[구현] Pub/Sub - Compensation and Correlation

주문 취소 이벤트를 발행하고, 수신하시오.

Instruction

Compensation and Correlation

이전 랩에서 주문을 생성하는 **OrderPlaced** 라는 이벤트를 발행하였다.
이번 랩에서는 주문서비스에서 주문을 취소하는 **OrderCancelled** 라는 이벤트를 발행 하고,
배송 서비스에서는 **OrderCancelled** 이벤트를 수신하여 **DeliveryCancelled** 라는 이벤트를 발행한다.

작업순서

주문서비스에서 주문을 취소하는 작업

Aggregate 에서 이벤트 발행

- **Ctrl + p 로 Order.java 리소스 찾기**
- **Order.java 27 라인에 아래 로직 추가 후 저장**

```
@PreRemove
public void onPreRemove(){
    OrderCancelled orderCancelled = new OrderCancelled();
    BeanUtils.copyProperties(this, orderCancelled);
    orderCancelled.publishAfterCommit();
}
```

OrderCancelled 도메인 이벤트 클래스 생성

- **shopmall 패키지에서 마우스 오른쪽 메뉴에서 'New File' 선택**
- **OrderCancelled.java 파일을 생성 한 후, OrderPlaced.java 파일 내용을 모두 복사한다.**
- **public class OrderCancelled 로 클래스명을 변경하고 저장한다.**

주문삭제 커맨드를 실행하여 **OrderCancelled** 이벤트 발행 확인

- **주문 서비스 및 카프카 Consumer 실행**

```
mvn spring-boot:run
/usr/local/kafka/bin/kafka-console-consumer.sh --bootstrap-server
localhost:9092 --topic shopmall --from-beginning
```

- **주문 생성 후, 삭제**

```
http localhost:8081/orders productId=1 productName="TV" qty=3
```

```
http DELETE localhost:8081/orders/1
```

배송서비스에서 주문 삭제시 배송을 취소하는 작업

배송서비스의 **PolicyHandler** 코드 수정

- **PolicyHandler.java 29** 라인에 아래 이벤트 수신 시, 실행할 로직을 추가한다.
- **List** 클래스에 대한 임포트를 추가해 준다.**(import java.util.List;)**

```
@StreamListener(KafkaProcessor.INPUT)
public void wheneverOrderCancelled_DeleteDelivery(@Payload
OrderCancelled orderCancelled){
    if(orderCancelled.isMe()){
        List<Delivery> deliveryList =
deliveryRepository.findByOrderId(orderCancelled.getId());
        if ((deliveryList != null) && !deliveryList.isEmpty()){
            deliveryRepository.deleteAll(deliveryList);
        }
    }
}
```

OrderCancelled 도메인 이벤트 클래스 복제

- **order** 서비스의 **OrderCancelled.java** 파일을 복사하여 **delivery** 서비스의 **shopmall** 패키지에 붙여넣는다.

주문번호에 해당하는 배송을 취소하기 위해서는 주문 **ID** 에 맞는 배송 **Entity** 를 찾아야 한다.

- **DeliveryRepository.java** 파일에서 아래와 같이 주문 아이디별 배송을 찾는 로직을 추가**(7** 라인**)**한다.

```
List<Delivery> findByOrderId(Long orderId);
```

- **(import** 구문이 필요하다. **import java.util.List;)**

Delivery Aggregate 에서 삭제시 이벤트 발행

- **Delivery.java** 에 아래 로직 입력**(25** 라인**)**

```
@PreRemove
public void onPreRemove(){
    DeliveryCancelled deliveryCancelled = new DeliveryCancelled();
    BeanUtils.copyProperties(this, deliveryCancelled);
    deliveryCancelled.publishAfterCommit();
}
```

DeliveryCancelled 이벤트 생성

- **DeliveryCancelled.java** 파일을 생성 한 후, **DeliveryStarted.java** 파일의 내용을 복사하여 붙여 넣는다.
- **public class DeliveryCancelled** 로 클래스명과 생성자 이름을 변경한다.

확인

- 파일 저장여부를 확인하고 주문과 배송서비스 모두 재기동한다.
- 새로운 주문과 주문삭제 커맨드로 DeliveryCancelled 이벤트 발행 확인

```
http localhost:8081/orders productId=1 productName="TV" qty=3
http localhost:8081/orders
http localhost:8082/deliveries
http DELETE localhost:8081/orders/1
```

- kafka Consumer 에서 이벤트 확인

Checkpoints 확인

- 1 번과 2 번작업을 모두 정상적으로 실행한다.
- IDE 상단의 메뉴에서 labs > 결과제출 버튼을 클릭한다.

Service Clear

- 다음 Lab 을 위해 기동된 모든 서비스 종료

```
fuser -k 8081/tcp
fuser -k 8082/tcp
```

상세설명

CheckPoints

1. OrderCancelled 이벤트 발행 ☐

2. DeliveryCancelled 이벤트 발행 ☐