Java Message Service (JMS)

보조자료

2009.10.6.

yiyoon@sm.ac.kr

Message Queue ?

- Queue란 FIFO(First In First Out) 구조의 저장소로 한쪽이 입구이고, 한쪽이 출구인 원통형 데이터 저장소이다. 메시지 큐라는 것은, 큐의 구조이긴 한데, 메시지를 저장하는 저장소를 이야기한다.

- 모든 응용 프로그램은 어떤 일을 하기 위해 운영체제에게 메시지를 보내는데 윈도우즈는 메시지 방식의 운영체제이다. 예를 들어 파일을 찾는 프로그램이 있다면, 파일 찾기 프로그램은 윈도우에게 파일을 찾아달라는 메시지를 보낸다. 이때 찾고자 하는 파일 이름을 같이 보내는데 그러면 윈도우즈는 파일 찾기 프로그램에게 그 파일이 어떤 디렉토리에 있다는 메시지를 보내주게 된다. 이런 식으로 응용 프로그램과 윈도우즈는 서로 메시지를 사용해서 의사소통을 하게 되는 것이다. 그리고 사용자 입력(마우스 입력, 키보드 입력)도, 응용프로그램이 먼저 받는 것이 아니라 윈도우즈가 먼저 받고서는 현재 가장 위에 떠있는 프로그램에게 사용자로부터 입력이 있었다는 메시지를 보낸다. 이런 메시지를 받아서 응용 프로그램은 처리를 하게 되는 것이다

- 이런 메시지를 쓰는 이유는 윈도우즈는 멀티태스킹, 즉 여러 프로그램이 동시에 돌아가는 것을 보장하기 위해서, 모든 자원을 윈도우즈가 관리하기 때문이다.그리고 어떤 프로그램에게서 요청이 오면 그 프로그램에게 자원을 넘겨주는 것이다. 이렇게 다른 프로그램이 어떤 자원을 쓰고 있을 때, 또 다른 프로그램이 그 자원을 요구하면, 윈도우즈는 잠시 기다리게 하고 다 쓰여진 자원을 윈도우즈가 돌려받아 또 다른 프로그램에게 던져준다.
- 이렇듯, 윈도우즈하의 모든 프로그램은 메시지를 기반으로 요청/응답을 하도록 되어 있다.
- 만약 어떤 프로그램에서 A라는 메시지를 처리하고 있는 중간에, 다른 메시지가 도착할 수 있는 상황이 발생하면 메시지를 즉시 처리하지 못하기 때문에 잠깐 저장해 두어야 한다.
- 이때 메시지를 저장하는 곳이 메시지 큐이다.
- 메시지 큐에 차곡차곡 쌓인 메시지들은, 순서적으로 프로그램에 의해서 처리되는 것이다.

