IDL/SSO를 이용한 Java Message Service의 개발†

정명희° · 문남두 · 안건태 · 김현규 · 이명준
울산대학교 컴퓨터 · 정보통신공학부

Development of Java Message Service using IDL/SSO

Myung-Hee Jung° · Nam-Doo Moon · Geon-Tae Ahn · Hyun-Gyu Kim · Myung-Joon Lee
School of Computer Engineering · Information Technology, Univ. of Ulsan.

요 약

JMS(Java Message Service)는 분산된 엔터프라이즈 시스템간의 메시지 생성, 전달, 수신 및 읽기를 위한 자바 기반의 표준 API를 제공한다. 이 API의 구조는 메시징 방식에 따라 Topic을 메시지 서버로 하는 Publish-Subscribe 모델과 Queue를 서버로 가지는 Point-To-Point 모델로 구분된다.

IDL/SSO는 병행성(Concurrency), 영속성(Persistency), 필터(filter)등의 기능을 제공하는 공유객체 명세언어 시스템이다. 본 논문에서는 이 IDL/SSO를 이용하여 JMS를 구현하는 방법에 대하여 기술한다.

1. 서 론

엔터프라이즈 메시징은 엔터프라이즈 어플리케이션을 개발하고 설계하는데 있어서 필수적인 도구로 인식되고 있다. 엔터프라이즈 메시징은 중요한 비즈니스 데이터의 비동기적인 교환을 위하여 신뢰성 있고, 유연한 서비스를 제공하는 것이 바람직하다. JMS는 메시징 시스템의 접근을 위한 API이며, 엔터프라이즈 컴퓨팅 문제들을 해결하기 위하여 자바 기술을 결합시킴으로써 새롭고 강력한 도구를 제공한다. JMS는 엔터프라이즈 메시징을 위하여 일반화된 API의 명세만을 제공하며, 이 명세를 기반으로 개발자들이 여러 가지 방법으로 구현함으로써 어플리케이션간의 메시징 서비스를 제공하게 된다.[1]

본 논문에서는 다양한 분산환경을 지원하는 공유객체 명세 언어 IDL/SSO(IDL for Specifying Shared Objects)를 이용하여 JMS의 기본 API를 이것의 명세대로 재정의 하고 구현코드를 삽입(Code Embedding)함으로써 메시지 서비스를 제공하고자 한다.

IDL/SSO는 CORBA와 JavaRMI를 비롯한 다양한 분산환경에서의 병행성(Concurrency), 영속성(Persistency), 필터(filter)등의 기능을 체계적으로 지원하기 위한 명세언어이다.[2] 이것은 CORBA의 명세언어인 IDL 에 위 기능을 정의 하기 위한 구문을 포함한 형태로 구성되었다. IDL/SSO가 정의하는 이러한 기능들은 분산시스템 플랫폼 환경에서 서버의 구현에 필수적인 요소이며, 서버의 안정성과 효율성을 보장한다. 이 IDL/SSO는 이러한 메시지 서비스에 있어서 필수적인 요소인 동기화 부분 구현을 보다 쉽고 간단하게 해준다.

본 논문에서는 메시징 방식에 따라 Publish-Subscribe (Topic) 모델과 Point-To-Point(Queue) 모델 두 가지로 구분하여 IDL/SSO의 명세를 따르는 Topic과 Queue를 구현하고자 한다.

†본 연구는 정보통신연구진흥원의 '99년도 정보통신 우수시범학교 지원 사업에 의하여 부분적으로 지원되었음.

본 논문의 구성은 다음과 같다. 2장에서 IDL/SSO에 대하여 간단히 기술하고, 3장에서 JMS 인터페이스 구조와 각각의 기능에 대하여 살펴보고, 4장에서는 3장의 API 구조에 따른 IDL/SSO상의 JMS 구현을 소개하며, 5장에서는 결론을 기술한다.

2. IDL/SSO

IDL/SSO는 다양한 분산환경을 지원하는 공유객체 명세언어이다. 분산시스템의 구현에 있어서 서버의 안정성과 효율성은 필수적인 속성으로 인식되고 있으며, 이를 지원하기 위하여 병행성, 영속성, 필터 등의 기능이 구현되어야 한다. 그러나 서버구현에 필요한 이들 기능은 난이도가 높은 구현기법으로서, 임기응변식의 접근방법은 서버의 구현을 더욱 어렵게 할 수 있다.

