Discover the Confidence of Knowing.

KINGLAND®

Java Message Queue

2021 April

KINGLAND.COM

Prepare to begin

- Download: ActiveMQ:
- https://activemq.apache.org/components/classic/download/
- Github source code:

https://github.com/matt19870107/activemq-example

Agenda

- Why use message queue
- MOM （Message Oriented Middleware)
- How to set up ActiveMQ
- JMS (Java Message Service)
- Integrate in Spring Boot

Why using Message queue

Message queue VS Synchronous access

Message queue	Synchronous access
Low coupling	High coupling
Asynchronous execution	Synchronous execution
High performance	Low performance
Little hard to monitor	Easy to monitor
Message size has max limited	Request size has no limited

Message queue VS Synchronous access

ActiveMQ and MOM

MOM （Message Oriented Middleware)

- **Basic Function**
  To transfer information in the form of messages from one application to another or more applications

- **Main feature**
  1. Asynchronous message reception
  2. Reliable reception of messages

- **Famous MOM**
  - Kafka - High-throughput
  - RabbitMQ - It's fast and it works with good metrics/monitoring
  - ZeroMQ - Lightweight
  - RocketMQ - Million-level message accumulation capacity in a single
  - AWS SQS - Easy to use, reliable
  - **ActiveMQ - Easy to use**

  https://stackshare.io/stackups/kafka-vs-rabbitmq-vs-zeromq
  https://stackshare.io/stackups/rocketmq-vs-amazon-sqs-vs-activemq

Which one is popular?

- Kafka

  Monitor Kafka Metric Performance With Datadog

  + Follow | + I use this

Stacks	Followers	Votes
17.3K	16.3K	585

- RabbitMQ

  Monitor RabbitMQ Metrics In Real-Time With Datadog

  + Follow | + I use this

Stacks	Followers	Votes
16.1K	14K	520

- ZeroMQ

  + Follow | + I use this

Stacks	Followers	Votes
223	497	71

- Apache RocketMQ

  + Follow | + I use this

Stacks	Followers	Votes
38	159	8

- Amazon SQS

  Check out Private StackShare for Teams

  + Follow | + I use this

Stacks	Followers	Votes
2K	1.7K	166

- ActiveMQ

  + Follow | + I use this

Stacks	Followers	Votes
496	1.1K	76

ActiveMQ

- **What is ActiveMQ**
  ActiveMQ was presented by Apache, a open source and fully support JMS1.1 and J2EE 1.4 standards JMS provider to implement MOM.

Set up ActiveMQ in local

1. Start activeMQ
   cd activeMQ/bin | ./activemq start
2. Visit admin page to validate activeMQ is started
   http://localhost:8161/admin
3. Stop activeMQ
   cd activeMQ/bin | ./activemq stop