Message System

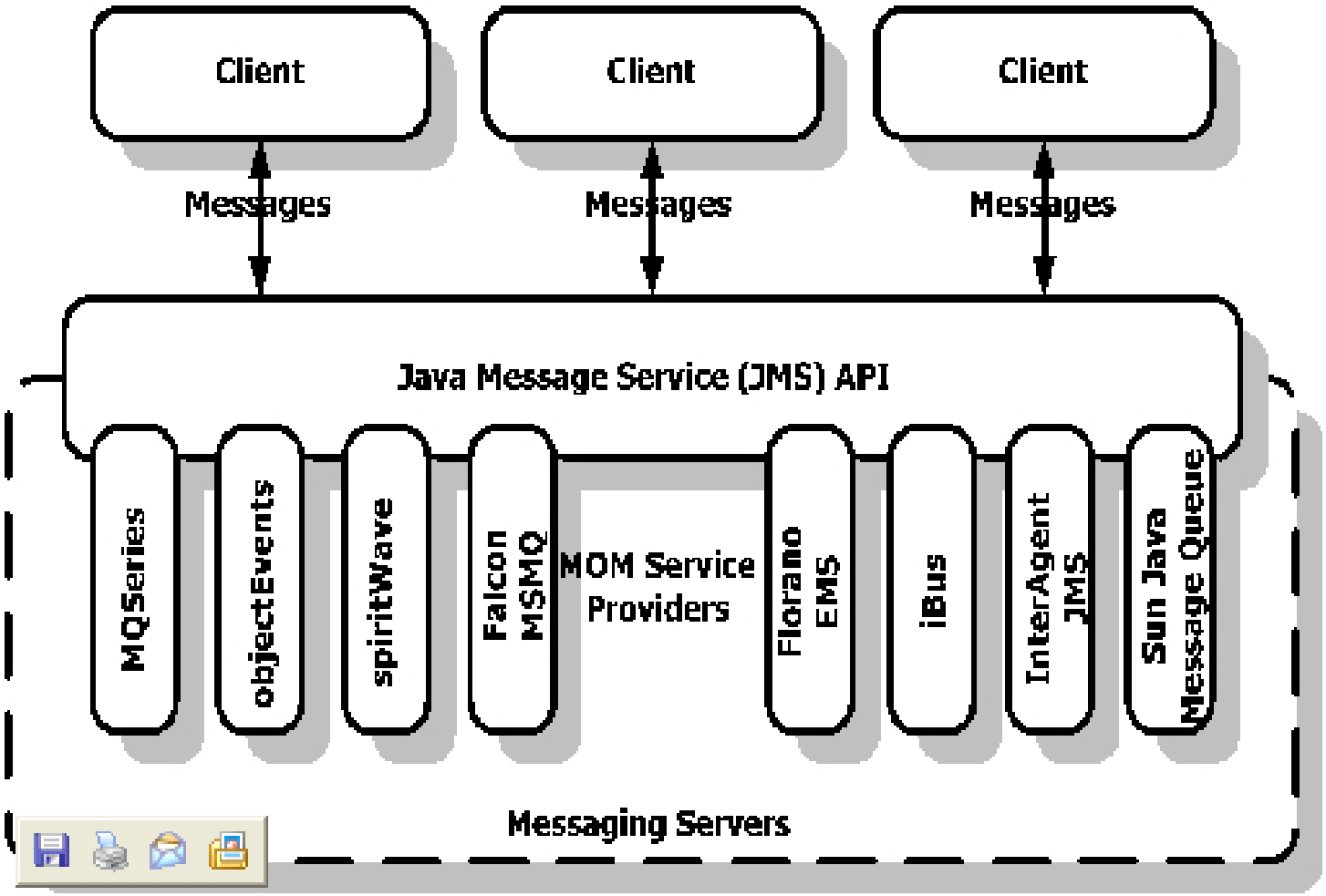
◉ Message System

Message System이란, Application과 Application이 서로 통신을 하도록 지원해주는 시스템 이다.

- 일반적으로 Message System이 만들어지기 전에는 이런 업무를 A와 B 시스템간에 Socket을 직접 연결해서 Packet을 정의하고 그 Packet에 따라서 통신을 했다. 이 통신을 위해서, Packet에 대한 flow control이나, 네트워크에 문제가 발생 했을 때의 예외처리 등을 다 **직접 프로그래밍을** 해야 했다. 그러나 Message System은 이런 모든 AP간의 통신에 대한 여러 기능들을 제공하여 통신에 대한 부분을 간결화 시켜준다

- Message System에는 이외에도, 하나의 Publisher(방송者)가 여러 Subscriber(구독者) 에게 메세지를 전송하는 모델이라 던지, P2P모델 그리고, 메세지가 중간에 유실되지 않게 하는 Reliable Messaging, **비동기 방식으로 메세지를 전달하는** 방법, 분산 트렌젝션, 클러스터링 등을 지원한다.

