Berner
Fachhochschule

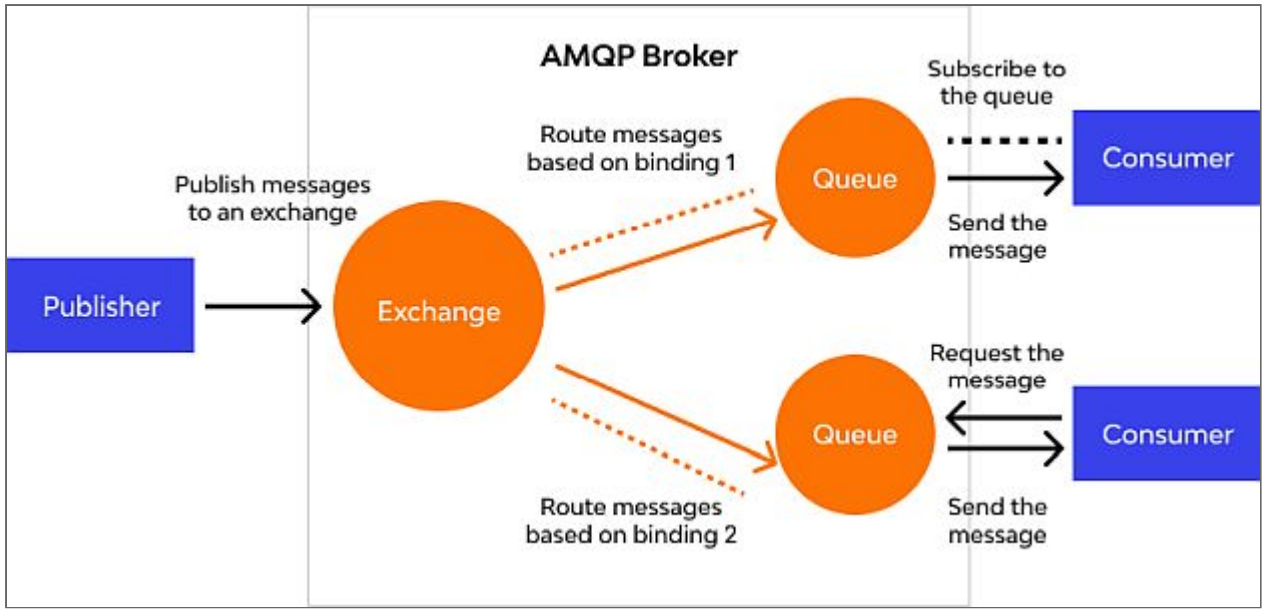
Spring AMQP

Stephan Fischli

Introduction

- AMQP is a standard for a platform-neutral binary messaging protocol
- Implementations can be written in different programming languages
- Well-known AMQP message brokers include **RabbitMQ**, OpenAMQ and StormMQ

Messaging Model



The AMQP messaging model is based on the following components:

- *Exchanges* receive messages from publishers and route them to queues
- *Bindings* connect exchanges to queues, optionally using *routing keys*
- *Queues* deliver messages to consumers

Exchange Types

There are four exchange types:

- Direct – routes messages to queues with a matching routing key
- Fanout – routes messages to all queues bound to the exchange
- Topic – routes messages based on pattern matching of routing keys
- Headers – routes messages based on message headers

Message Flow

1. A producer sends a message with an optional routing key to an exchange
2. The exchange routes the message to one or more queues based on the exchange type, its bindings, and the routing key
3. The message is delivered to consumers subscribed to the queues, or consumers explicitly fetch it from the queues
4. The message broker removes the message

Spring AMQP

- To use AMQP in a Spring Boot application, the `spring-boot-starter-amqp` starter can be added
- The RabbitMQ message broker is configured by the following application properties

```
spring.rabbitmq.host=localhost
spring.rabbitmq.port=5672
spring.rabbitmq.user=guest
spring.rabbitmq.password=guest
```

Configuration

- Exchanges, queues and bindings can be defined using bean methods in a configuration class

```
@Configuration
public class AMQPConfig {
    @Bean
    public TopicExchange exchange() {
        return new TopicExchange("order-exchange");
    }
    @Bean
    public Queue queue() {
        return new Queue("order-queue", false);
    }
    @Bean
    public Binding binding(Queue queue, TopicExchange exchange) {
        return BindingBuilder.bind(queue).to(exchange).with("order-
key");
    }
}
```

Sending Messages

- Messages can be sent to an exchange using the `RabbitTemplate` provided by Spring Boot

```
@Component
public class MessageSender {

    @Autowired
    private RabbitTemplate rabbitTemplate;

    public void sendMessage(String message) {
        rabbitTemplate.convertAndSend("order-exchange", "order-key",
message);
    }
}
```

Receiving Messages

- Messages can be received from a queue using a listener method

```
@Component
public class MessageReceiver {

    @RabbitListener(queues = "order-queue")
    public void receiveMessage(String message) {
        ...
    }
}
```