Send your first message

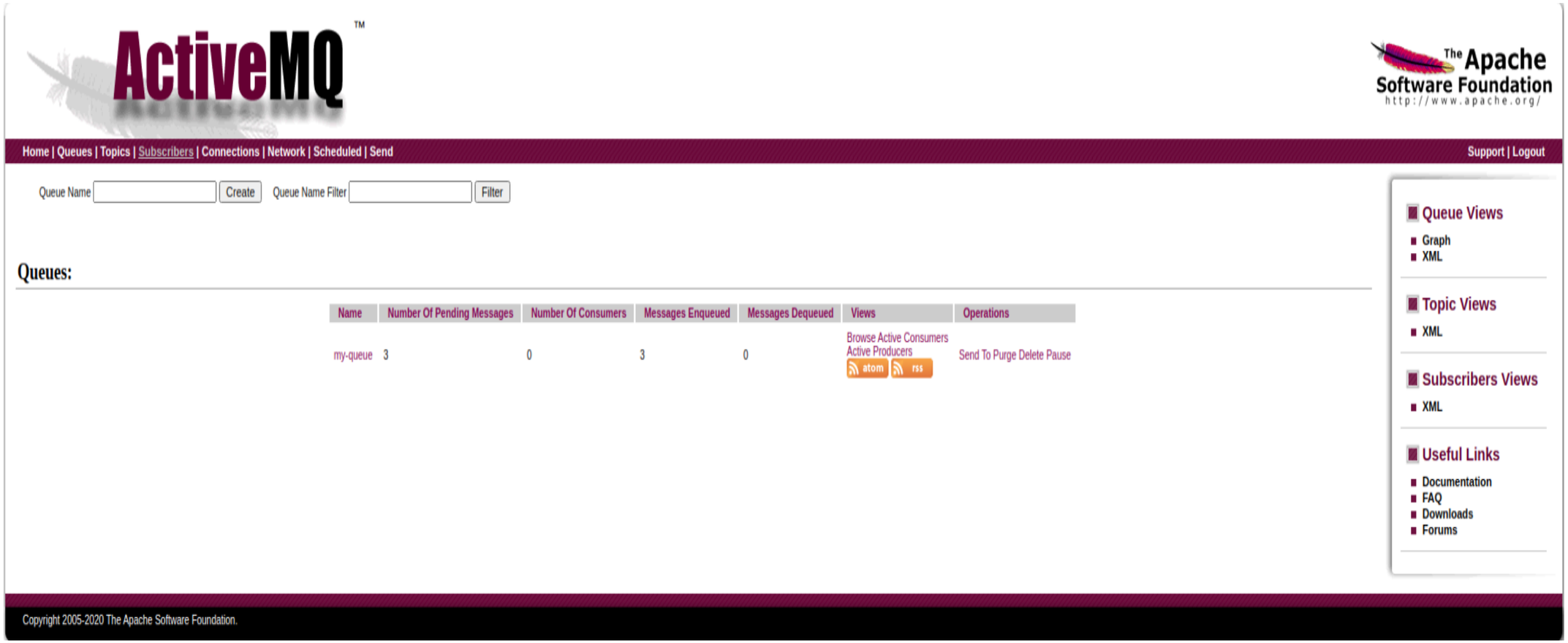
- **Add gradle dependencies for activeMQ**

```
compile group: 'org.apache.activemq', name: 'activemq-all', version: '5.16.1'
compile group: 'org.apache.xbean', name: 'xbean-spring', version: '4.18'
```

- **Send Message to ActiveMQ**

```
public static void main (String args[]) throws JMSException, InterruptedException {
    ConnectionFactory connectionFactory = new ActiveMQConnectionFactory( brokerURL: "tcp://127.0.0.1:61616");
    Connection connection = connectionFactory.createConnection();
    connection.start();

    Session session = connection.createSession(Boolean.TRUE, Session.AUTO_ACKNOWLEDGE);
    Destination destination = session.createQueue( queueName: "my-queue");

    MessageProducer producer = session.createProducer(destination);

    for(int i = 0; i<3; i++) {
        TextMessage message = session.createTextMessage("message-" + i);
        Thread.sleep( l: 1000);
        producer.send(message);
    }

    session.commit();;
    session.close();
    connection.close();
}
```