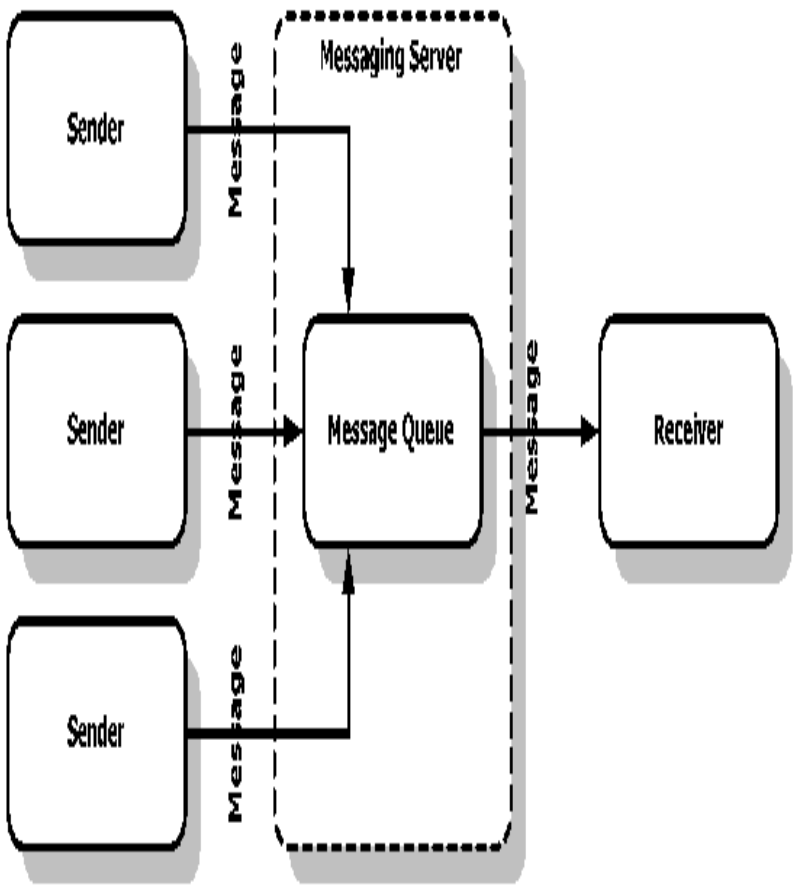
Message System의 구조



[Message System의 architecture]

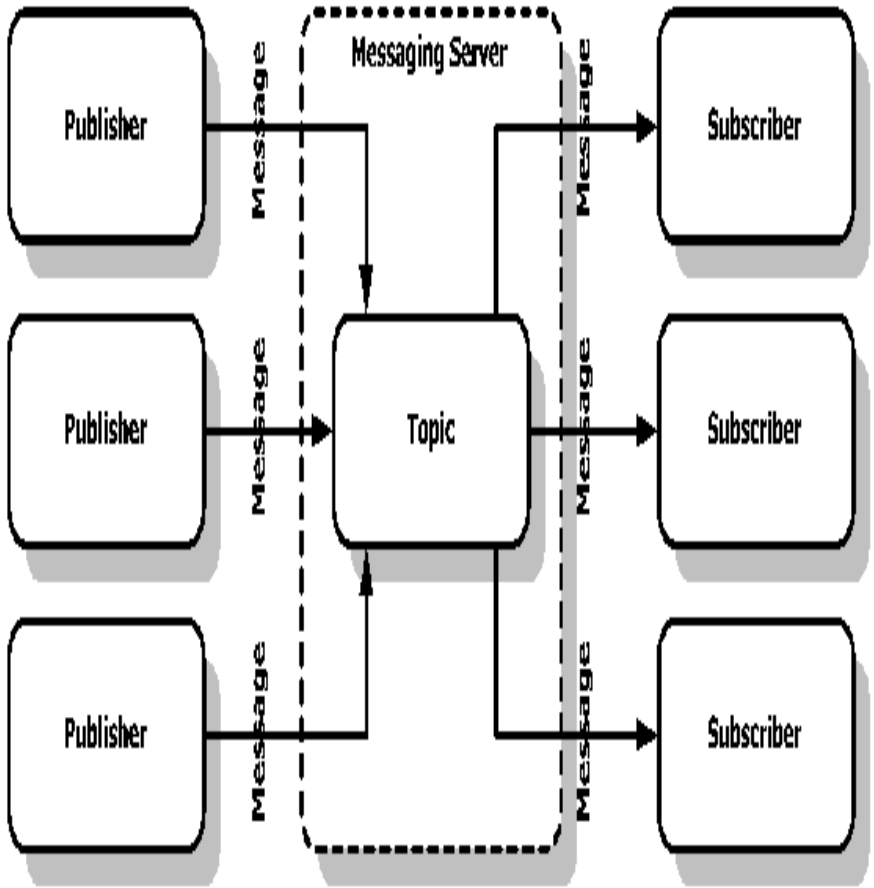
Message System 방식

1. PTP(Peer-To-Peer)방



[PTP Message System Architecture]

2. Pub/Sub (Publisher-Subscriber)방식



[Sub/Pub Message System Architecture]

두 방식의 비교

	P2P	Pub/Sub
저장소	Queue	Topic
저장대상	message	client
메시지전달	-client들이 Queue에 message를 전송 해 두면, -다른 client가 능동적으로 message를 가져간다.	-client가 message를 출판하면(Event), -Topic에 등록된 client에게 message를 배달한다.