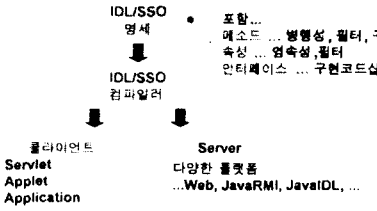
본 장에서는 CORBA와 JavaRMI를 비롯한 다양한 분산 환경에서 이들의 기능을 보다 체계적으로 지원하기 위한 명세 언어인 IDL/SSO (IDL for Specifying Shared Objects)에 대하여 간단히 소개한다.

IDL/SSO는 공유객체(Shared Object)의 개념으로부터 위에서 제시한 기능을 서버 객체의 인터페이스에서 논리적으로 정의할 수 있도록 지원한다. 이로부터 IDL/SSO는 CORBA의 명세 언어인 IDL에 위 기능을 정의하기 위한 구문을 포함한 형태로 구성되었다.[5] IDL/SSO로 정의된 명세는 IDL/SSO 컴파일러를 통해 Java 구현 코드로 변환되며, 이는 지역(Local) 환경, Sun사의 분산 시스템 환경인 JavaRMI, 그리고 Sun사의 CORBA지원 환경인 JavaIDL등의 다양한 플랫폼을 지원할 수 있도록 구성된다.[6]

IDL/SSO는 서버 객체 인터페이스상에 구현코드를 바로 삽입(Code Embedding)할 수 있도록 하는 기능을 지원하여 분산시스템의 세부구조에 관계없이 논리적인 구현을 할 수 있으며, 시스템의 구조 파악 역시 쉽게 할 수 있다. 뿐만 아니라, 전처리 과정(Pre-processor)이나 환경변수 설정 등 시스템 구현에 필요한 부가적인 중간 단계를 개발자로부터 보이지 않게 추상화시키는 역할도

제공하여 개발자의 할 일을 단축시킬 수 있다.[2]

[그림 1]에서는 IDL/SSO의 정의로부터 생성되는 클라이언트/서버의 구조를 보이고 있다.

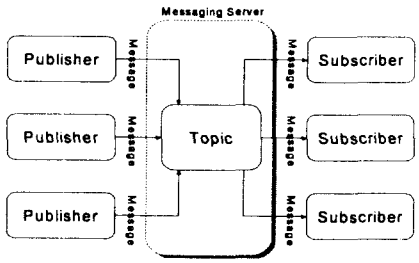


[그림 1] IDL/SSO의 기능과 생성구조

3. JMS API의 구조

메시징 시스템의 일반적인 모델은 메시지 서비스에 따라 3가지로 구분된다. Publish-Subscribe 모델, Point-To-Point 모델과 Request-Reply 모델로 나뉜다. 이 중에서 JMS API는 앞의 두 모델만을 정의하고 있으며, 첫 번째 모델은 Topic, 두 번째 모델은 Queue가 주요 개념이 된다. 그럼 이 JMS API에 의한 이 두 모델을 살펴보자.

3.1 Topic (Publish-Subscribe 모델)



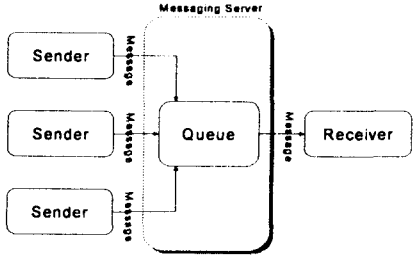
[그림 2] Publish-Subscribe Messaging

이 모델은 다중 어플리케이션이 같은 메시지를 받을 필요가 있을 때 사용된다. 다중 Publisher들은 Topic으로 메시지를 전달하고 모든 Subscriber들이 이 메시지를 전달받게 된다. [그림 2]에서 보는 바와 같이 어플리케이션의 한 그룹이 각기 다른 notify가 있을 때 이 Topic 모델을 사용하게 된다. Publish-Subscribe 메시징의 요점은 다중 sender와 다중 receiver라 할 수 있다.

3.2 Queue (Point-To-Point 모델)

이 모델은 한 프로세스가 또 다른 프로세스로 메시지를 보내려 할 때 사용된다. 이것은 클라이언트가 메시징 시스템으로 메시지를 보내기만(Send) 할 수도 있고, 받기만(Receive) 할 수도 있고, 둘 다 할 수도 있다. 가장 간단한 경우는 한 클라이언트가 메시지의 Sender가 되고, 다른 클라이언트가 Receiver가 되는 경우이다.

