안전하게 메시지 Consume 하기 위한 설정

우선 `auto-ack` 를 false로 설정해주어야한다.

`auto-ack` 가 true로 설정되면 다음과 같이 동작한다.

1. queue로 메시지 publish
2. consumer로 메시지 전달
3. consumer는 queue에게 ack 리턴
4. queue에서 메시지 삭제

이때 실제로 consumer에서 메시지를 처리하는 시간이 더 길기 때문에 queue에서의 메시지가 먼저 삭제된다. 따라서 consumer에서 문제가 생겼을 경우 메시지 유실이 발생하게된다.

그리고 메시지 전달에 실패했을 경우는 DLQ(Dead Letter Queue)로 가도록 해야한다.

코드 적용

Spring AMQP에서는 간단하게 `acknowledge-mode` 속성 값을 `manual` 로 설정해주면 된다. 기본 값은 `auto` 이며 `none` 값으로 설정해주면 `auto-ack=true` 와 동일하다.

```
spring:
  config.activate.on-profile: amqp
  rabbitmq:
    listener:
      simple:
        acknowledge-mode: manual # default: auto
```

위처럼 설정해준 후 아래처럼 작업이 완료된 후 ack를 직접 전송해주도록 하여 메시지 유실을 방지할 수 있다.

```
@RabbitListener(queues = "#{testQueue.name}")
public void receive(
        Channel channel, @Header(AmqpHeaders.DELIVERY_TAG) long tag,
@Payload AlarmPayload alarmPayload) {
    Alarm alarm = saveAlarm(alarmPayload);
    receive(alarmPayload.memberId(), AlarmResponse.fromEntity(alarm));
    // 직접 ack 전송
    channel.basicAck(tag, false);
}
```

Delivery Tag

delivery tag는 RabbitMQ가 특정 채널을 통해 Consumer에게 전달한 각 메시지에 대해 고유하게 할당하는 숫자이다. 같은 연결 내의 다른 채널에서는 동일한 delivery tag가 다른 메시지를 가리킬 수 있다.

예를 들어, 채널 A와 채널 B가 있을 경우 채널 A의 delivery tag 1이 할당된 메시지가 있고 채널 B에서도 delivery tag 1이 할당된 메시지가 있을 수 있다. 따라서 채널 A와 채널 B에서 같은 delivery tag를 서로 다른 메시지에 사용될 수 있지만, 각 채널 내에서 고유하기 때문에 충돌하지 않는다.

multiple flag

`basicReject()` 또는 `basicAck()` 메소드의 두 번째 인자는 `multiple` flag를 의미한다. `multiple` flag를 `true`로 설정하면 현재 채널을 통해 전달되는 메시지들을 모두 `reject` 하거나 `ack`하는 것이다. `false`로 설정하면 지정한 `delivery tag`에 해당하는 메시지만 적용된다.

[reference]

- https://teragoon.wordpress.com/2012/01/26/message-durability%EB%A9%94%EC%8B%9C%EC%A7%80-%EC%9E%83%EC%96%B4%EB%B2%84%EB%A6%AC%EC%A7%80-%EC%95%8A%EA%B8%B0-autoackfalse-1/
- https://docs.spring.io/spring-amqp/reference/amqp/template.html