Java Message Service

- JMS는 Java Message Service의 약자로 Java에서 Message System을 사용하기 위한 API들을 제공한다.

- JMS API는 엔터프라이즈 메시징 제품을 사용하기 위한 **공통의 API**를 제공한다. 즉 공통의 인터페이스가 있고, 실제 구현은 서비스 제공자가 제공하도록 되어있어 어떤 제품을 사용하든지 사용하는 방법이 같다.

- JMS를 사용하기 위해 반드시 필요한 사항은 다음과 같다.
  - Connection Factory: JMS서비스 제공자로의 Connection을 생성을 한다.
  - Destination: 메시지를 보내고, 혹은 받고자 할 때 사용하는 목적지이다.
  - Message: 정보를 주고받을 때 사용된다.

- JMS Message는 다음과 같은 형태로 구성이 된다.
  - Header: 메시지를 식별하고, JMS서비스 제공자와 클라이언트 정보가 들어있다.
  - Properties: 추가적인 헤더 정보를 넣고 싶을 때 사용을 한다.
  - Body: 여러 가지 타입의 데이타를 저장할 때 사용을 한다.

JMS (Java Message Service)

♣JMS란 무엇인가?

자바 프로그램이 messaging system을 통한 메시지를 생성, 보내기, 받기, 읽기 위한 공통의 방법(API)을 제공한다.

- Messaging System : Message Oriented Middleware (MOM)
- 자바 언어로 된 클라이언트들과 자바 언어의 개발된 메시징 시스템을 이용할 수 있어야 하는데, 이때 JMS가 이러한 시스템에 접근하기 위한 공통 방식을 정의하고 있다.
- JMS는 인터페이스와 JMS 클라이언트가 메시징의 facilities에 접근하는 방식을 정의한 associated semantics를 가지고 있다.
- 메시징은 peer-to-peer임으로 JMS의 모든 사용자는 일반적으로 클라이언트로 생각하면 된다. JMS 애플리케이션은 애플리케이션에서 정의한 메시지와 그것을 주고 받는 클라이언트로 구성된다. JMS를 구현하고 있는 상품들은 JMS 인터페이스를 구현한 provider를 제공한다.

☞ 여기서 메시지란 (사람이 아닌) 엔터프라이즈 애플리케이션 간의 비동기 커뮤니케이션을 말한다.

What is JMS

- JMS는 비동기적 의미를 내포하고 있는 UDP 통신개념의 Mail System과 유사하지만 **Component 간 통신이라는** 점에서 다르다.

- JMS의 비동기적 메세지 전송은 **MOM (Message Oriented Middleware)**이 중간에서 Message를 받아두었다가 Receiver가 접속을 하면 Message를 전달해 주는 방식으로 이루어진다.

- Messaging System의 경우 JMS API뿐 아니라 C와 같은 Non-Java AP를 위한 Lib Code를 제공하는 경우가 있는 데 이런 경우에는 Java Application이 아닌 다른 언어로 개발된 Application과도 호환이 되기 때문에, JMS기반의 Messaging 시스템은 Main frame이나 **다른 Application들과 Java Application을 연동하는데** 많이 사용이 되며, 근래에 많이 출시되는 EAI (Enterprise Application Integration) 솔루션도 JMS를 이용하는 경우가 많다.

- JMS API를 이용하기 위해서는 JMS명세를 알아야 하는데, JMS 명세에 따르면 JMS를 이용하는 응용 프로그램은 **여러 가지 요소로 구성된다**. 각각의 요소는 API에서 제공하는 객체가 될 수도 있으며 어떤 부분은 응용 프로그램일 수 있다.

JMS application 구성요소

- JMS Clients - These are the Java language programs that send and receive messages.

