Java Message Service

- JMS lets you send messages containing data (for example a String, array of byte or a serializable Java object), from one program to another.
- It doesn't use a direct connection from Program A to Program B, instead the message is sent to a JMS provider and saves it where it waits the other program receives it.
- The Java Message Service (JMS) API is a Java Message Oriented Middle (MOM) API for sending messages between two or more clients that allow application to create, send, receive, and read messages.
- The JMS API enables communication that is loosely coupled, asynchronous and reliable.
- Apache ActiveMQ is a JMS provider which is lightweight and easy to use.
- Here MessageProducer is a Java Program sending a JMS Message to a Queue or Topic on the JMS Provider.
- MessageConsumer is another program which receives that message.
- These two programs can run on separate machines and all they have to know to communicate is the URL of the JMS Provider.

Maven POM

```
<dependency>
	<groupId>org.apache.activemq</groupId>
	<artifactId>activemq-all</artifactId>
	<version>5.15.4</version>
</dependency>
```

Producer Queue Example

```
public class JMSProducer {
	private static final String QUEUE_NAME = "Message";
	private static final String URL = "tcp://localhost:61616";
	public static void main(String[] args) throws JMSException, IOException {

		ConnectionFactory factory = new ActiveMQConnectionFactory(URL);
		System.out.println("Creating Connection factory object");

		Connection connection = factory.createConnection();
		System.out.println("Creating Connection via factory");

		Session session =
			connection.createSession(false,Session.AUTO_ACKNOWLEDGE);
		System.out.println("Creating Session via Connection");

		Destination destination = session.createQueue(QUEUE_NAME);
		System.out.println("Creating Queue via Session");

		TextMessage message = session.createTextMessage("Sending message to
							ActiveMQ as Queue");
		System.out.println("Creating TextMessage via Session");

		MessageProducer producer = session.createProducer(destination);
		System.out.println("Creating MessageProducer via session");

		producer.send(message);
		System.out.println("Sending the message the ActiveMQ using Produser");
		connection.close();
	}
}
```

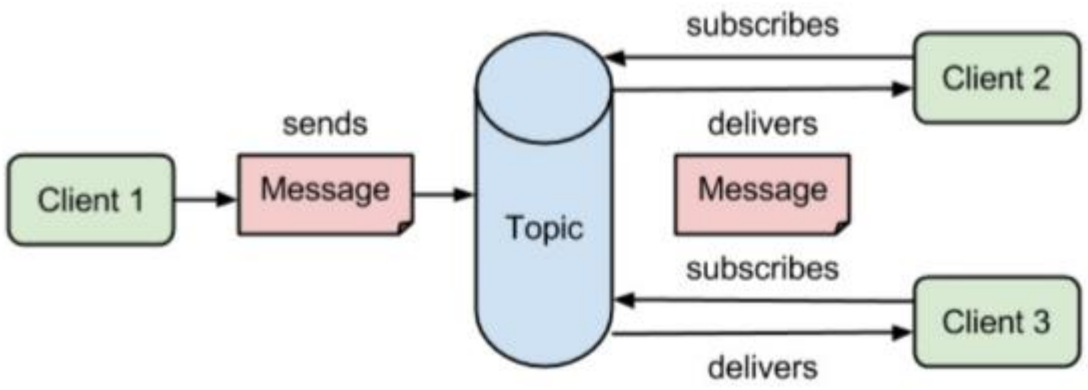
Consumer Queue Example

```
public class JMSConsumer {
	private static final String QUEUE_NAME = "Message";
	private static final String URL = "tcp://localhost:61616";

	public static void main(String[] args) throws JMSException, IOException {
		ConnectionFactory factory = new ActiveMQConnectionFactory(URL);
		System.out.println("Creating Connection factory object");

		Connection connection = factory.createConnection();
		connection.start();
		System.out.println("Creating Connection via factory");

		Session session =
				connection.createSession(false,Session.AUTO_ACKNOWLEDGE);
		System.out.println("Creating Session via Connection");
```

```
            Destination destination = session.createQueue(QUEUE_NAME);
            System.out.println("Creating Queue via Session");

            MessageConsumer consumer = session.createConsumer(destination);
            System.out.println("Creating Queue using Queue");

            Message message = consumer.receive();
            System.out.println("Reciving Message from the Queue");

            TextMessage textMessage = (TextMessage)message;

            System.out.println(textMessage.getText());
            session.close();
            connection.close();
        }
}
```

JMS Publisher-Subscriber (pub-sub)

In a pub/sub product or application, clients address messages to a topic, which functions somewhat like a bulletin board. Subscribers can receive information, in the form of messages, from publishers. Topics retain messages only as long as it takes to distribute them to current subscribers.



Pub/Sub messaging characteristics

- Each message can have multiple consumers.
- Publishers and subscribers have a timing dependency. A client that subscribes to a topic can consume only messages published after the client has created a subscription, and the subscriber must continue to be active in order for it to consume messages.

Producer Topic Example

```
public class JMSProducer {
	private static final String TOPIC_NAME = "Message";
	private static final String URL = "tcp://localhost:61616";
	public static void main(String[] args) throws JMSException, IOException {

		ConnectionFactory factory = new ActiveMQConnectionFactory(URL);
		System.out.println("Creating Connection factory object");

		Connection connection = factory.createConnection();
		System.out.println("Creating Connection via factory");

		Session session =
			connection.createSession(false,Session.AUTO_ACKNOWLEDGE);
		System.out.println("Creating Session via Connection");

		Destination destination = session.createTopic(TOPIC_NAME);
		System.out.println("Creating Topic via Session");

		TextMessage message = session.createTextMessage("Sending message to
							ActiveMQ as Topic");
		System.out.println("Creating TextMessage via Session");

		MessageProducer producer = session.createProducer(destination);
		System.out.println("Creating MessageProducer via session");

		producer.send(message);
		System.out.println("Sending the message the ActiveMQ using Produser");
		connection.close();
	}
}
```

Consumer TOPIC Example

```
public class JMSConsumer {
	private static final String TOPIC_NAME = "Message";
	private static final String URL = "tcp://localhost:61616";

	public static void main(String[] args) throws JMSException, IOException {
		ConnectionFactory factory = new ActiveMQConnectionFactory(URL);
		System.out.println("Creating Connection factory object");

		Connection connection = factory.createConnection();
		connection.start();
		System.out.println("Creating Connection via factory");

		Session session =
				connection.createSession(false,Session.AUTO_ACKNOWLEDGE);
		System.out.println("Creating Session via Connection");

		Destination destination = session.createTopic(TOPIC_NAME);
		System.out.println("Creating Topic via Session");

		MessageConsumer consumer = session.createConsumer(destination);
		System.out.println("Creating Queue using Queue");

		Message message = consumer.receive();
		System.out.println("Reciving Message from the Queue");

		TextMessage textMessage = (TextMessage)message;

		System.out.println(textMessage.getText());
		session.close();
		connection.close();
	}
}
```