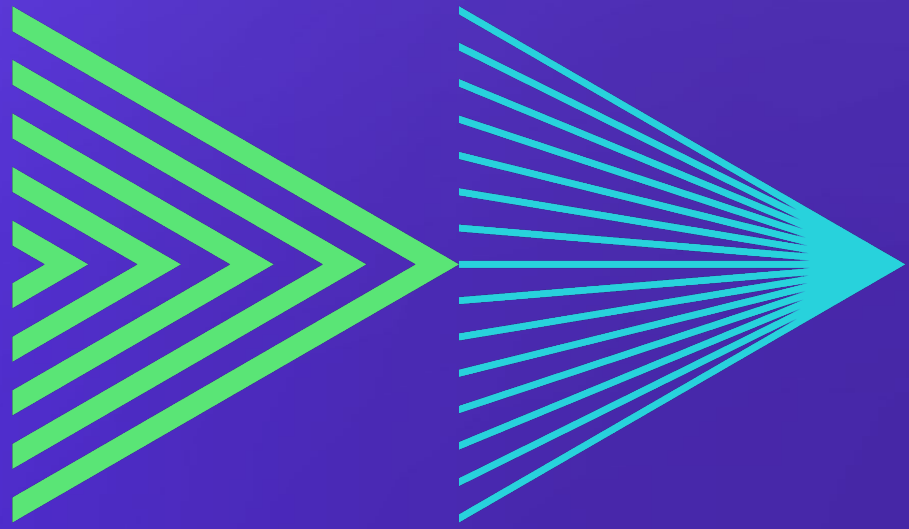
NHN FORWARD

분산 시스템에서 데이터를 전달하는 효율적인 방법

NHN Dooray 협업서비스개발그린팀
김병부



다룰 내용

- 데이터 전달 보장 방법론
- RDB를 사용하는 애플리케이션에서 전달 방법
- RabbitMQ를 사용한 전달 방법
- Kafka를 사용하는 애플리케이션의 전달 방법

분산 시스템에서 데이터를 전달하는 효율적인 방법

분산 시스템이란

분산 시스템이란?

- 목표를 달성하기 위하여 여러 개의 컴퓨터 리소스를 사용하는 시스템
- 시스템은 두 개 이상의 컴포넌트로 구성되어있다.
  - 엔터프라이즈 애플리케이션
  - 마이크로 서비스 아키텍처 애플리케이션
  - 모놀리식 아키텍처 애플리케이션 + 검색엔진
- 네트워크를 사용하여 컴포넌트 간의 기능을 통합

분산 시스템에서 데이터를 전달하는 효율적인 방법

데이터 전달

NHN FORWARD

데이터를 전달하는 방법

- Remote API
- MessageQueue

데이터 전달

Remote API를 사용한 데이터 전달

- 서버-클라이언트 구조
  - 서버에 데이터를 조작(C/U/D)
  - 서버의 데이터를 조회(R)
- 사용자 요청에 즉각 응답하는 API에서 주로 사용하는 방식
- 비교적 간단한 개발

데이터 전달

MessageQueue를 사용한 데이터 전파

- Publisher – Consumer 구조
  - Consumer가 데이터를 조작(C/R/U/D)
  - MessageQueue
- 배치 작업, 비동기 작업에서 주로 사용
- 비교적 복잡한 개발

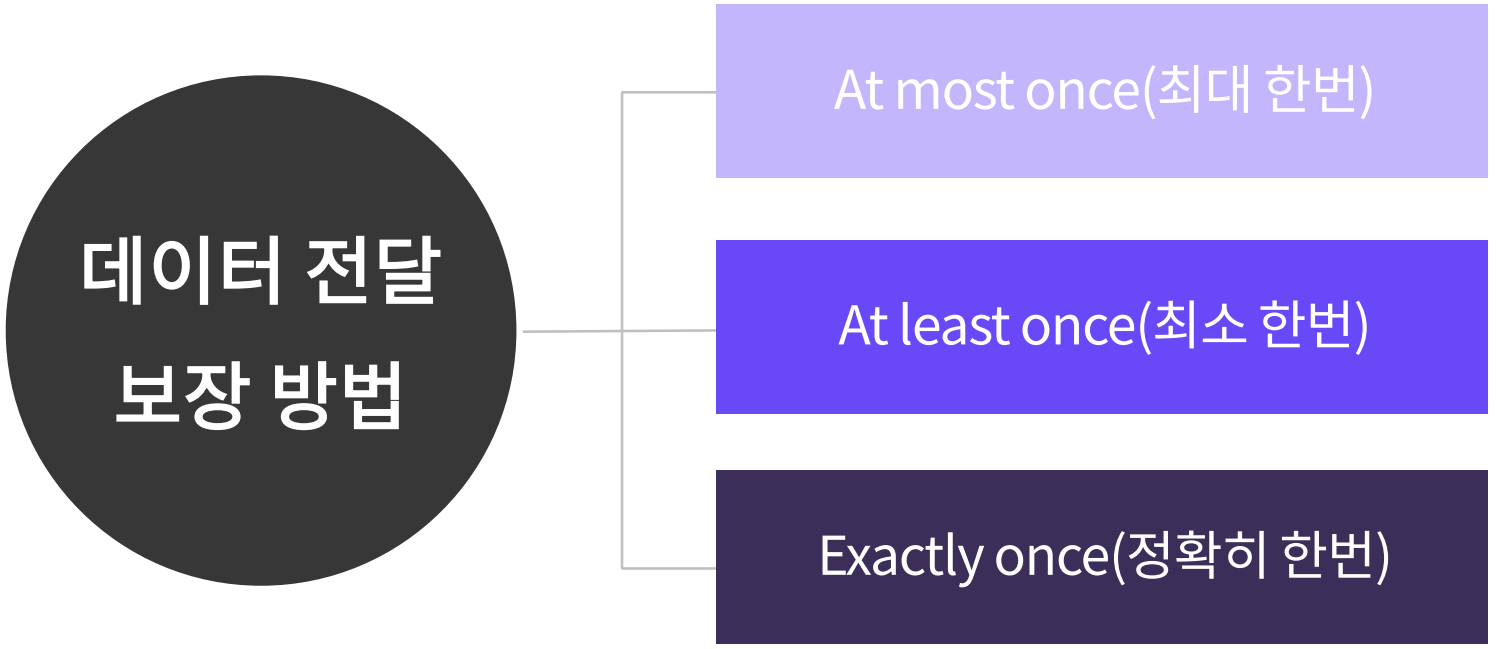
분산 시스템에서 데이터를 전달하는 효율적인 방법

효율적인 방법

분산 시스템에서 컴포넌트들은 네트워크로 연결

- 네트워크는 시스템을 연결하는 유일한 수단
- 하지만 네트워크는 신뢰할 수 없는 매체(Media)
  - 패킷 손실(Packet loss)
  - 네트워크 지연(Latency)
  - 네트워크 다운(Network down)
- 항상 데이터 유실에 대비

효율적인 방법


데이터 전달
보장 방법
At most once(최대 한번)
At least once(최소 한번)
Exactly once(정확히 한번)

효율적인 방법

At-most once delivery(최대 한번 전달)

- Producer는 메시지를 최대 한번만 전송
  - Fire and forget
- Consumer는 메시지를 최대 한번만 수신
  - 네트워크 유실
  - Producer 애플리케이션에서 발생한 예외
  - Consumer 애플리케이션에서 발생한 예외
- 장점
  - 간단한 구조
  - 간단한 개발
- 단점
  - 메시지 유실

효율적인 방법

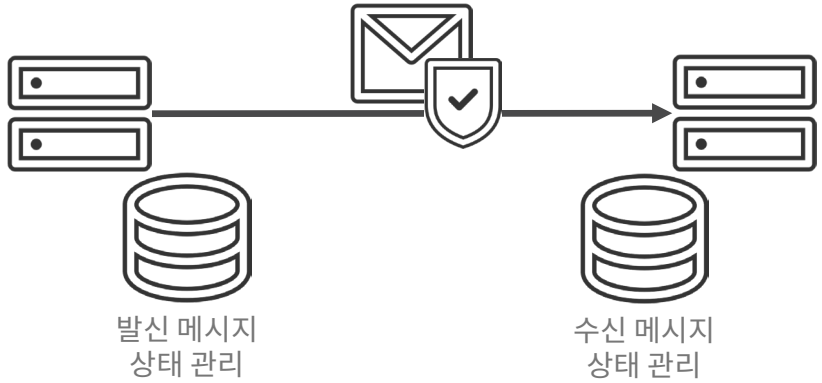
At-least-once delivery(최소 한번 전달)

- Producer는 메시지를 최소 한번 이상 발송
  - 발송 메시지 상태 관리
- Consumer는 메시지를 최소 한번 이상 수신
  - ACK 유실로 인한 재 발송
- 장점
  - Producer는 메시지 발송 보장
  - 효과 대비 쉬운 개발
- 단점
  - 멱등성(idempotent)을 보장해야 하는 Consumer

효율적인 방법

Exactly-once-delivery(정확히 한번 전달)

- 메시지는 정확하게 한번만 전송한다.
- 장점
  - 누락과 중복이 없음
- 단점
  - 가장 어려운 개발 난이도
  - Producer, Consumer에서 모든 상태 관리
  - MessageQueue 기능에 의존한 개발
  - MessageQueue 추가로 인한 시스템 복잡도 증가



여러분의 애플리케이션은

데이터를 어떻게 전달하나요?