- Non-JMS Clients - These are clients that use a message system's native client API instead of JMS. If the application predated the availability of JMS it is likely that it will include both JMS and non-JMS clients.

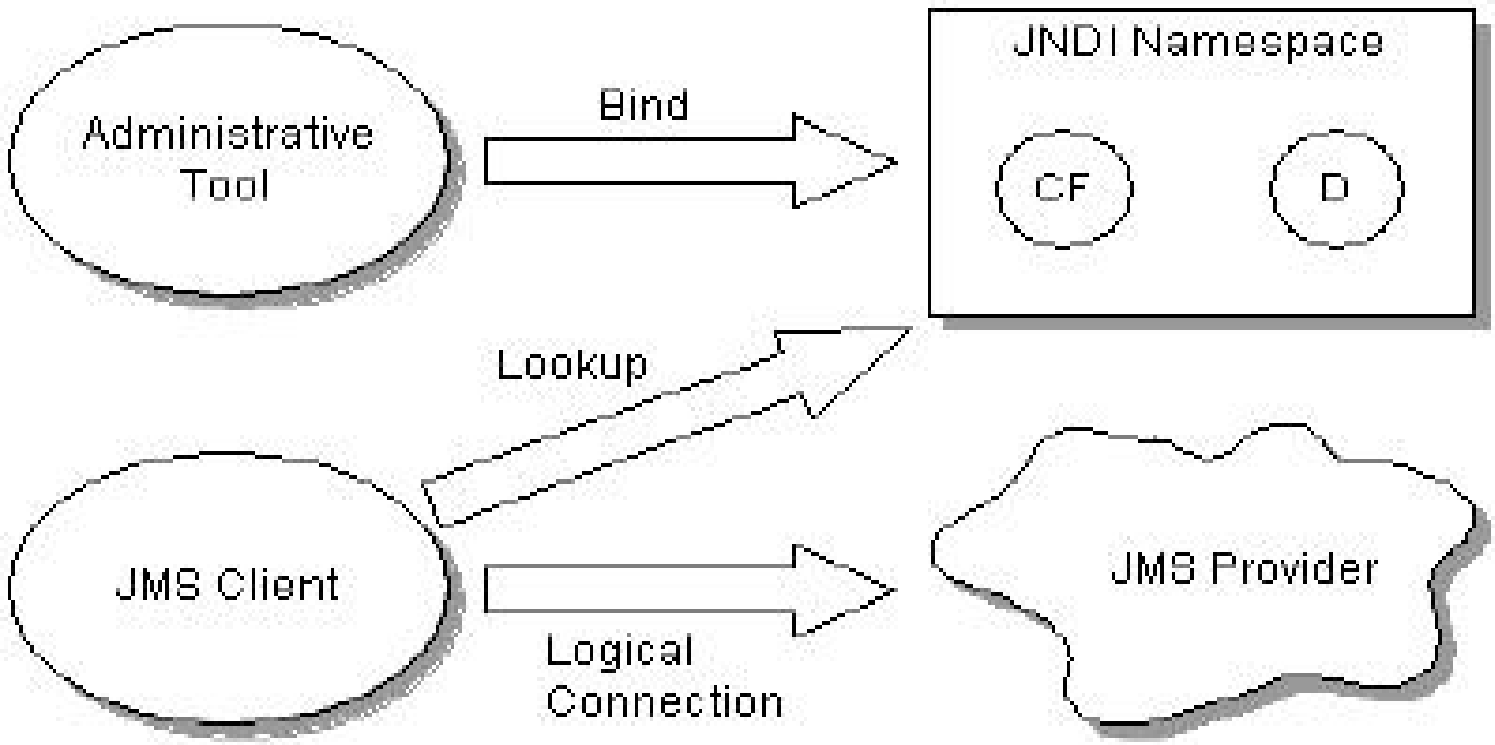
- Messages - Each application defines a set of messages that are used to communicate information between its clients.

- JMS Provider - This is a messaging system that implements JMS in addition to the other administrative and control functionality required of a full featured messaging product.

- Administered Objects - Administered objects are preconfigured JMS objects created by an administrator for the use of clients. There are two types of JMS administered objects:
  - ConnectionFactory - This is the object a client uses to create a connection with a provider.
  - Destination - This is the object a client uses to specify the destination of messages it is sending and the source of messages it receives.