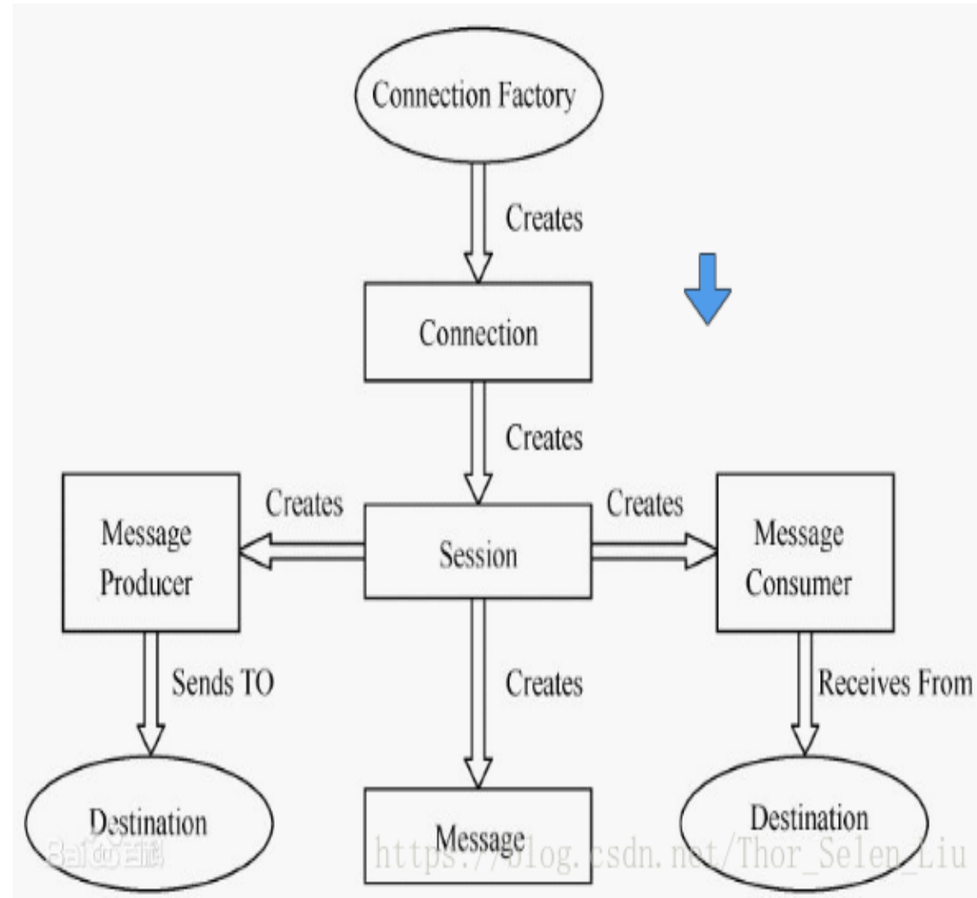
Queue status view



KINGLAND

Receive your first message

```java
public static void main (String args[]) throws JMSException, InterruptedException {
    ConnectionFactory connectionFactory = new ActiveMQConnectionFactory( brokerURL: "tcp://127.0.0.1:61616");
    Connection connection = connectionFactory.createConnection();
    connection.start();

    final Session session = connection.createSession(Boolean.TRUE, Session.AUTO_ACKNOWLEDGE);
    Destination destination = session.createQueue( queueName: "my-queue");

   MessageConsumer consumer  = session.createConsumer(destination);

    while (true) {
        TextMessage message = (TextMessage) consumer.receive();
        if (message != null) {
            String text = message.getText();
            System.out.println(text);
            session.commit();
        } else {
            break;
        }
    }
    consumer.close();
    session.close();
    connection.close();
}
```