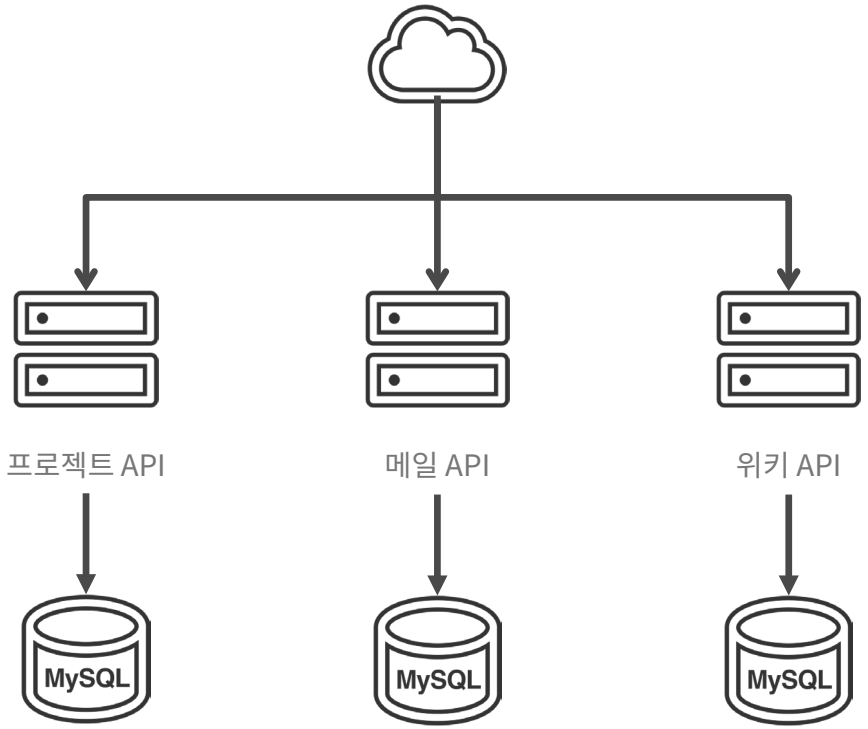
최소 한번은 전달 하나요?

RDB를 사용하는 애플리케이션에서 전달 방법

RDB를 사용하는 애플리케이션에서 전달 방법

서비스별 데이터베이스 패턴(Database per Service Pattern)

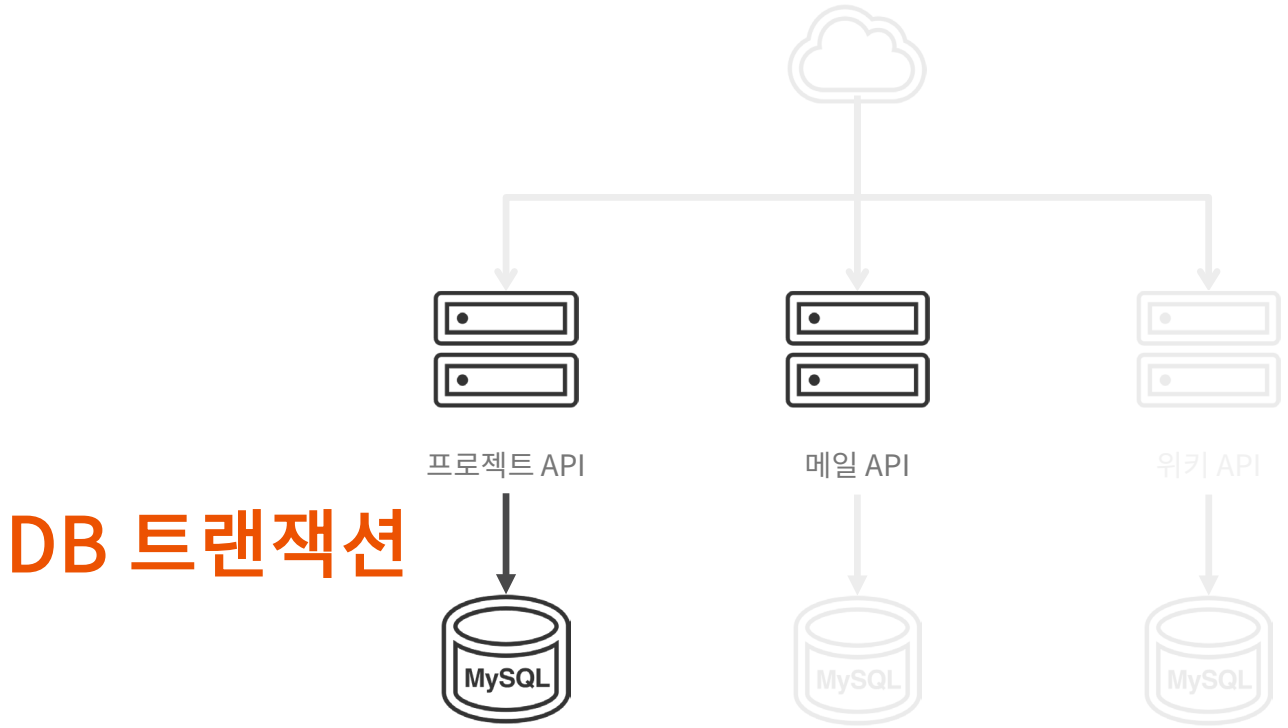
- 마이크로 서비스 아키텍처 패턴
- 모던 애플리케이션의 일반적인 형태



RDB를 사용하는 애플리케이션에서 전달 방법

서비스별 데이터베이스 패턴(Database per Service Pattern)

- 마이크로 서비스 아키텍처 패턴
- 모던 애플리케이션의 일반적인 형태

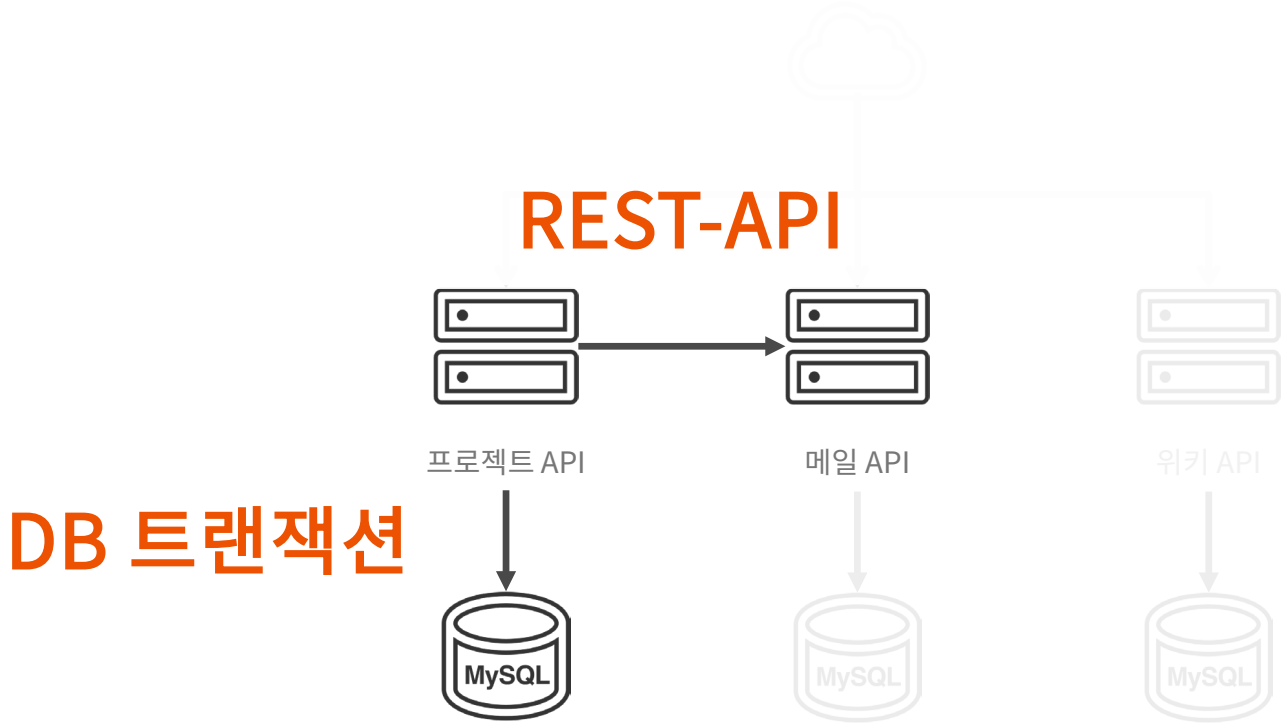


RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

서비스별 데이터베이스 패턴(Database per Service Pattern)

- 마이크로 서비스 아키텍처 패턴
- 모던 애플리케이션의 일반적인 형태



RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

DB 트랜잭션과 REST-API 코드 예제

```
@Service
public class CreateTaskService implements CreateTaskUserCase {

    @Transactional
    public CreateTaskResponse createTask(CreateTaskCommand createTaskCommand) {

        Task task = createTaskCommand.toTask();
        taskRepository.save(task);
        eventHandler.propagate(CreateTaskEvent.of(task));
        return CreateTaskResponse.of(task);
    }
}
```

RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

DB 트랜잭션과 REST-API 코드 예제

```
@Service
public class CreateTaskService implements CreateTaskUserCase {

    @Transactional
    public CreateTaskResponse createTask(CreateTaskCommand createTaskCommand) {

        Task task = createTaskCommand.toTask();
        taskRepository.save(task);  /** Save task entity **/
        eventHandler.propagate(CreateTaskEvent.of(task));
        return CreateTaskResponse.of(task);
    }
}
```

RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

DB 트랜잭션과 REST-API 코드 예제

```
@Service
public class CreateTaskService implements CreateTaskUserCase {

    @Transactional
    public CreateTaskResponse createTask(CreateTaskCommand createTaskCommand) {

        Task task = createTaskCommand.toTask();
        taskRepository.save(task);
        eventHandler.propagate(CreateTaskEvent.of(task)); /** REST-API **/
        return CreateTaskResponse.of(task);
    }
}
```

RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

DB 트랜잭션과 REST-API 코드 예제

```
@Service
public class CreateTaskService implements CreateTaskUserCase {

    @Transactional /** DB Transaction **/
    public CreateTaskResponse createTask(CreateTaskCommand createTaskCommand) {

        Task task = createTaskCommand.toTask();
        taskRepository.save(task);
        eventHandler.propagate(CreateTaskEvent.of(task));
        return CreateTaskResponse.of(task);
    }
}
```