JMS 구조

Figure 2-1 JMS Administration

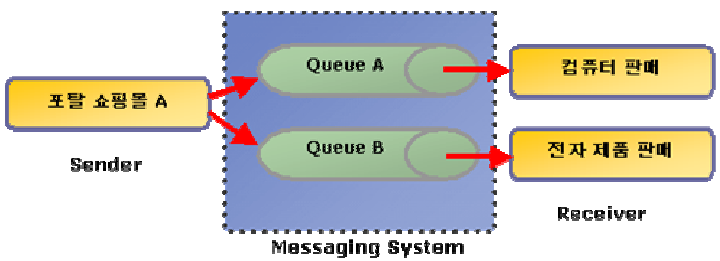


JMS의 Queue 개념

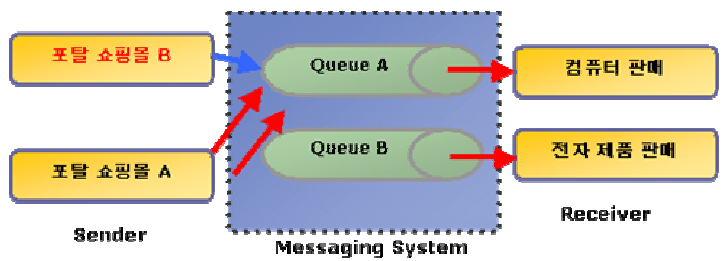
- JMS Messaging 시스템에서 sender가 메세지를 보내는 destination이 되고, reciever가 메세지를 읽어오게 되는 부분을 Queue 또는 Bus라고 한다.

- 포탈 쇼핑몰에 입점한 여러 개의 쇼핑몰이 있다고 하자.

- 포탈쇼핑몰에서는
  - 주문 내용을 메세지로 만들어서 보내고, 이 메세지를 컴퓨터에 대한 주문은 QueueA로,
  - 전자제품에 대한 주문은 QueueB로 보낸다고 한다.
- 컴퓨터 판매 쇼핑몰은 QueueA에서만 주문 정보를 받고, 전자제품 쇼핑몰을 QueueB에서 주문을 받는다.
- Queue는 이처럼 Sender와 Receiver간에 Message를 주고받는 채널의 역할을 한다.



- 여기서 만약 포탈 쇼핑몰이 하나 더 늘어났다고 하자. 그러면 주문 연동은 어떻게 할것인가?
  - 답은 간단하다. 새로운 포탈 쇼핑몰 B가 컴퓨터 판매 쇼핑몰에 주문을 넘기기 위해서는 Queue A에 주문 Message를 보내기만 하면 된다.



Apache Active MQ

- ActiveMQ
  - It is an open source (Apache 2.0 licensed) message broker which fully implements the Java Message Service 1.1 (JMS).
  - It provides "Enterprise Features"[1] like clustering, multiple message stores, and availability to use any DB as a JMS persistence provider besides VM, cache, and journal persistency.

- The Advanced Message Queuing Protocol (AMQP)
  - It is an attempt to define behavior of the messaging server and client such that implementations are truly interoperable.
  - The flexibility of this protocol to define installations with point to point messaging, publish/subscribe messaging, or combinations of both, is extremely powerful and may become the standard that is needed in this domain.
  - The protocols which can be claimed to be a standard that are used for Message-oriented middleware include XMPP and Streaming Text Orientated Messaging Protocol.

ActiveMQ

- ActiveMQ는 애플리케이션 간의 데이터를 공유하는데 사용되는 자바 기반 오픈 소스 메시징 소프트웨어다. 메시지 중심 빈(MDB)의 오픈 소스 Java Message Service(JMS) 애플리케이션 공급자와 지원자이다.
- Apache ActiveMQ는 가장 대중적이고 강력한 오픈소스 메시지 브로커이다.

