Routing Model을 이용한 Log 수집

Routing 모델

Routing 모델은 메시지를 Routing Key에 따라 특정 큐에 전달하는 기능으로 Fanout Exchange와 함께 가장 일반적인 모델

Direct Exchange와 Topic Exchange 모두에서 사용 가능.

특징	Direct Exchange	Topic Exchange
라우팅 키 매칭 방식	정확히 일치	패턴 기반 (* 와 # 지원)
특징	매칭에 제한이 있음	매우 유연(다양한 패턴 매칭)
사용 사례	단순 명확한 목적의 라우팅	복잡하고 동적인 라우팅

- Topic 의 경우 패턴 매칭 기반 라우팅으로 binding key와 매칭되는 메시지만 수신.
  - * : 한 단어에 해당하는 모든 단어 수신
  - # : 0개 이상의 단어 (와일드 카드)

주요 특징

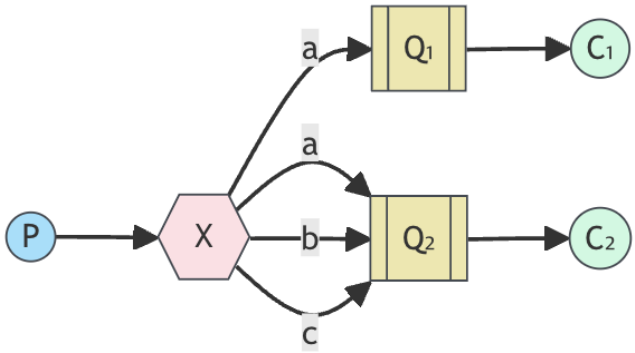
1. **고성능**
   - 메시지를 필요한 곳에만 전달하기 때문에 네트워크 부하 감소 효과가 있고 브로드캐스트보다 자원을 효율적으로 사용.
2. **라우팅 키 기반의 메시지 분배**:
   - 각 큐는 **하나 이상의 라우팅 키**와 매칭(Direct Exchange의 경우 라우팅 키가 정확히 일치해야 1:1 로 동작한다고 알고 있는데, 1:N으로 동작함 - 동일한 라우팅 키로 여러 큐에 바인딩 할 수 있음)
   - Fanout 방식과 달리, 메시지가 **특정 큐로만 전달**됨.
3. **바인딩 설정**:
   - Direct Exchange는 정확한 매칭 기반으로 메시지를 라우팅.

- Topic Exchange는 패턴 기반 매칭으로 다양한 방식으로 유연하게 연결
- Exchange와 큐 사이의 관계를 바인딩 키(binding key)를 통해 메시지가 전달

그러나 라우팅 키와 바인딩 키가 일치 하지 않을 경우 메시지가 전달이 안되고. 다수의 큐와 라우팅을 관리할 경우 복잡성이 늘어난다.

메시지 흐름

1. Producer가 메시지와 함께 **라우팅 키**를 설정해 메시지를 Exchange로 전송.
2. Exchange는 **바인딩 키**를 확인하고, 해당 키와 매칭되는 큐로 메시지를 전달.
3. Consumer는 해당 큐에서 메시지를 소비.

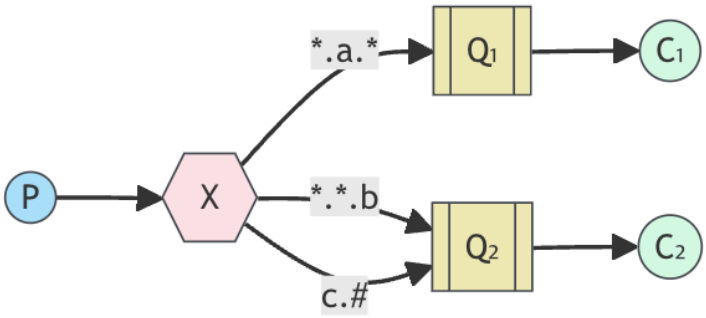


Topic Exchange 에러 로그 수집의 경우 아래와 같이 Topic 을 세분화 하여 적용할 수 있음

> 에러 로그 : `log.error`
> 경고 로그 :
> `log.warn`
> 정보성 로그 :
> `log.info`
> 사용자의 에러 :
> `user.error`

주문의 완료(order.completed) 이후 배송지시(order.completed.shipped), 재고 차감(order.completed.inventory), 마케팅 이메일 발송(order.completed.email) 등의 큐 전달

> ✔️ 각각 log.* 나 *.error으로 끝나는 모든 에러나 order.completed.* 모든 로그 수신 처리
> ✔️ # 와일드 카드로 모든 에러 수신 가능



Routing 모델의 활용 사례

- 로그 수집
- 주문 상태 전이 (주문 완료시 order.completed 를 키로 배송 지시, 재고 차감 지시, 이메일 마케팅 전송 큐 등에 분배)
- 채팅의 방 개설 단위로 분류하여 메시지 전달

튜토리얼 Step 5. 로그를 수집하는 Routing 모델 예제 (using DirectExchange)

- Exception이 발생했을 때 해당 메시지를 송/수신하는 로직으로 메시지를 routing
- 에러 로그별 매칭되는 큐로 전달도 가능 (Step 6. Topic Exchange)

로그 수집 Package Structure

```
src/main/java/net/harunote/hellomessagequeue git:[tutorial-step5
├── HelloController.java
├── HelloMessageQueueApplication.java
```

```
└── step5
    ├── CustomExceptionHandler.java
    ├── LogConsumer.java
    ├── LogController.java
    ├── LogPublisher.java
    └── RabbitMQConfig.java
```

개발 프로세스

1. RabbitMQConfig 설정 : 3개의 Queue Bean 선언(Error, Warn, Info), 3개의 Binding 빈 (to DirectExchage) 선언
2. 로그 발행 (LogPublisher) 빈 생성
3. 로그 소비 (LogConsumer) 빈 생성
4. Exception 처리 로직(CustomExsceptionHandler → LogPublisher로 전송
5. Controller로 REST API 작성
6. cURL 호출 테스트

```
curl -X GET "http://localhost:8080/api/logs/error" // error 로그
curl -X GET "http://localhost:8080/api/logs/warn"  // warn 로그
curl -X POST "http://localhost:8080/api/logs/info" \
     -H "Content-Type: application/json" \
     -d "\"System initialized successfully.\""
```

튜토리얼 Step 6. 패턴별 로그를 수집하는 Routing 모델 예제 (using TopicExchange)

개발 프로세스

1. All log queue 빈 추가, Topic Exchange로 변경
2. 들어오는 로그 패턴을 지정 (ex : log.error, log.warn, log.info ...)

3. 모든 로그를 받는 all log queue RabbitListener 추가
4. cURL 호출 테스트시 지정된 queue 와 모든 로그를 수집하는 큐에 동시에 적재되는지 확인