RDB를 사용하는 애플리케이션에서 전달 방법

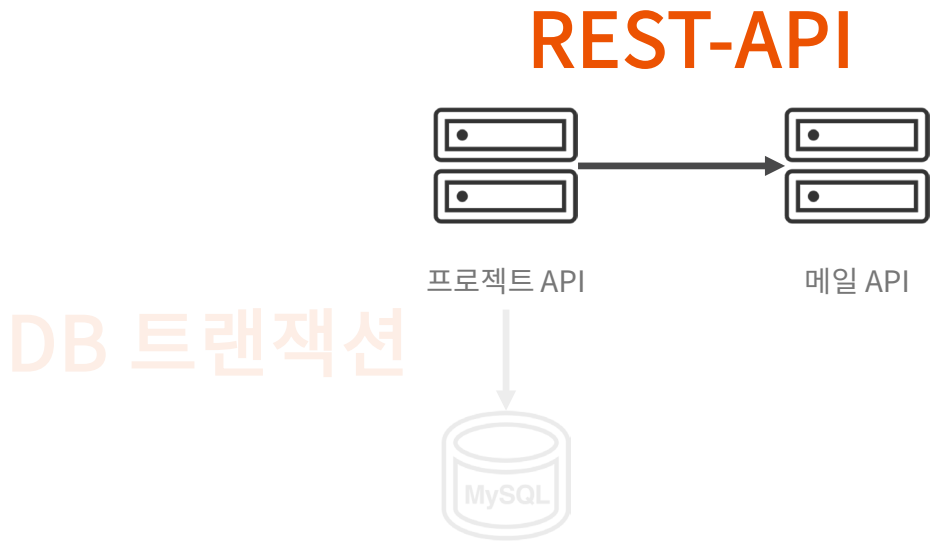
NHN FORWARD

@Transactional

- 스프링 프레임워크에서 제공하는 애너테이션
- AOP를 사용하여 Proxy 객체 생성
  - Target 객체에 추가적인 코드 삽입
  - o.s.transaction.aspectj.AnnotationTransactionAspect.java
- 실행 순서
  - 데이터 저장
  - 이벤트 전달
  - 트랜잭션 커밋

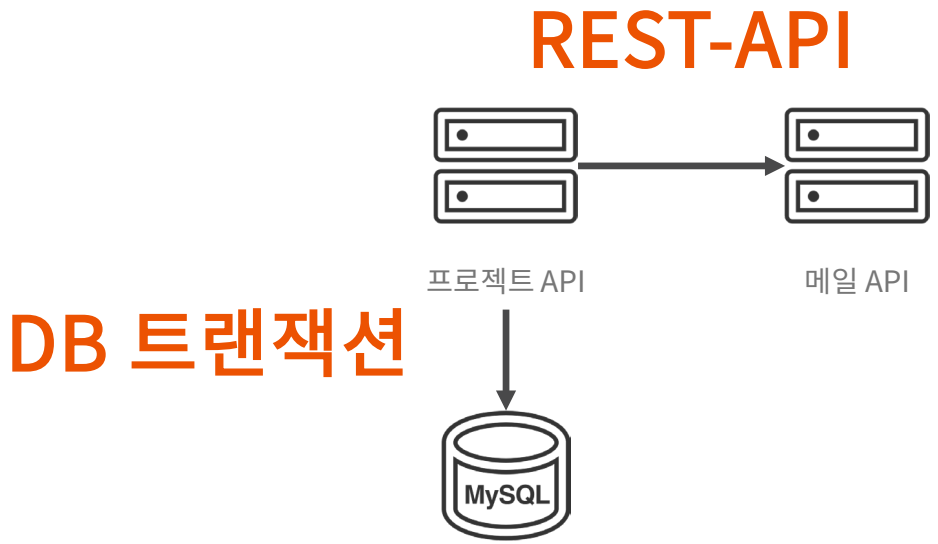
RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

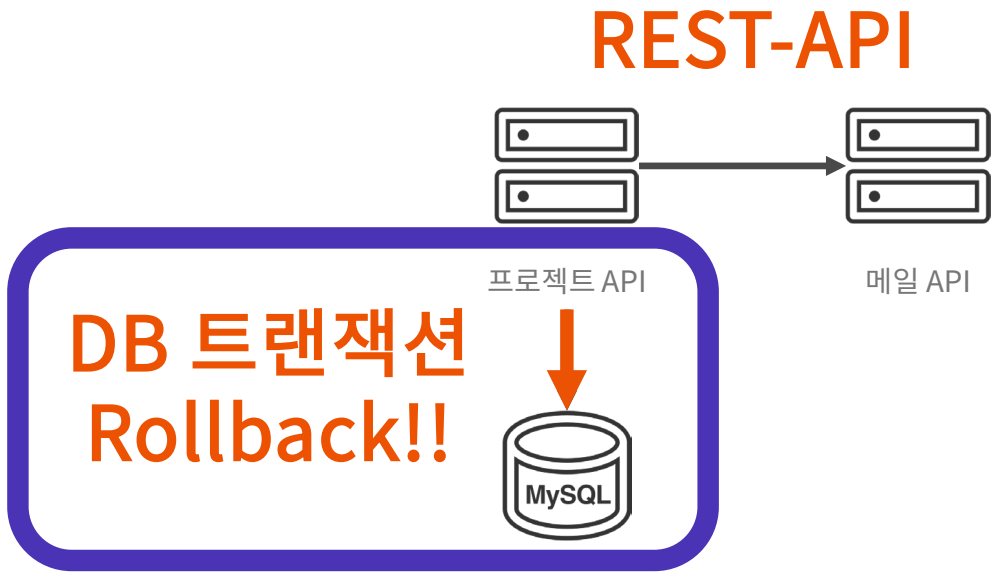


RDB를 사용하는 애플리케이션에서 전달 방법

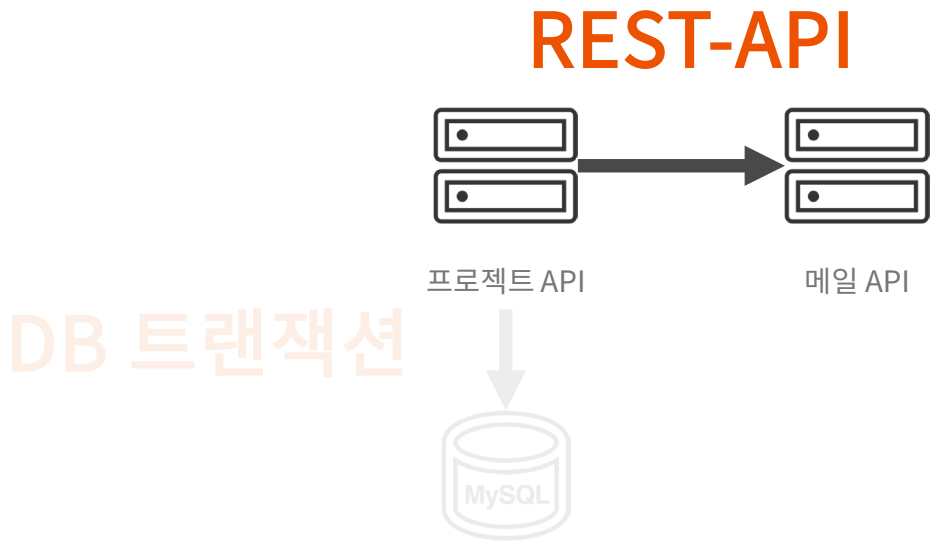
NHN FORWARD



RDB를 사용하는 애플리케이션에서 전달 방법



RDB를 사용하는 애플리케이션에서 전달 방법



RDB를 사용하는 애플리케이션에서 전달 방법

트랜잭션 Commit 이벤트를 사용하는 방법

- @TransactionalEventListener
- TransactionSynchronizationManager, TransactionSynchronization

RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

@TransactionalEventListener

```
@Service
public class EventHandler {

    @TransactionalEventListener(phase = TransactionPhase.AFTER_COMMIT)
    public void propagate(CreateTaskEvent event) {
        // 이벤트 발생 로직
        // + restTemplate.execute(...);
        // + rabbitTemplate.send(...)
    }
}
```

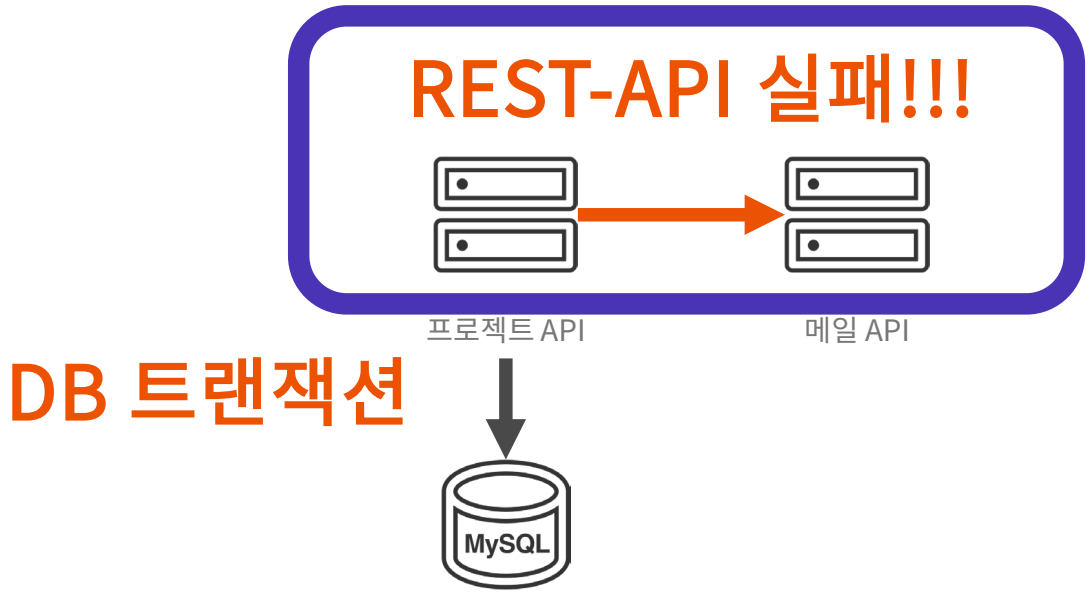
RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