  Apache ActiveMQ는 빠르고, 많은 Cross Language Clients와 프로토콜, JMS 1.1 과 J2EE 1.4를 완전히 지원하는 동안 많은 이점을 지원한다.
- Apache ActiveMQ는 JMS 클라이언트와 함께 자바로 개발된 메시지 브로커이다. 그러나 Apache ActiveMQ는 다른 언어의 클라이언트를 지원하면서 Stomp와 OpenWire같은 프로토콜과 통신되도록 설계되었다.
- ActiveMQ는 TCP, SSL, UDP, multicast, intra-JVM, NIO 같은 다양한 트랜스포트와 push, pull, publish/subscribe 같은 클라이언트 상호작용을 지원한다. ActiveMQ 서버는 어떤 콘테이너(예로 J2EE)로부터 완벽하게 독립되어 동작할 수 있다.

ActiveMQ 란 무엇인가?

♣ Apache ActiveMQ

완전한 JMS 클라이언트와 더불어 Java로 쓰인 message broker이다.

- Message Broker란?

  Message broker는 프로그램들이 형식적으로 지정된 메시지에 의해 네트워크 상에서 sender로부터의 메시징 프로토콜을 receiver에게 번역하는 중개 프로그램이다.
- ActiveMQ

  ActiveMQ는 많은 프로토콜과 플랫폼을 연결되어 동작이 가능하다. (그림 ◀)

  즉, Net이나 C/C++또는 Perl, Python, PHP와 Ruby와 같은 스크립팅 언어 에도 사용되어질 수 있다.

- Apache ActiveMQ가 제공하는 Language Clients:

  ☞ Ajax, C, C++, C# and .Net, Delphi, Flash / ActionScript, JavaScript, Perl, PHP, Pike, Python, Ruby and Rails support via ActiveMessaging, Smalltalk
- Apache ActiveMQ 가 제공하는 프로토콜:.

  ☞ OpenWire, REST, Stomp, W S Notification, XMPP, AMQP
- ActiveMQ의 특징; http://activemq.apache.org/features.html
- ActiveMQ의 최근버젼; http://activemq.apache.org/activemq-411-release.html

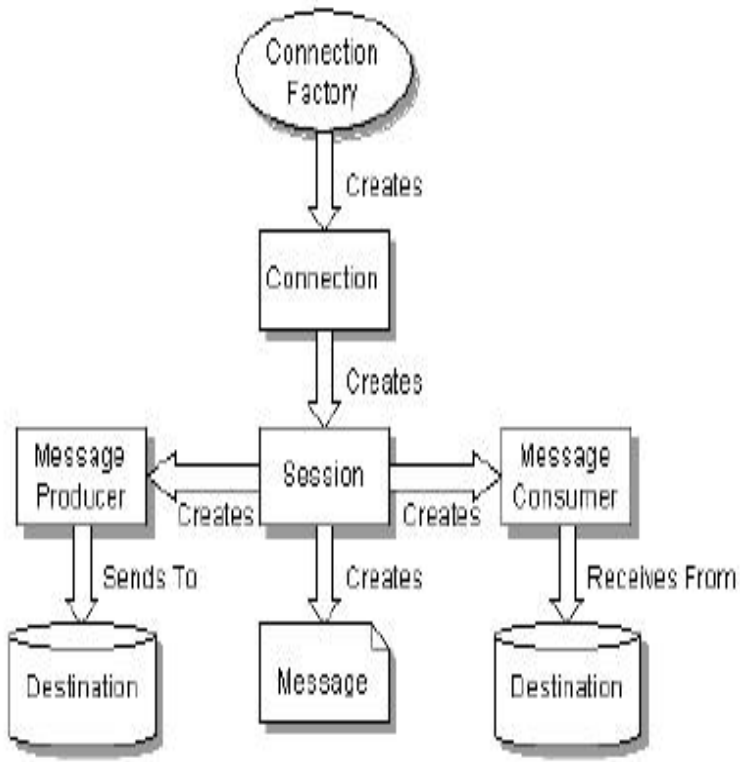
Two Messaging Styles

- JMS Point-to-Point Model - Queue
- JMS Publish/Subscribe Model - Topic

-JMS 오브젝트간 관계

- ConnectionFactory - an administered object used by a client to create a Connection
- Connection - an active connection to a JMS provider
- Destination - an administered object that encapsulates the identity of a message destination
- Session - a single-threaded context for sending and receiving messages
- MessageProducer: an object created by a Session that is used for sending messages to a destination
- MessageConsumer:an object created by a Session that is used for receiving messages sent to a destination

Figure 2-1 Overview of JMS object relationships



ActiveMQ 테스트

- JMS API는 엔터프라이즈 메시징 제품을 사용하기 위한 공통의 API를 제공한다. 즉 공통의 인터페이스가 있고, 실제 구현은 서비스 제공자가 제공하도록 되어있다. 즉 어떤 제품을 사용하든지 사용하는 방법이 같다.

