하는 경우 레코드 수를 늘리지 않고
- 단순히 데이터의 정보만 늘려주기 때문에 페이징에 영향을 주지 않는다.
- 따라서 `Order`와 `ToOne` 관계인 `User`는 `fetch join`으로 조회한다.

```
package com.griotold.bankshop.domain.order;

import com.griotold.bankshop.support.Querydsl4RepositorySupport;
import com.querydsl.core.types.dsl.BooleanExpression;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Repository;

import static com.griotold.bankshop.domain.order.QOrder.*;
import static com.griotold.bankshop.domain.orderItem.QOrderItem.orderItem;
import static com.griotold.bankshop.domain.user.QUser.user;

@Repository
public class OrderQueryRepository extends Querydsl4RepositorySupport {
    public OrderQueryRepository() {
        super(Order.class);
    }

    public Page<Order> findOrderHistory(String orderStatus, Pageable pageable) {
        return applyPagination5(pageable, query ->
                query.selectFrom(order)
                        .innerJoin(order.user, user).fetchJoin()
                        .where(statusCheck(orderStatus)),
                countQuery -> countQuery.select(order.count())
                        .from(order)
                        .where(statusCheck(orderStatus)));
    }

    private BooleanExpression statusCheck(String orderStatus) {

        if (OrderStatus.valueOf(orderStatus) == OrderStatus.ORDER) {
            return order.orderStatus.eq(OrderStatus.ORDER);
        } else if (OrderStatus.valueOf(orderStatus) == OrderStatus.CANCEL) {
            return order.orderStatus.eq(OrderStatus.CANCEL);
        } else {
            return null;
        }

    }
}
```

2. 컬렉션은 지연 로딩 한다.

- 애초에 지연로딩으로 세팅해 놓았기 때문에 건드릴 것은 없다.
- 문제는 `N + 1` 인데
- `Order`의 `List<OrderItem>`들을 하나 하나씩 가져올 때마다 `SELECT` 쿼리가 날라가게 된다.
- 이 문제를 해결하기 위한 방법이 다음 단계다.

3. hibernate.default.batch_fetch_size

- `application.yml`에 해당 설정을 지정하면
- 지연 로딩시 `in` 쿼리로 한꺼번에 조회해온다.
- List값을 하나하나씩 `SELECT` 하지 않고 한꺼번에 가져온다는 의미다.

```
spring:

  jpa:

    hibernate:

    properties:
      hibernate.default_batch_fetch_size: 100
```

- `100`을 지정했으므로 `in` 쿼리로 `100`개를 조회해온다.

```
select
        item0_.item_id as item_id1_4_0_,
        item0_.created_at as created_2_4_0_,
        item0_.item_detail as item_det3_4_0_,
        item0_.item_name as item_nam4_4_0_,
        item0_.item_sell_status as item_sel5_4_0_,
        item0_.price as price6_4_0_,
        item0_.stock_number as stock_nu7_4_0_,
        item0_.updated_at as updated_8_4_0_
    from
        item_tb item0_
    where
        item0_.item_id in (
            ?, ?
        )
```

리팩토링 후

```
{
    "code": 1,
    "msg": "주문 이력 조회 버젼2",
    "data": {
        "userId": 2,
        "username": "griotold",
        "orderDtoList": {
            "content": [
                {
                    "orderId": 1,
                    "orderTotalPrice": 60000,
                    "orderItemDtoList": [
                        {
                            "itemId": 1,
                            "itemName": "츄르",
                            "count": 2,
                            "totalPrice": 20000
                        },
                        {
                            "itemId": 2,
                            "itemName": "안경닦이",
                            "count": 2,
                            "totalPrice": 20000
                        },
                        {
                            "itemId": 3,
                            "itemName": "물티슈",
                            "count": 2,
                            "totalPrice": 20000
                        }
                    ]
                }
            ],
```


- 주문을 기준으로 컬렉션 페이징 쿼리가 정상 작동하고 있다!