@TransactionalEventListener

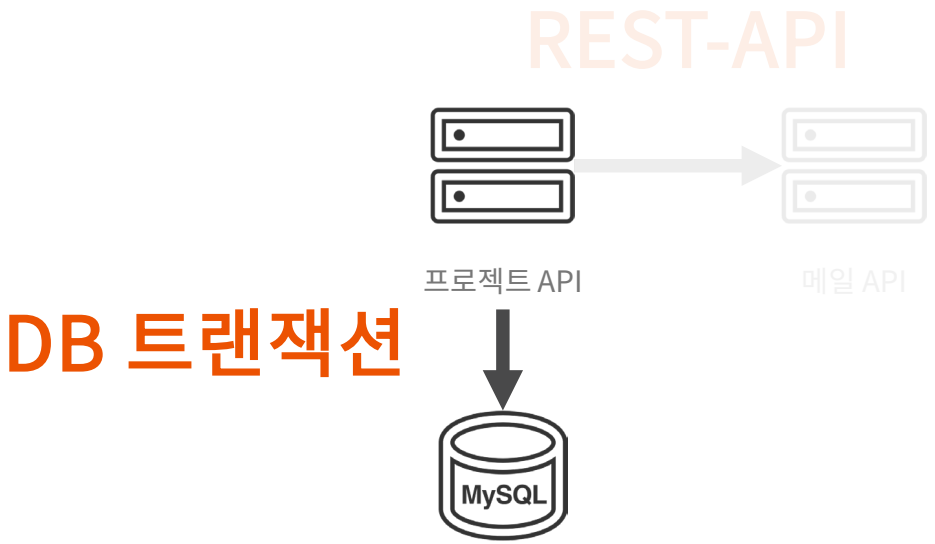
```
@Service
public class EventHandler {

    @TransactionalEventListener(phase = TransactionPhase.AFTER_COMMIT)
    public void propagate(CreateTaskEvent event) {
        // 이벤트 발생 로직
        // + restTemplate.execute(...);
        // + rabbitTemplate.send(...)
    }
}
```

RDB를 사용하는 애플리케이션에서 전달 방법



RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD



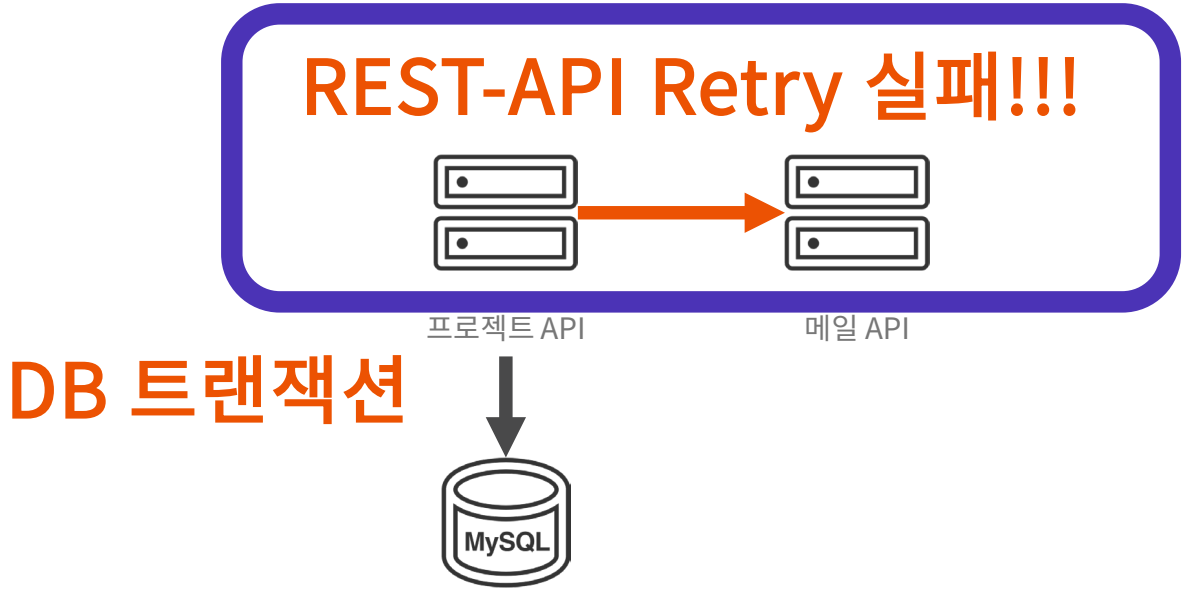
RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

@TransactionalEventListener + @Retryable

```
@Service
public class EventHandler {

    @Retryable
    @TransactionalEventListener(phase = TransactionPhase.AFTER_COMMIT)
    public void propagate(CreateTaskEvent event) {
        // 이벤트 발생 로직
        // + restTemplate.execute(...);
        // + rabbitTemplate.send(...)
    }
}
```

RDB를 사용하는 애플리케이션에서 전달 방법



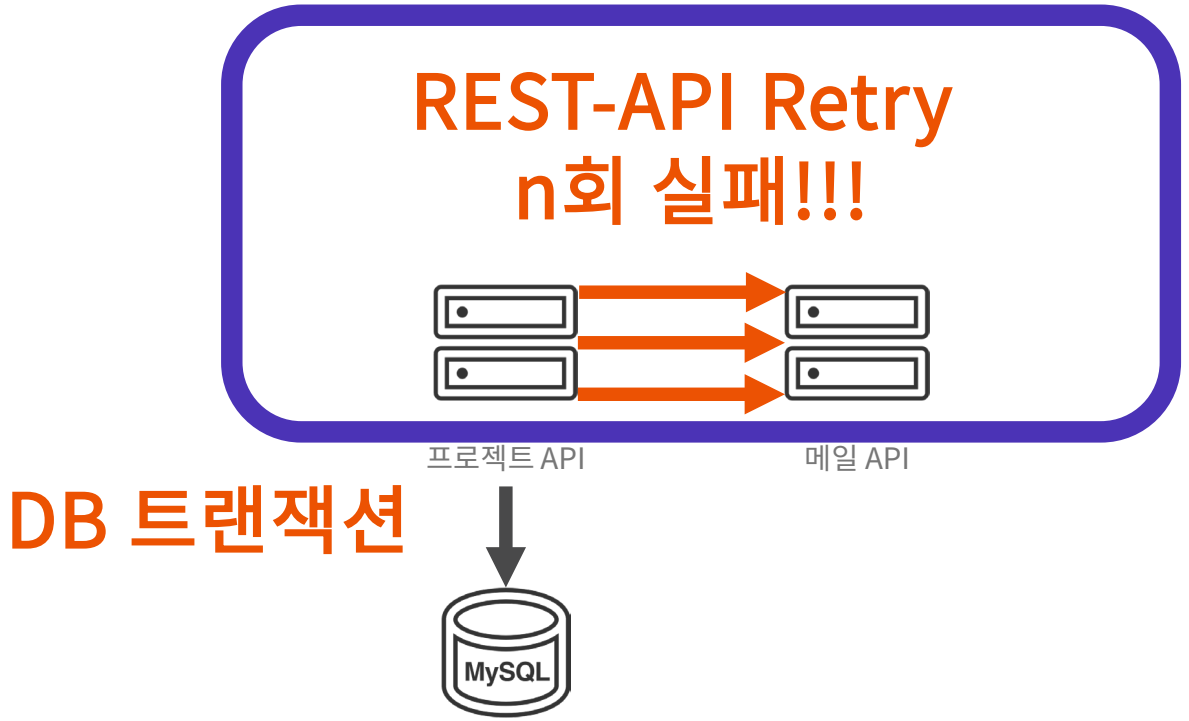
RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

@TransactionalEventListener + @Retryable

```
@Service
public class EventHandler {

    @Retryable(
            maxAttempts = 3,
            backoff = @Backoff(delay = 100L)
    )
    @TransactionalEventListener(phase = TransactionPhase.AFTER_COMMIT)
    public void propagate(CreateTaskEvent event) {
        // 이벤트 발생 로직
        // + restTemplate.execute(...);
        // + rabbitTemplate.send(...)
    }
}
```

RDB를 사용하는 애플리케이션에서 전달 방법



실패 없는

트랜잭션 처리와 **이벤트 전달**

이라면?

RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

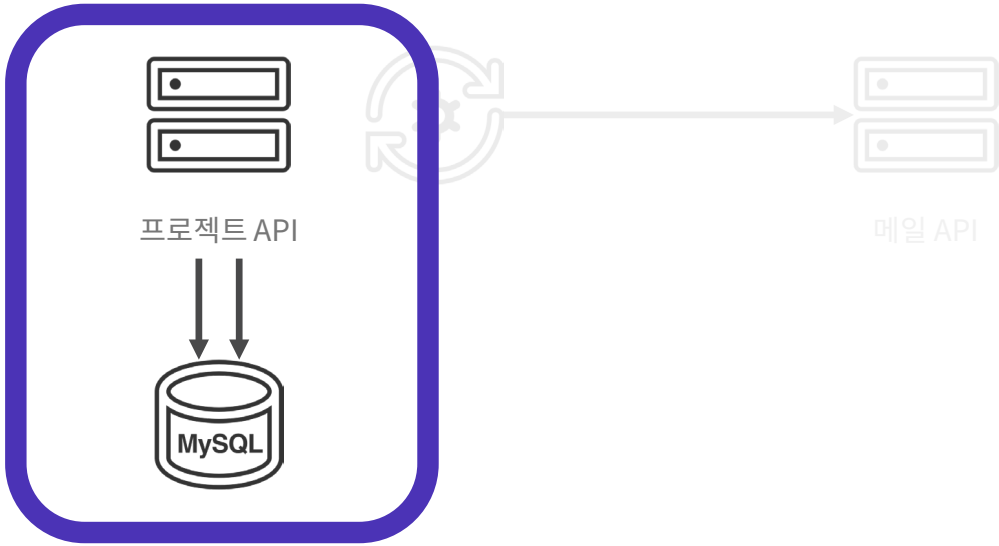
마이크로서비스 아키텍처 패턴

- Transactional Outbox Pattern
  - RDB를 Message Queue로 사용
  - OLTP에 Event Message를 포함하는 패턴
- Polling Publisher Pattern
  - RDB Message Queue Polling & Publishing

RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

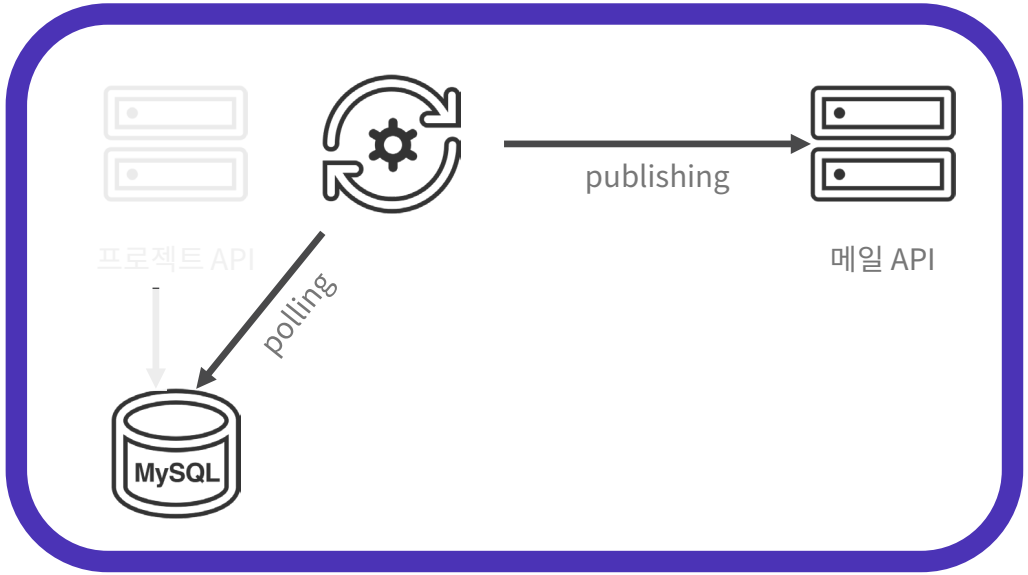
Transactional outbox 패턴
["이벤트", "데이터"]