- JMS 를 사용하기 위해 반드시 필요한 사항은 다음과 같다.
  - ConnectionFactory : JMS서비스 제공자로의 Connection을 생성을 한다.
  - Destination : 메세지를 보내고, 혹은 받고자 할때 사용하는 목적지이다.
  - Message : 정보를 주고 받을때 사용된다.

- MSMessage는 다음과 같은 형태로 구성이 된다.
  - Header : 메세지를 식별하고, JMS 서비스 제공자와 클라이언트 정보가 들어있다.
  - Properties : 추가적인 헤더 정보를 넣고 싶을때 사용을 한다.
  - Body : 여러가지 타입의 데이타를 저장할때 사용을 한다.

- Cf) Spring은 JMS API의 사용을 단순화하고 JMS 1.0.2와 1.1 API사이의 차이점으로부터 사용자를 감싸는 JMS 추상 프레임워크를 제공한다.

Active MQ를 통한 JMS의 사용 예

1. ConnectionFactory 정의

MQ서버에 접속할 연결정보를 생성해야한다. Broker`URL로 MQ에 연결할 주소 즉 IP와 Port 정보를 넘겨주면 된다.

tcp://l203.252.201.16:61616

JmsTemplate에서는 연결정보를 가진 connetionFactory를 넘겨주어야 한다.

2. TestController 파라미터 추가

샘플로 TestController에서 MQ를 컨트롤 하기 위한 jmsTemplate를 파라미터로 넘겼다.

3. Send Massage

- 넘겨주는 JmsTemplate를 받기 위해 setter를 생성을 한다. Session은 메세지를 보내고 수신하는 단일 쓰레드 컨텍스트이다. JMS에서는 Message Producer라는 개념이 있다. 역할은 메세지를 목적지로 보내는 역할을 하는데, Spring에서 사용할 때는 JmsTemplate 내에 개념이 들어있어, 여기서는 사용하지 않아도 된다.

```
public class TestController extends AbstractController {
        protected JmsTemplate jms;

        public void setJms(JmsTemplate jms) {
           this.jms = jms;
        }
        protected ModelAndView handleRequestInternal
                (HttpServletRequest request, HttpServletResponse response) {

                 jms.send("dest", new MessageCreator(){
                     public Message createMessage(Session session) throws JMSException
    {
                     return session.createTextMessage("Testing");
                    }

                }
                return new ModelAndView(view, "movie", movie);
         }
}
```

- JMSMessage의 경우 내부적으로 데이타를 분류를 할 수 있다. 즉 A라는 곳에 저장하고, B라는 곳에 따로 저장이 되서, A와 B의 독립적으로 큐의 사용이 가능해진다. 이런 A, B 라는 분류가 Destination이 된다, 위의 예에서는 "dest"라고 명시를 하였다.

4. Receive Massage

- Message를 보낼 때와 반대로 JMS에서는 Message Consumer라는 개념이 있다. 역할은 메세지를 목적지에서 받을 때 사용을 하는데, 마찬가지로 Spring에서 사용할 때는 JMSTemplate 내에 개념이 들어있어, 여기서는 사용하지 않아도 된다.

```
try {
        Message msg= jms.receive("dest");
        if (msg instanceof TextMessage)
        {
            TextMessage txtMsg = (TextMessage) msg;
            System.out.println("Received: " + txtMsg.getText());
        }
    } catch (JMSException e) {
      e.printStackTrace();
}
```

- Send 할 때 보냈던 텍스트를 Receive 할 때 받을 수 있다.

- JMS sample directory
  - /sun/MessageQueue/demo/helloworldmessage/*.java

→실행시 고려점

→환경변수 설정

→Set CLASSPATH=%IMQ_HOME%₩lib₩jms.jar;%IMQ_HOME%₩lib₩imq.jar;.