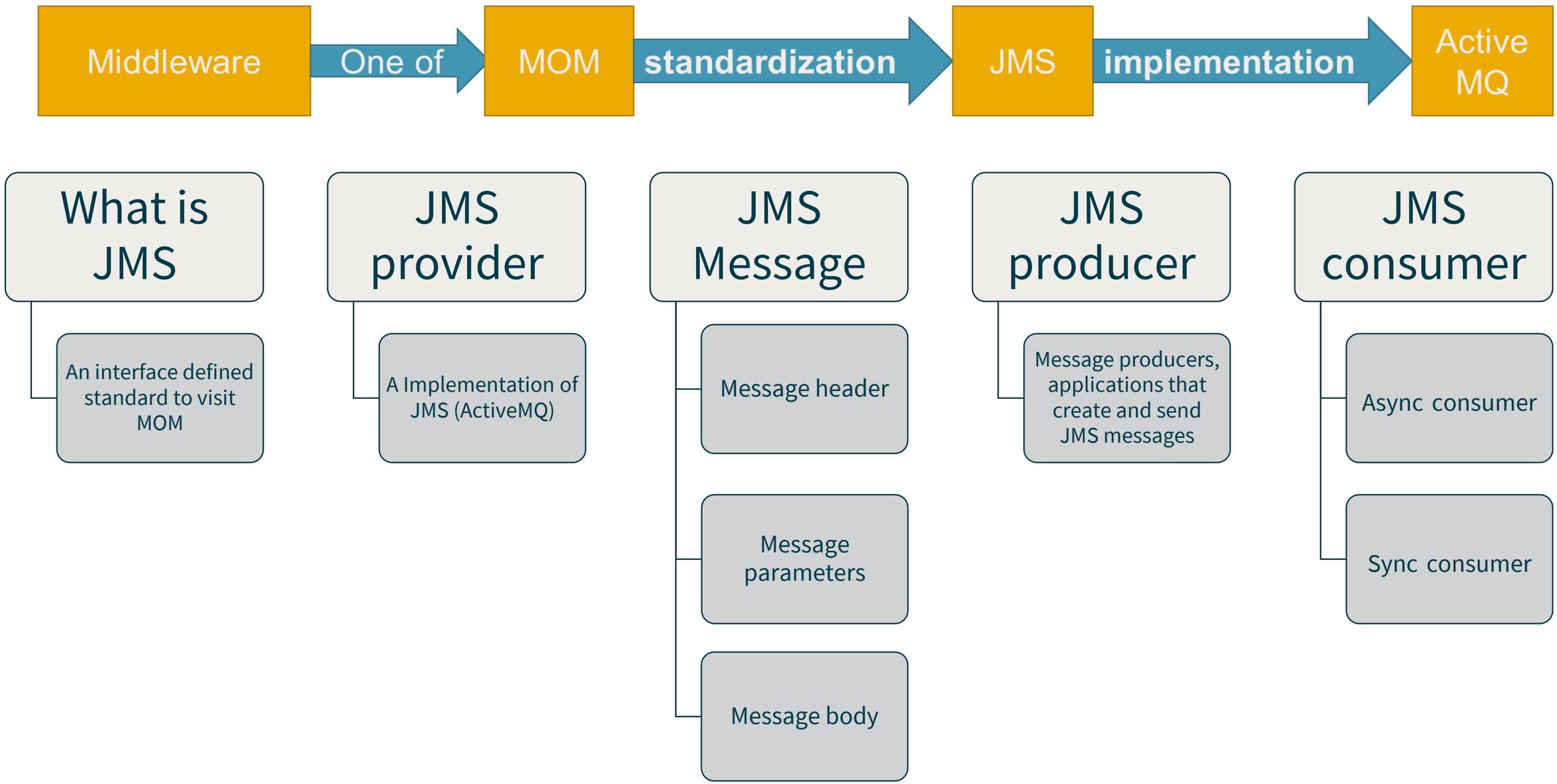



```
> Task :MqMessageReceiver.main()
message-0
message-1
message-2
```