RDB를 사용하는 애플리케이션에서 전달 방법

Polling Publisher 패턴

[이벤트]



RDB를 사용하는 애플리케이션에서 전달 방법

필드명	데이터 타입	설명
event_id	BIGINT	이벤트의 순서를 보장
created_at	Datetime(3~6)	이벤트 발생 시간
status	smallint	Ready(0) / Done (1)
payload	jsonb	JSON 타입의 Message payload
…	…	…

RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

Transactional Outbox 패턴을 적용한 코드

```
@Service
public class CreateTaskService implements CreateTaskUserCase {

    @Transactional
    public CreateTaskResponse createTask(CreateTaskCommand createTaskCommand) {
        Task task = createTaskCommand.toTask();

        taskRepository.save(task);
        eventRepository.save(CreateTaskEvent.of(task));

        return CreateTaskResponse.of(task);
    }
}
```

RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

Transactional Outbox 패턴을 적용한 코드

```
@Service
public class CreateTaskService implements CreateTaskUserCase {

    @Transactional
    public CreateTaskResponse createTask(CreateTaskCommand createTaskCommand) {
        Task task = createTaskCommand.toTask();

        taskRepository.save(task);
        eventRepository.save(CreateTaskEvent.of(task));

        return CreateTaskResponse.of(task);
    }
}
```

RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

Polling Publisher 패턴을 적용한 코드

```
@Service
public class MessagePublisher {

    @Scheduled(cron = "0/5 * * * * *")
    @Transactional
    public void publish() {
        LocalDateTime now = LocalDateTime.now();
        eventRepository.findByCreatedAtBefore(now, EventStatus.READY)
                       .stream()
                       .map(event -> restTemplate.execute(event))
                       .map(event -> event.done())
                       .forEach(eventRepository::save);
    }
}
```

RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

Polling Publisher 패턴을 적용한 코드

```
@Service
public class MessagePublisher {

    @Scheduled(cron = "0/5 * * * * *")
    @Transactional
    public void publish() {
        LocalDateTime now = LocalDateTime.now();
        eventRepository.findByCreatedAtBefore(now, EventStatus.READY)
                       .stream()
                       .map(event -> restTemplate.execute(event))
                       .map(event -> event.done())
                       .forEach(eventRepository::save);

    }
}
```

RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

Polling Publisher 패턴을 적용한 코드

```
@Service
public class MessagePublisher {

    @Scheduled(cron = "0/5 * * * * *")
    @Transactional
    public void publish() {
        LocalDateTime now = LocalDateTime.now();
        eventRepository.findByCreatedAtBefore(now, EventStatus.READY)
                       .stream()
                       .map(event -> restTemplate.execute(event))
                       .map(event -> event.done())
                       .forEach(eventRepository::save);
    }
}
```

RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

Polling Publisher 패턴을 적용한 코드

```
@Service
public class MessagePublisher {

    @Scheduled(cron = "0/5 * * * * *")
    @Transactional
    public void publish() {
        LocalDateTime now = LocalDateTime.now();
        eventRepository.findByCreatedAtBefore(now, EventStatus.READY)
                       .stream()
                       .map(event -> restTemplate.execute(event))
                       .map(event -> event.done())
                       .forEach(eventRepository::save);
    }
}
```

RDB를 사용하는 애플리케이션에서 전달 방법

NHN FORWARD

Polling Publisher 패턴을 적용한 코드

```
@Service
public class MessagePublisher {

    @Scheduled(cron = "0/5 * * * * *")
    @Transactional
    public void publish() {
        LocalDateTime now = LocalDateTime.now();
        eventRepository.findByCreatedAtBefore(now, EventStatus.READY)
                       .stream()
                       .map(event -> restTemplate.execute(event))
                       .map(event -> event.done())
                       .forEach(eventRepository::save);
    }
}
```

RDB를 사용하는 애플리케이션에서 전달 방법

- 장점
  - REST-API 환경에서 At-least-once를 구현할 수 있다.
- 단점
  - Polling, Publisher 과정으로 의한 지연 처리
  - 데이터 베이스 부하
  - 데이터 베이스 비례한 처리속도

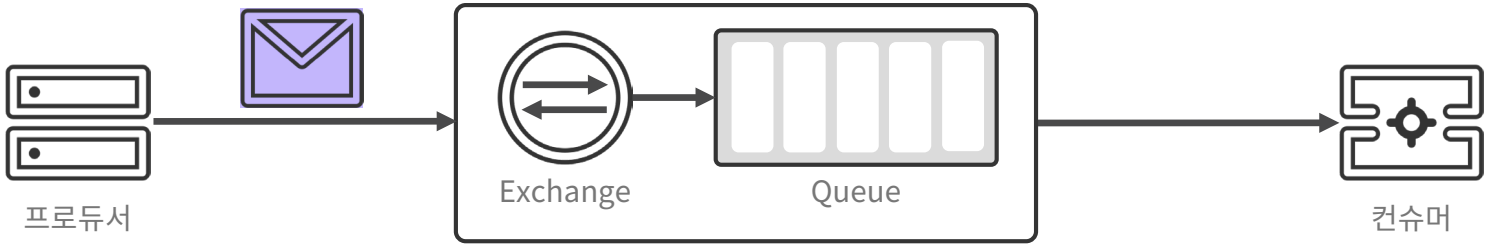
RabbitMQ를 사용한 전달 방법

RabbitMQ를 사용한 전달 방법

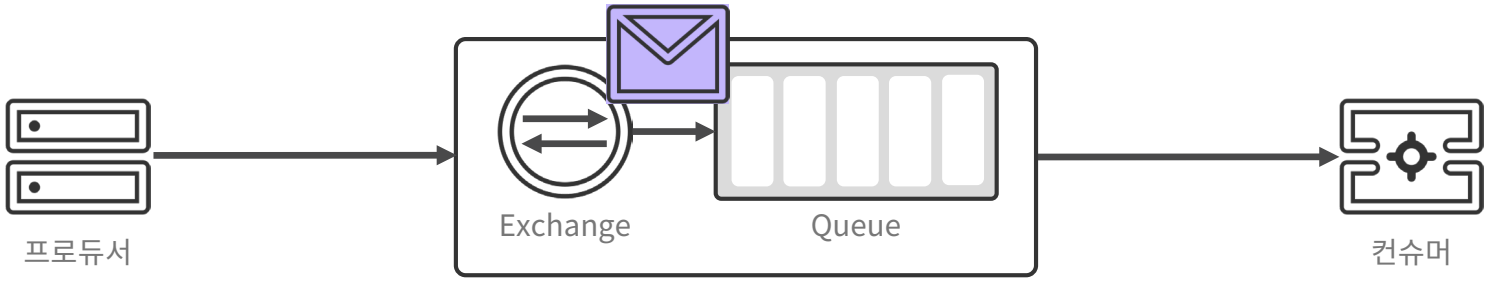
RabbitMQ

- AMQP(Advanced Message Queuing Protocol)을 구현한 메시지 브로커
- Publish/Subscribe 방식 지원
- ACK(Acknowledgement)를 메시지 응답 처리 메커니즘
  - Producer Confirm
  - Consumer Ack