클라이언트가 다른 클라이언트로 바로 메시지를 전달하는 것과 Queue를 거쳐 전달하는 두 가지 기본 타입이 있는데 JMS는 Queue를 사용하는 모델을 제시하고 있다. 이 Point-To-Point 모델의 요점은 [그림 3]에서처럼 다중 Sender에 하나의 Receiver를 가진다는 것이다.



[그림 3] Point-To-Point Messaging

3.3 Topic 과 Queue 인터페이스

Publish-Subscribe 메시징과 Point-To-Point 메시징은 JMS parent의 인터페이스를 상속하는 6개의 주요 인터페이스를 정의하고 있으며, [그림 4]에서 그 관계를 보이고 있다.

JMS Parent	Publish-Subscribe	Point-To-Point
Destination	Topic	Queue
ConnectionFactory	TopicConnectionFactory	QueueConnectionFactory
Connection	TopicConnection	QueueConnection
Session	TopicSession	QueueSession
MessageProducer	TopicPublisher	QueueSender
MessgaeConsumer	TopicSubscriber	QueueReceiver

[그림 4] Publish-Subscribe와 PTP 인터페이스의 관계

4. IDL/SSO상의 JMS 인터페이스 구현

[그림 4]에서 볼 수 있는 이 JMS 인터페이스들은 자바코드로 되어 있으며, 이것을 IDL/SSO의 명세대로 재매핑 시키고 구현코드를 삽입하게 된다. 메시징 서비스의 관리(administration)를 위하여 필요한 ConnectionFactory와 Connection 인터페이스를 구현하기 위하여 IDL/SSO 시스템의 네이밍 서버를 사용하고, Sun사에서 제공하는 분산 시스템 환경 JavaRMI를 플랫폼으로 한다.

IDL/SSO에서 병행성 정의는 서버객체의 각 메소드에 대하여 이루어질 수 있으며, procedure나 function중 하나의 값이 정의될 수 있다. 이들 인터페이스에 정의된 메소드들의 기능에 따라 상호 배제적인 수행을 요하는 메소드는 procedure로, 병행적인 수행을 요하는 메소드는 function로, 각 메소드 앞에 단순히 이러한 키워드 하나만의 삽입으로 병행성이 쉽게 정의될 수 있다. IDL/SSO는 서버객체의 인터페이스 이외에 명세 내에서 인터페이스에 대한 구현코드를 기술할 수 있도록 지원한다.

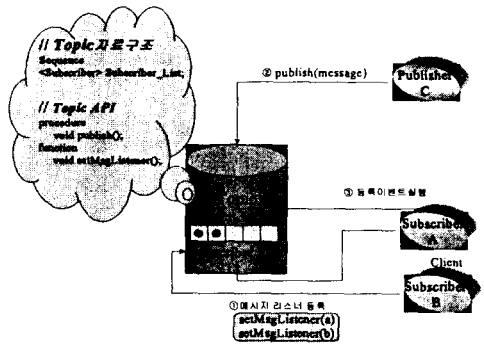
[그림 5]는 메시징 서버 모델 Topic과 Queue에 대한 JMS API이다.

Topic 인터페이스의 세부구조는 [그림 6]에서 보는 바와 같고, API가 정의하는 메소드 외에 별도의 구현 메소드 send()와 setMsgListener()를 추가하였다.

Topic	Queue
interface Topic extends Destination{ public String getTopicName() throws RemoteException;}	interface Queue extends Destination{ public String getQueueName() throws RemoteException;}

[그림 5] JMS API (Topic과 Queue)

setMsgListener()를 통해 메시지 리스너를 등록하고, Subscriber_List를 Sequence type으로 저장한다. Publisher가 Topic의 send() 메소드를 호출하고, 여기서 등록된 Subscriber의 리스트를 검사하여 메시지를 전송하게 된다.



[그림 6] Topic 인터페이스의 구현 구조

아래의 코드는 Topic 인터페이스에 대한 구현의 일부이다.