KINGLAND

JMS Message

JMS (Java Message service) basic conception

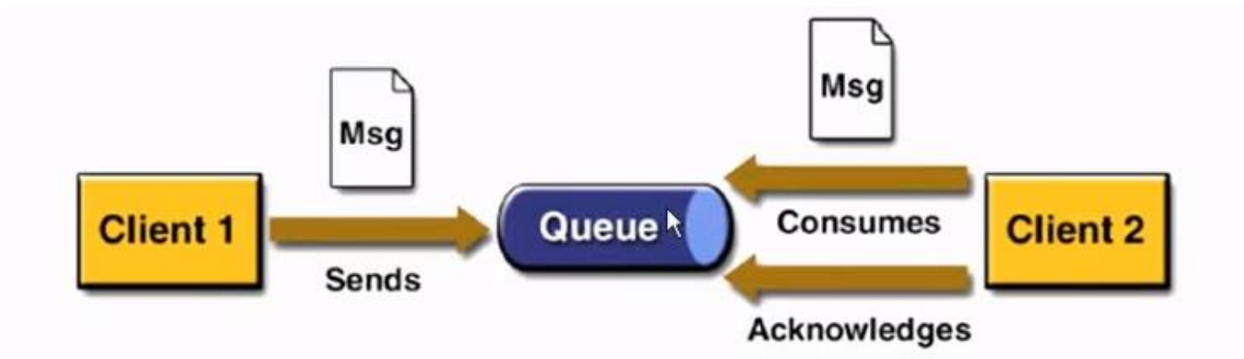


JMS basic conception

JMS domains

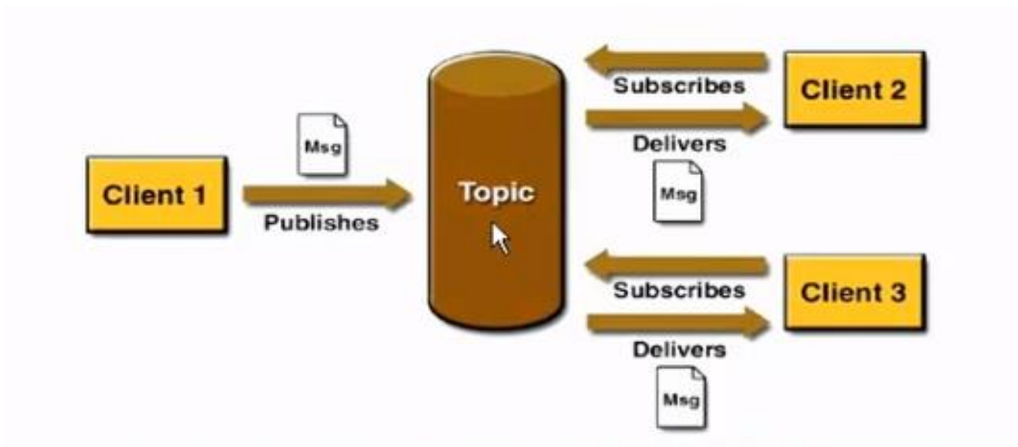
Point to point (PTP)

- Each message only has one consumer
- No temporal correlation between producers and consumers
- Destination is queue



Publish/subscribe (pub/sub)

- Each message has one or more consumers
- Has temporal correlation between producers and consumers, only after consumer subscribing the topic it would receive message from topic
- Destination is topic



KINGLAND

JMS basic conception

Connection factory

JMS Connection

JMS session

Destination

Acknowledge

Transaction

JMS client

JMS Message Header

Automatically assigned message headers	JMSDestitation **(send)**
	JMSDeliveryMode (PERSISTENT | NON_PERSISTENT) **(send)**
	JMSExpiration **(send)**
	JMSPriority **(send)**
	JMSRedeliverde **(Provider)**
	JMSMessageID **(Provider/client)**
	JMSTimestamp **(send)**
Developer assigned header	JMSCorrelationID **(client)**
	JMSReplayTo **(client)**
	JMSType **(client)**

https://docs.tibco.com/pub/ems/8.6.0/doc/html/GUID-A23051B9-8772-412C-8946-46CC1BBF39CE.html

JMS Message Parameters

JMS defined properties	JMSXUserID
	JMSXAppID
	JMSXProducerTXID
	JMSXConsumerTXID
	JMSXRcvTimestamp
	JMSXDeliveryCount
	JMSXState
	JMSXGroupID
	JMSXGroupSeq
Provider-specific properties	message.setStringProperty("username",username);
	message.setIntProperty("userId",1);

Message body types

- TextMessage
- StreamMessage
- MapMessage
- ObjectMessage
- BytesMessage

Integrate with Spring Boot

Environment set up

Quick start first demo

- **Add gradle dependencies in build.gradle**

```
dependencies {
    compile("org.springframework.boot:spring-boot-starter-web")
    compile('org.springframework.boot:spring-boot-starter-activemq')
    compile group: 'org.messaginghub', name: 'pooled-jms', version: '1.1.0'
    testCompile("org.springframework.boot:spring-boot-starter-test")
}
```

Quick start first demo

- **Add application properties**

```
spring.activemq.user=admin
spring.activemq.password=admin
spring.activemq.broker-url=tcp://127.0.0.1:61616
queue.name=myQueue
```

Quick start first demo

- **Use Configuration to define a Queue bean instance**

```java
package demo.config;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;

@Configuration
public class ActivemqConfig {
    @Value("${queue.name}")
    String queueName;
}
```