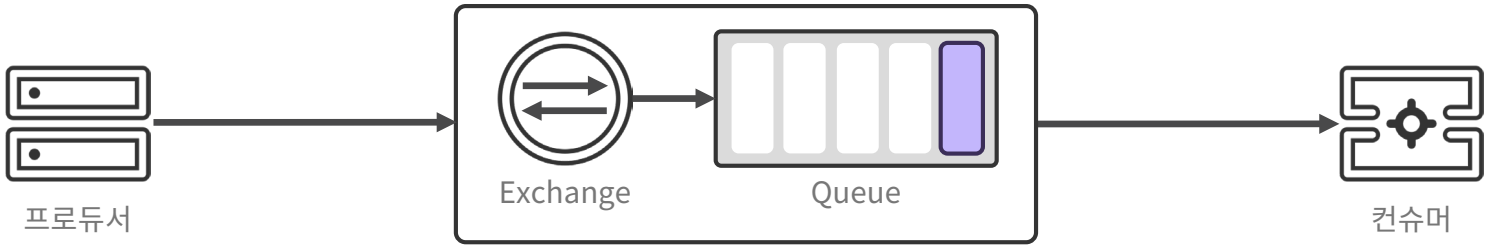
RabbitMQ를 사용한 전달 방법



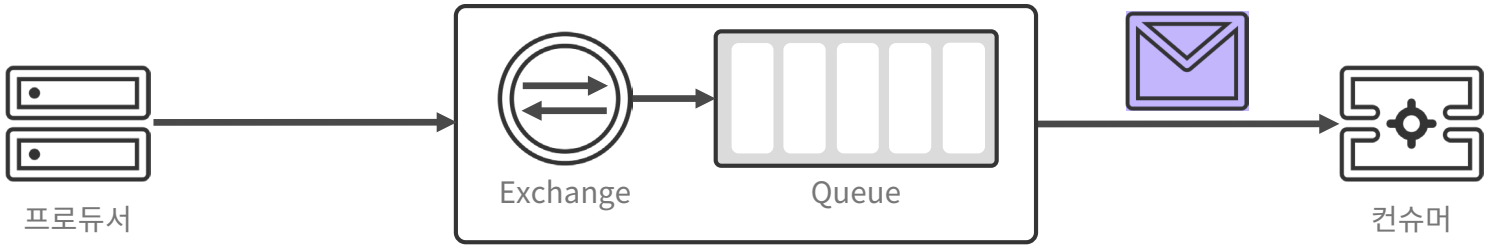
RabbitMQ를 사용한 전달 방법



RabbitMQ를 사용한 전달 방법

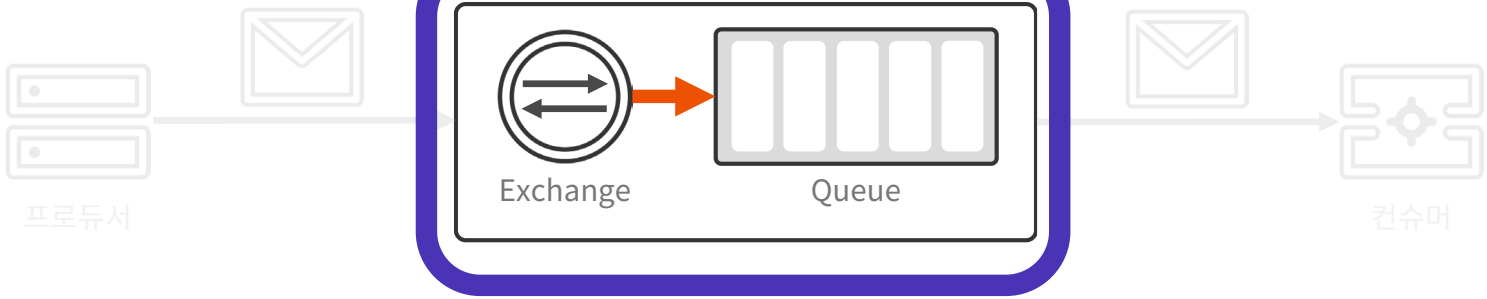


RabbitMQ를 사용한 전달 방법

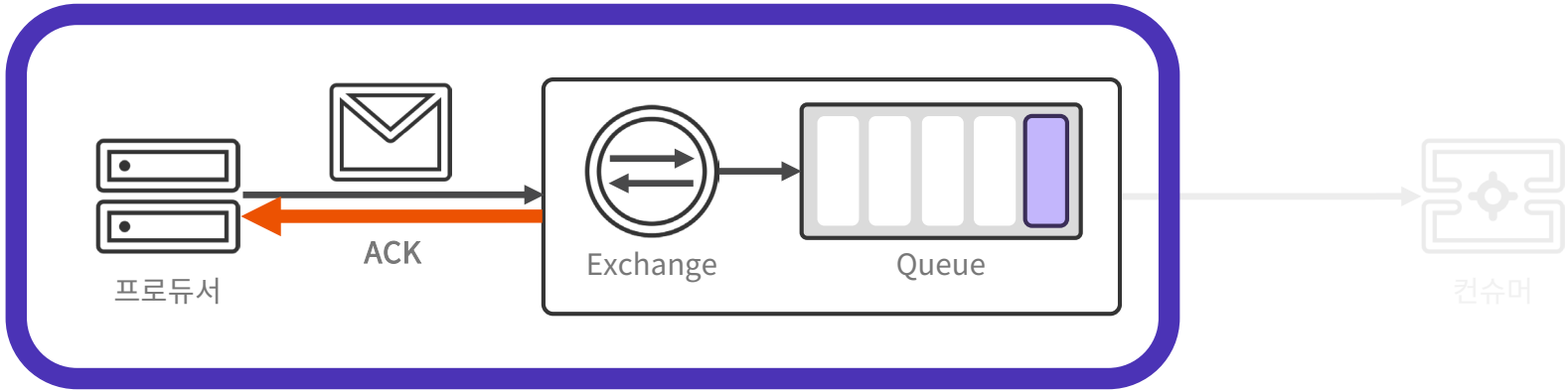


RabbitMQ를 사용한 전달 방법

Routing 실패??



RabbitMQ를 사용한 전달 방법



Producer Confirm

RabbitMQ를 사용한 전달 방법

Producer Confirm

- org.springframework.amqp.rabbit.connection.CorrelationData.java
  - Producer Confirm을 확인할 수 있는 기본 클래스
  - RabbitTemplate와 사용

RabbitMQ를 사용한 전달 방법

NHN FORWARD

CorrelationData를 사용한 코드

```
@Service
public class MessagePublisher {

    public void sendMessage(CreateTaskEvent createTaskEvent) throws JsonProcessingException {
        String json = objectMapper.writeValueAsString(createTaskEvent);
        rabbitTemplate.send(EXCHANGE_NAME,
                            ROUTING_KEY,
                            new Message(json.getBytes(StandardCharsets.UTF_8)),
                            new CorrelationData(UUID.randomUUID().toString()));
    }
}
```

RabbitMQ를 사용한 전달 방법

NHN FORWARD

CorrelationData를 사용한 코드

```
@Service
public class MessagePublisher {

    public void sendMessage(CreateTaskEvent createTaskEvent) throws JsonProcessingException {
        String json = objectMapper.writeValueAsString(createTaskEvent);
        rabbitTemplate.send(EXCHANGE_NAME,
                            ROUTING_KEY,
                            new Message(json.getBytes(StandardCharsets.UTF_8)),
                            new CorrelationData(UUID.randomUUID().toString()));
    }
}
```

RabbitMQ를 사용한 전달 방법

RabbitTempate의 ConfirmCallback을 사용한 코드

```
@Bean
public RabbitTemplate rabbitTemplate(ConnectionFactory rabbitConnectionFactory) {
    RabbitTemplate rabbitTemplate = new RabbitTemplate(rabbitConnectionFactory);

    // ...
    rabbitTemplate.setConfirmCallback((correlationData, ack, cause) -> {
        if (!ack) {
            Message message = correlationData.getReturned().getMessage();
            byte[] body = message.getBody();
            log.error("Fail to produce. ID: {}", correlationData.getId());
        }
    });
    return rabbitTemplate;
}
```

RabbitMQ를 사용한 전달 방법

NHN FORWARD

RabbitTempate의 ConfirmCallback을 사용한 코드

```
@Bean
public RabbitTemplate rabbitTemplate(ConnectionFactory rabbitConnectionFactory) {
    RabbitTemplate rabbitTemplate = new RabbitTemplate(rabbitConnectionFactory);

    // ...
    rabbitTemplate.setConfirmCallback((correlationData, ack, cause) -> {
        if (!ack) {
            Message message = correlationData.getReturned().getMessage();
            byte[] body = message.getBody();
            log.error("Fail to produce. ID: {}", correlationData.getId());
        }
    });
    return rabbitTemplate;
}
```

RabbitMQ를 사용한 전달 방법

NHN FORWARD

RabbitTempate의 ConfirmCallback을 사용한 코드

```
@Bean
public RabbitTemplate rabbitTemplate(ConnectionFactory rabbitConnectionFactory) {
    RabbitTemplate rabbitTemplate = new RabbitTemplate(rabbitConnectionFactory);

    // ...
    rabbitTemplate.setConfirmCallback((correlationData, ack, cause) -> {
        if (!ack) {
            Message message = correlationData.getReturned().getMessage();
            byte[] body = message.getBody();
            log.error("Fail to produce. ID: {}", correlationData.getId());
        }
    });
    return rabbitTemplate;
}
```

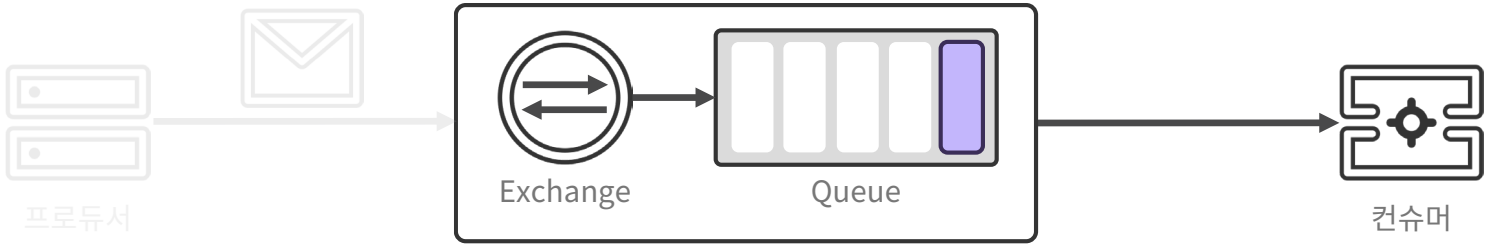
RabbitMQ를 사용한 전달 방법

NHN FORWARD

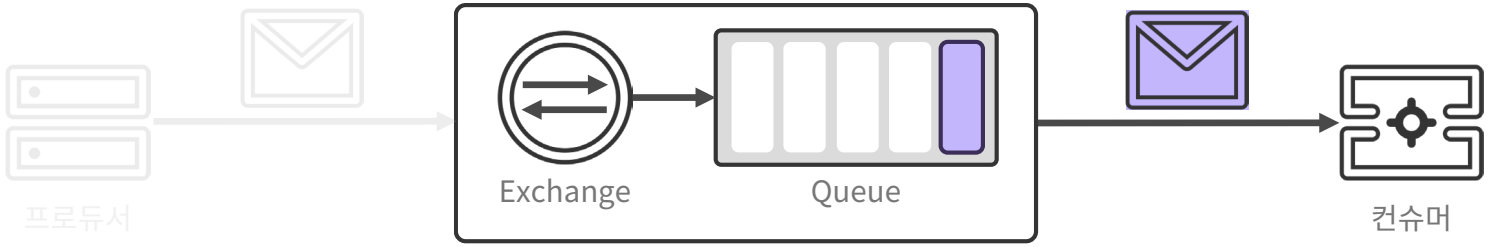
Publisher Confirm 설정 활성

```
// #1. Application.properties
spring.rabbitmq.publisher-confirm-type=correlated

// #2. 스프링 빈을 직접 정의하는 경우
@Bean
public ConnectionFactory rabbitConnectionFactory() {
    CachingConnectionFactory cf = new CachingConnectionFactory();
    //...
    cf.setPublisherConfirmType(CachingConnectionFactory.ConfirmType.CORRELATED);
    return cf;
}
```