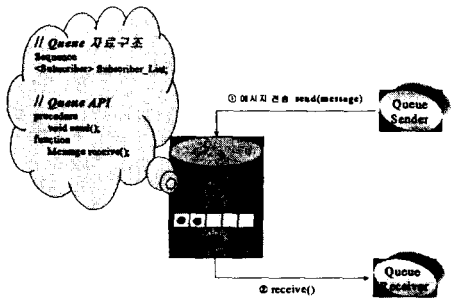
```
public:
    sequence <Subscriber> Subscriber_List;
    function string getTopicName();
    procedure void send(in string data);
    procedure void setMsgListener();
//중략...
%javacode start:
    //중략...
    procedure void send(in string data) implements {:
        try{
            for(int a=0; a<Subscriber_List_size(); a++) {
                Subscriber subscriber =
             (Subscriber)Subscriber_List_elementAt(a);
                Subscriber.onMessage(data);
            }
        } catch (Exception e) {}
    :};
    procedure void setMsgListener(Subscriber s) implements
    {:
        try{
            Subscriber_List_add(s);
        } catch (Exception e) {}
    :};
%javacode end;
};
```

[코드 1] Topic 인터페이스 정의와 구현코드

코드에서 보는 바와 같이 Topic을 구현하기 위해 만든 메소드 send()와 setMsgListener()는 상호배제적인 수행을 요하므로, 키워드 procedure를 사용한다. 클라이언트의 연결에 대한 리스트를 저장하고, 관리하기 위해 IDL/SSO의 명세에 정의된 데이터 타입중 하나인 Sequence 타입을 사용한다.

Queue 인터페이스의 구조는 [그림 7]에서 보이고 있으며, 이것은 Topic이 가지는 구조와 비슷하다. Queue 에서는 Sender가 보낸 메시지를 Sequence 데이터 타입인 Message_list에 유지하고, Receiver의 명시적인 receive() 호출이 있을 때 마다 메시지를 전달하게 된다.

Topic은 Publisher(공급자)가 메시지 전달의 주도자가 되고, Subscriber(소비자)가 수동적인 목표물이 되는 푸시모델(push model), Queue는 Receiver(소비자)가 능동적으로 메시지를 요청하는 풀모델(full model)이다.[3]



[그림 7] Queue 인터페이스 구현 구조

5. 결론

메시지 서비스는 통합된 엔터프라이즈 어플리케이션 내부 동작에 필수적인 요소이다. 이러한 메시지 서비스를 위하여 JMS는 엔터프라이즈 메시징 시스템을 사용하고자 하는 클라이언트에게 보다 신뢰성 있고, 유용한 접근 방법을 제시한다.

효율적인 메시지 시스템의 동작을 위해서는 병행성을 최대화하는 것이 바람직하다. IDL/SSO 시스템은 체계적인 명세기법을 통하여 병행성을 지원하면서도 메시지 전달에 대한 효율성과 신뢰성을 제공하여 준다.

본 논문에서는 JMS API를 IDL/SSO로 구현함으로써 메시지 시스템의 병행성을 용이하게 정의할 수 있었고, 외부 API의 구현뿐만 아니라, 내부적인 구현체계에 있어서도 병행성의 정도와 구현의 신뢰도를 증대시킬 수 있었다.

[참고문헌]

[1] Gopalan Suresh Raj, "Java Message Service(JMS)" Sept 1999, Available at http://www. execpc.com/~gopalan/jms/jms.html

[2] 박양수, 김현규, 이명준, 한상영, "CORBA 개방형 분산 환경을 위한 공유 객체 명세 언어 시스템." 한국정보처리학회 논문지 제5권 제2호. Feb.1998.

[3] 성혜영, 문남두, 김현규, 박양수, 이명준 "CORBA 유형 이벤트 서비스의 지원." 한국정보과학회 '98 봄 학술발표논문집(25, 1), pp.208-210. 1998.

[4] Fiorano Software Inc., "Fiorano Message Queue" Feb 1996, Available at http://www. fiorano.com.

[5] JavaSoft Inc., "JavaIDL: An Integrating Java with CORBA," (alpha.1) Feb 1996, Available at http://www. javasoft.com.

[6] JavaSoft Inc., "JavaRmi: Java Remote Method Invocation," 1996, Available at http://www.javasoft.com.

[7] Zhonghua Yang, Keith Duddy, "Distributed Object Computing with CORBA," DSTC Technical Report 23, June 1995.

[8] JavaSoft Inc., "Mapping IDL to Java," (alpha.1) Feb 1996, Available at http://www.javasoft.com.