Quick start first demo

- **Provider class**

```java
package demo.controller;

import ...


@RestController
public class activemqController {
    @Autowired
    private Queue queue;

    @Autowired
    private JmsMessagingTemplate jmsMessagingTemplate;

    @RequestMapping("send")
    public void send(String name) {
        //方法一：添加消息到消息队列
        jmsMessagingTemplate.convertAndSend(queue, name);
        //方法二：这种方式不需要手动创建queue，系统会自行创建名为test的队列
        //jmsMessagingTemplate.convertAndSend("test", name);
    }
}
```

Quick start first demo

- **Consumer class**

```
@Component
public class QueueConsumer {
    @JmsListener(destination = "${queue.name}")
    public void receiveMsg(String text) {
        System.out.println("接收到消息 : "+text);
    }}
```

Try it now !

http://localhost:8080/send/queue?name=hello

Use Pub/Sub

- **Add application.properties**

```
spring.jms.pub-sub-domain=true
```

- **Create topic bean instance**

```
@Bean
public Topic topic() {
    return new ActiveMQTopic(topicName);
}
```

- **Add provier for Sub**

```
@RequestMapping("send/sub")
public void sendSub(String name) {
    //方法一：添加消息到消息队列
    jmsMessagingTemplate.convertAndSend(topic, name);
}
```

Use Pub/Sub

- **Add Consumer for sub**

```
@JmsListener(destination = "${topic.name}")
public void receiveTopic1(String text) {
    System.out.println("receiveTopic1接收到Topic消息 : " + text);
}

@JmsListener(destination = "${topic.name}")
public void receiveTopic2(String text) {
    System.out.println("receiveTopic2接收到Topic消息 : " + text);
}
```

Try it now !

http://localhost:8080/send/sub?name=hello

Support both Queue & Sub/Pub

- **Define QueueListenerFactory and topicListenerFactory**

```
@Bean("queueListenerFactory")
public JmsListenerContainerFactory<?> queueListenerFactory(ConnectionFactory connectionFactory) {
    DefaultJmsListenerContainerFactory factory = new DefaultJmsListenerContainerFactory();
    factory.setConnectionFactory(connectionFactory);
    factory.setPubSubDomain(false);
    return factory;
}
@Bean("topicListenerFactory")
public JmsListenerContainerFactory<?> topicListenerFactory(ConnectionFactory connectionFactory) {
    DefaultJmsListenerContainerFactory factory = new DefaultJmsListenerContainerFactory();
    factory.setConnectionFactory(connectionFactory);    //设置为发布订阅方式，默认情况下使用的生产消费者方式
    factory.setPubSubDomain(true);
    return factory;
}
```

Support both Queue & Sub/Pub

- **Use containerFactory in JmsListener**

```
public class QueueConsumer {
  @JmsListener(destination = "${queue.name}", containerFactory = "queueListenerFactory")
  public void receiveMsg(String text) { System.out.println("接收到消息 : "+text); }}
```

```
@JmsListener(destination = "${topic.name}", containerFactory = "topicListenerFactory")
public void receiveTopic1(String text) {
    System.out.println("receiveTopic1接收到Topic消息 : " + text);
}

@JmsListener(destination = "${topic.name}", containerFactory = "topicListenerFactory")
public void receiveTopic2(String text) {
    System.out.println("receiveTopic2接收到Topic消息 : " + text);
}
```

Kingland Systems. Discover Progress.

Our clients **know** that Kingland Systems delivers **faster, smarter,** more **reliable** solutions**.**

INDUSTRY SOLUTIONS

Kingland has been delivering Industry-specific solutions to leading global enterprises for more than 23 years.

SOLUTION PLATFORM

The Kingland Strategic Solution Platform means continuously smarter technology to deliver today and into the future.

EXPERT SERVICES

Kingland brings deep data and software expertise to every solution, helping you realize benefits swiftly — and with less risk.

KINGLAND

KINGLAND®

Thank You!

KINGLAND.COM