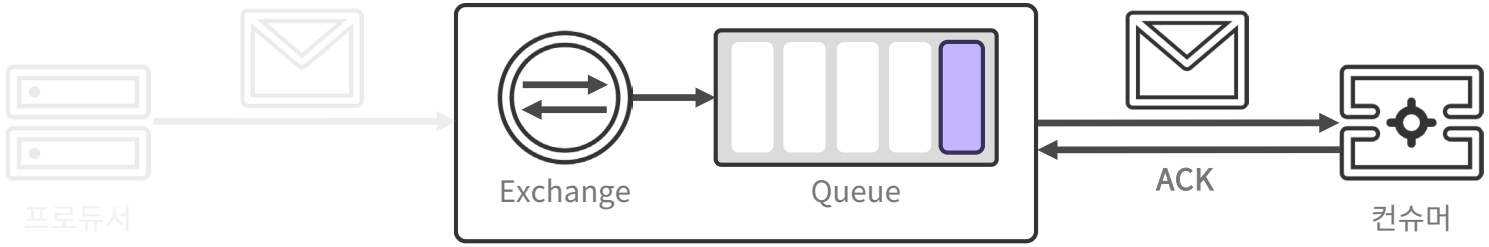
RabbitMQ를 사용한 전달 방법



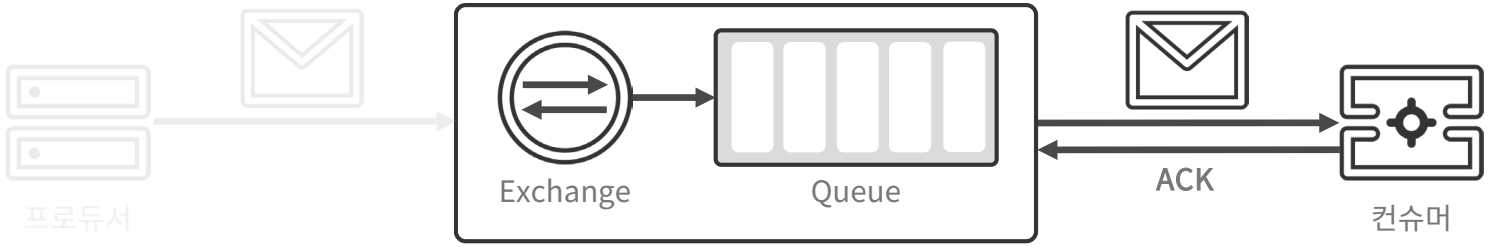
RabbitMQ를 사용한 전달 방법



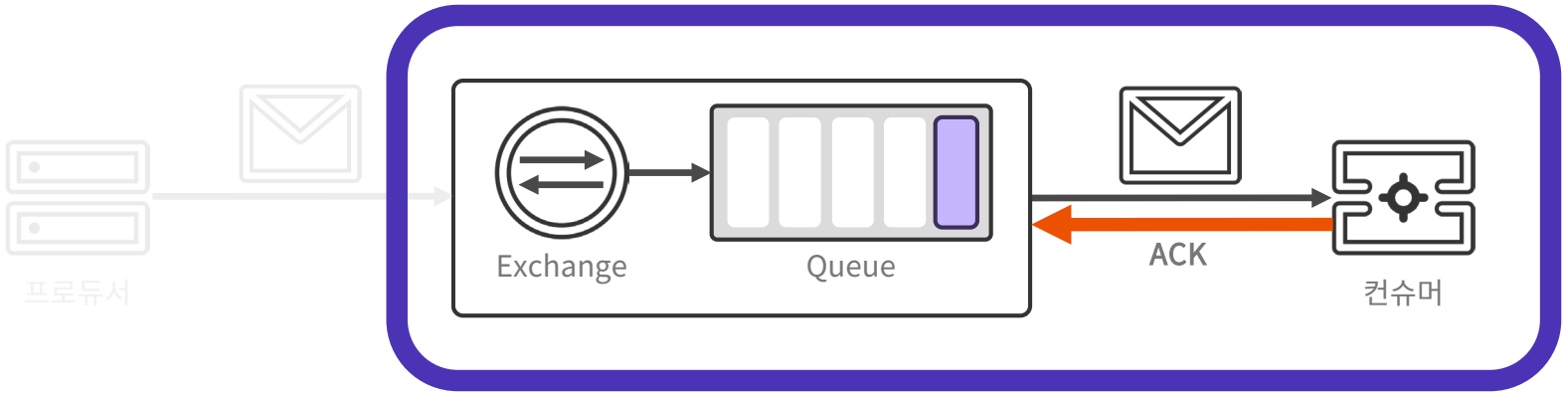
RabbitMQ를 사용한 전달 방법



RabbitMQ를 사용한 전달 방법



RabbitMQ를 사용한 전달 방법



Consumer Acknowledge

RabbitMQ를 사용한 전달 방법

Consumer Ack

- com.rabbitmq.client.Channel.java
  - RabbitMQ와 애플리케이션 사이의 가상의 커넥션
  - Queue 에서 데이터를 발행(Publishing), 소비(Consuming)하는 기능을 제공
  - ACK, NACK

RabbitMQ를 사용한 전달 방법

@RabbitListener 애너테이션

```java
@Component
public class MessageListener {

    @RabbitListener(queues = "dooray.task")
    public void receiveMessage(Message message, Channel channel) {

        // 수동 ACK 전송
        channel.basicAck(message.getMessageProperties().getDeliveryTag(), false);

        // 수동 NACK 전송
        channel.basicNack(message.getMessageProperties().getDeliveryTag(), false, true);
    }
}
```

RabbitMQ를 사용한 전달 방법

@RabbitListener 애너테이션

```
@Component
public class MessageListener {

    @RabbitListener(queues = "dooray.task")
    public void receiveMessage(Message message, Channel channel) {

        // 수동 ACK 전송
        channel.basicAck(message.getMessageProperties().getDeliveryTag(), false);

        // 수동 NACK 전송
        channel.basicNack(message.getMessageProperties().getDeliveryTag(), false, true);
    }
}
```

RabbitMQ를 사용한 전달 방법

NHN FORWARD

@RabbitListener 애너테이션

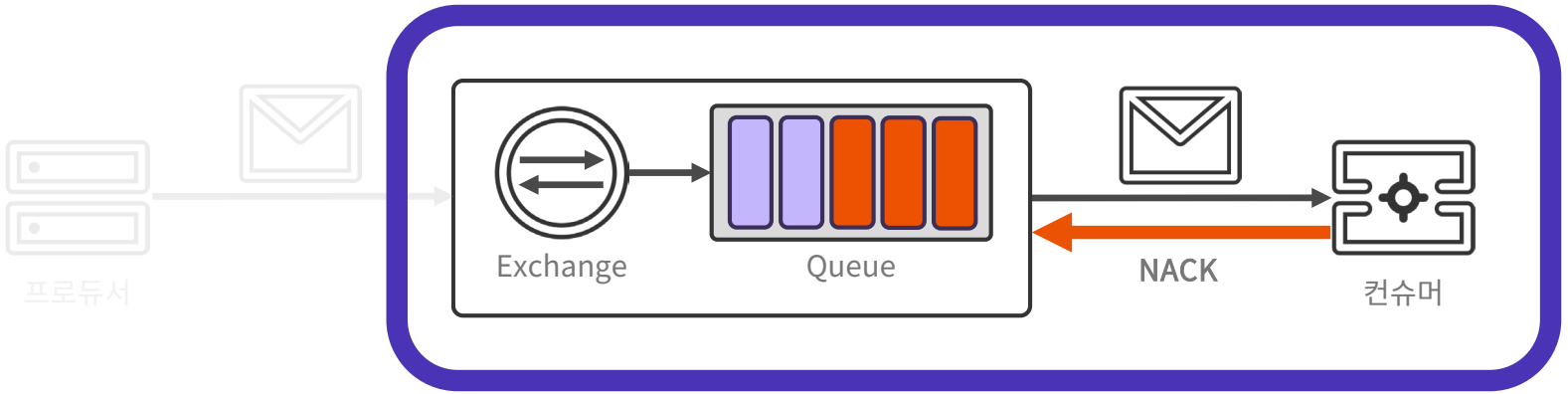
```
@Component
public class MessageListener {

    @RabbitListener(queues = "dooray.task")
    public void receiveMessage(Message message, Channel channel) {

        // 수동 ACK 전송
        channel.basicAck(message.getMessageProperties().getDeliveryTag(), false);

        // 수동 NACK 전송
        channel.basicNack(message.getMessageProperties().getDeliveryTag(), false, true);
    }
}
```

RabbitMQ를 사용한 전달 방법

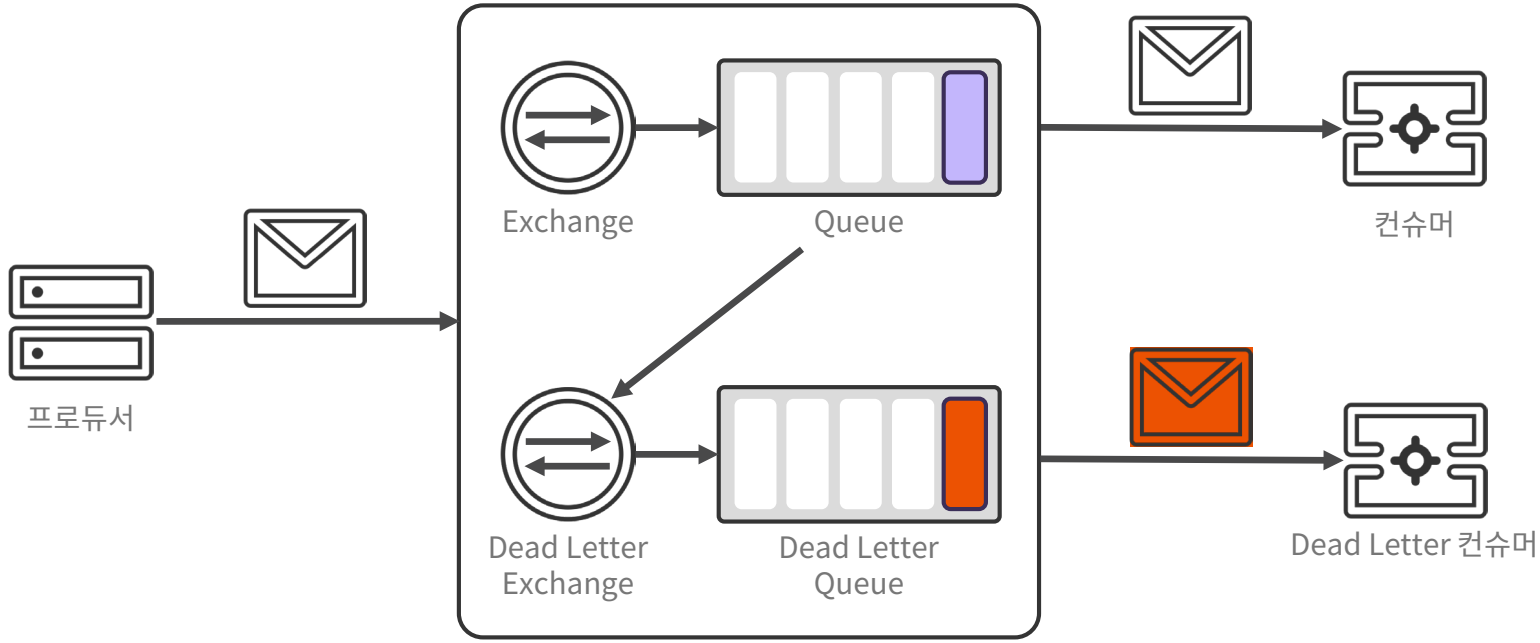
Consumer Ack

- org.springframework.amqp.core.MessageListener 인터페이스
  - BatchMessagListener.java
  - ChannelAwareBatchMessageListener.java
  - ChannelAwareMessageListener.java

RabbitMQ를 사용한 전달 방법



RabbitMQ를 사용한 전달 방법



RabbitMQ를 사용한 전달 방법

Retry 후 DeadLetter로 이동

```
@Bean
public SimpleRabbitListenerContainerFactory retryContainerFactory(ConnectionFactory connFactory) {

    //...
    var containerFactory = new SimpleRabbitListenerContainerFactory();
    containerFactory.setConnectionFactory(connFactory);
    containerFactory.setAdviceChain(
            RetryInterceptorBuilder.stateless()
                                   .maxAttempts(3)
                                   .backOffOptions(1000, 2, 2000)
                                   .recoverer(new RejectAndDontRequeueRecoverer())
                                   .build());
    return containerFactory;
}
```

RabbitMQ를 사용한 전달 방법

Retry 후 DeadLetter로 이동

```
@Bean
public SimpleRabbitListenerContainerFactory retryContainerFactory(ConnectionFactory connFactory) {

    //...
    var containerFactory = new SimpleRabbitListenerContainerFactory();
    containerFactory.setConnectionFactory(connFactory);
    containerFactory.setAdviceChain(
            RetryInterceptorBuilder.stateless()
                                   .maxAttempts(3)
                                   .backOffOptions(1000, 2, 2000)
                                   .recoverer(new RejectAndDontRequeueRecoverer())
                                   .build());

    return containerFactory;
}
```

RabbitMQ를 사용한 전달 방법

NHN FORWARD

Retry 후 DeadLetter로 이동

```
@Bean
public SimpleRabbitListenerContainerFactory retryContainerFactory(ConnectionFactory connFactory) {

    //...
    var containerFactory = new SimpleRabbitListenerContainerFactory();
    containerFactory.setConnectionFactory(connFactory);
    containerFactory.setAdviceChain(
            RetryInterceptorBuilder.stateless()
                                   .maxAttempts(3)
                                   .backOffOptions(1000, 2, 2000)
                                   .recoverer(new RejectAndDontRequeueRecoverer())
                                   .build());

    return containerFactory;
}
```

RabbitMQ를 사용한 전달 방법

NHN FORWARD

Retry 후 DeadLetter로 이동

```
@Bean
public SimpleRabbitListenerContainerFactory retryContainerFactory(ConnectionFactory connFactory) {

    //...
    var containerFactory = new SimpleRabbitListenerContainerFactory();
    containerFactory.setConnectionFactory(connFactory);
    containerFactory.setAdviceChain(
            RetryInterceptorBuilder.stateless()
                                   .maxAttempts(3)
                                   .backOffOptions(1000, 2, 2000)
                                   .recoverer(new RejectAndDontRequeueRecoverer())
                                   .build());

    return containerFactory;
}
```

RabbitMQ를 사용한 전달 방법

NHN FORWARD

Retry 후 DeadLetter로 이동

```
@Bean
public SimpleRabbitListenerContainerFactory retryContainerFactory(ConnectionFactory connFactory) {

    //...
    var containerFactory = new SimpleRabbitListenerContainerFactory();
    containerFactory.setConnectionFactory(connFactory);
    containerFactory.setAdviceChain(
            RetryInterceptorBuilder.stateless()
                                   .maxAttempts(3)
                                   .backOffOptions(1000, 2, 2000)
                                   .recoverer(new RejectAndDontRequeueRecoverer())
                                   .build());

    return containerFactory;
}
```

RabbitMQ를 사용한 전달 방법

DeadLetter Queue 리스너

```
@RabbitListener(
        queues = "dooray.member.deadletter",
        containerFactory = "deadLetterContainerFactory"
)
public void onDeadLetterMessage(Map<String, Object> rawEvent) {
    // Alert failure or fallback
}
```

Kafka를 사용한 전달 방법

Kafka를 사용한 전달 방법

Producer Confirm

```
@Slf4j
@Component
@RequiredArgsConstructor
public class Producer {

    public void sendEvent(CreateTaskEvent event){
        ListenableFuture<SendResult<String, CreateTaskEvent>> future =
kafkaTemplate.send(TOPIC_TASK, event);
        future.addCallback(
                result -> log.info("offset : {}", result.getRecordMetadata().offset()),
                throwable -> log.error("fail to publish", throwable)
        );
    }
}
```

Kafka를 사용한 전달 방법

NHN FORWARD

Producer Confirm

```
@Slf4j
@Component
@RequiredArgsConstructor
public class Producer {

    public void sendEvent(CreateTaskEvent event){
        ListenableFuture<SendResult<String, CreateTaskEvent>> future =
kafkaTemplate.send(TOPIC_TASK, event);
        future.addCallback(
                result -> log.info("offset : {}", result.getRecordMetadata().offset()),
                throwable -> log.error("fail to publish", throwable)
        );
    }
}
```

Kafka를 사용한 전달 방법

NHN FORWARD

Producer Confirm

```
@Slf4j
@Component
@RequiredArgsConstructor
public class Producer {

    public void sendEvent(CreateTaskEvent event){
        ListenableFuture<SendResult<String, CreateTaskEvent>> future =
kafkaTemplate.send(TOPIC_TASK, event);
        future.addCallback(
                result -> log.info("offset : {}", result.getRecordMetadata().offset()),
                throwable -> log.error("fail to publish", throwable)
        );
    }
}
```

Kafka를 사용한 전달 방법

Producer Confirm

```
@Slf4j
@Component
@RequiredArgsConstructor
public class Producer {

    public void sendEvent(CreateTaskEvent event){
        ListenableFuture<SendResult<String, CreateTaskEvent>> future =
kafkaTemplate.send(TOPIC_TASK, event);
        future.addCallback(
                result -> log.info("offset : {}", result.getRecordMetadata().offset()),
                throwable -> log.error("fail to publish", throwable)
        );
    }
}
```

Kafka를 사용한 전달 방법

NHN FORWARD

Producer Confirm

```
@Slf4j
@Component
@RequiredArgsConstructor
public class Producer {

    public void sendEvent(CreateTaskEvent event){
        ListenableFuture<SendResult<String, CreateTaskEvent>> future =
kafkaTemplate.send(TOPIC_TASK, event);
        future.addCallback(
                result -> log.info("offset : {}", result.getRecordMetadata().offset()),
                throwable -> log.error("fail to publish", throwable)
        );
    }
}
```

Kafka를 사용한 전달 방법

Consumer ACK - AcknowledgingMessageListener

```java
@FunctionalInterface
public interface AcknowledgingMessageListener<K, V> extends MessageListener<K, V> {
    default void onMessage(ConsumerRecord<K, V> data) {
        throw new UnsupportedOperationException("Container should never call this");
    }

    void onMessage(ConsumerRecord<K, V> var1, Acknowledgment var2);
}
```

Kafka를 사용한 전달 방법

NHN FORWARD

Consumer ACK - AcknowledgingMessageListener

```
@FunctionalInterface
public interface AcknowledgingMessageListener<K, V> extends MessageListener<K, V> {
    default void onMessage(ConsumerRecord<K, V> data) {
        throw new UnsupportedOperationException("Container should never call this");
    }

    void onMessage(ConsumerRecord<K, V> var1, Acknowledgment var2);
}
```

Kafka를 사용한 전달 방법

NHN FORWARD

Consumer ACK - AcknowledgingMessageListener

```
@Override
@KafkaListener(
        //...
        containerFactory = "kafkaListenerContainerFactory"
)
public void onMessage(ConsumerRecord<String, String> consumerRecord, Acknowledgment acknowledgment)
{
    try {
        // Do something...
        acknowledgment.acknowledge();
    } catch (Exception e) {
        log.error("Error to receive messages.", e);
    }
}
```

Kafka를 사용한 전달 방법

NHN FORWARD

Consumer ACK - AcknowledgingMessageListener

```
@Override
@KafkaListener(
        //...
        containerFactory = "kafkaListenerContainerFactory"
)
public void onMessage(ConsumerRecord<String, String> consumerRecord, Acknowledgment acknowledgment)
{
    try {
        // Do something...
        acknowledgment.acknowledge();
    } catch (Exception e) {
        log.error("Error to receive messages.", e);
    }
}
```

Kafka를 사용한 전달 방법

Consumer ACK - ConsumerFactory 설정

```
@Bean
public ConsumerFactory<String, String> consumerFactory() {

    Map<String, Object> props = new HashMap<>();
    props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapServer);
    props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class);
    props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class);
    props.put(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "false");
    // ....

    return new DefaultKafkaConsumerFactory<>(props);
}
```

Kafka를 사용한 전달 방법

Consumer ACK - ConsumerFactory 설정

```
@Bean
public ConsumerFactory<String, String> consumerFactory() {

    Map<String, Object> props = new HashMap<>();
    props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapServer);
    props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class);
    props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class);
    props.put(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "false");
    // ....

    return new DefaultKafkaConsumerFactory<>(props);
}
```

마무리

분산 시스템에서 데이터를 전달하는 효율적인 방법

- Event driven Architecture의 기본은 데이터 전달
- 최소 At Least Once (최소 한번) 설정
- Producer Confirm, Consumer Ack 고려

Q & A

NHN FORWARD

고맙습니다.