EAI를 위한 통합 메신저 어댑터

정성혜[0] 이은서 이경환
중앙대학교 컴퓨터 공학과
{comet, eslee, kwlee}@object.cse.cau.ac.kr

Integration Messenger Adapter for EAI

Seong-Hye Cheong[0] Eun-Seo Lee Kyung-Whan Lee
Dept. of Computer Science and Engineering, Chung-Ang University

요 약

클라이언트/서버 환경, 다양한 플랫폼지원, 다양한 어플리케이션 인터페이스 통합관점에서 소프트웨어 재사용 방법은 소프트웨어 공학의 주요 이슈가 되고 있다. 최근 호환되지 않는 어플리케이션을 통합 처리하는 방법으로 EAI(Enterprise Application Integration: 전사적 어플리케이션 통합)가 대두되고 있다. EAI는 이 기종간에 어플리케이션을 통합하는 것으로서, 필요한 정보의 통합, 관리를 제공하며, 다른 환경의 어플리케이션에서 이질적인 데이터를 사용할 수 있도록 환경을 구현 한 것이다. 이러한 어플리케이션을 통합하는데 있어서 본 논문에서는 최소한의 변경만으로 통합이 가능하고, 쉽게 적용 할 수 있도록 하기 위한 방법으로 통합 메시지 패싱 방법을 제시 하고자 한다. 본 논문에서는 EJB(Enterprise Java Beans), JMS(Java Messaging Service)를 이용하여 통합 메시지 어댑터를 정의하고 그 방법을 제시한다.

1. 서 론

최근 몇 년 동안 소프트웨어 산업이 급속도로 발전해 왔다. 다양한 플랫폼 및 어플리케이션 인터페이스를 통합하는 방법의 요구사항이 증가되었으며, 그 방법 중의 하나인 EAI(Enterprise Application Integration)이다. EAI는 기업 내 상호 연관된 모든 어플리케이션을 유기적으로 연동하여 필요한 정보를 중앙 집중적으로 통합, 관리, 사용할 수 있는 환경의 구현을 의미한다[1]. EAI는 재사용성, 유지보수성, 무 결성, 안정성에 뛰어난 효과를 얻을 수 있고, 어플리케이션 통합시 각 어플리케이션간의 메시지 교환이 주된 일이라고 볼 수 있다. 본 논문에서는 이 부분을 중점적으로 연구 하고자 한다.

본 논문에서는 어플리케이션의 변화를 최소화하고, 이기종간의 데이터 재사용성을 높일 수 있도록 하기 위한 방법으로서, EJB(Enterprise Java Beans), JMS(Java Messaging Service)를 이용하여, 전사적인 환경에서 EAI를 구현 가능하기 위한 Integration Messenger Passing 방법을 제시한다.

본 논문의 구성은 다음과같다. 2장에서는 기반 연구로서 EAI, EJB, JMS에 대해서 설명하고, 3장에서는 EAI의 기법 중에서 메시지 패싱 방법을 정의하고, customization 가능하도록 제시한다. 4장에서는 3장에서 제시한 Integration Messenger Passing 방법을 기반으로 EJB, JMS를 이용하여 Banking 시스템을 구현한다. 마지막으로 5장에서는 결론 및 향후 연구 과제로 구성되어 있다.

2. 기반 연구

2.1 EAI (Enterprise Application Integration)

EAI는 상호 연관된 모든 어플리케이션을 유기적으로 연동하여 필요한 정보를 중앙 집중적으로 통합, 관리, 사용할 수 있는 미들웨어를 구현하는 것을 말한다[1]. EAI 구축 3단계는 다음과 같다. 첫째, 어플리케이션간의 연결성을 메시지기반 미들웨어나 어플리케이션 어댑터를 통해 연결성을 확보한다. 둘째, 이종의 데이터를 데이터 브로커에 의해서 자료를 자동 변환한다. 셋째, 어플리케이션을 통합하는 비즈니스 프로세스를 자동화 도구나 트랜잭션 및 재사용 하여 생성한다.

2.2 EJB (Enterprise Java Beans)

EJB(아키텍처는 객체지향 분산 엔터프라이즈 어플리케이션의 개발 및 분산 배치를 위한 컴포넌트 아키텍쳐이다. EJB에는 세 종류의 빈이 있는데, 세션 빈(Session Bean)과 엔티티 빈(Entity Bean)과 메시지 드라이브 빈(Message Driven Bean)으로 나누어진다[3]. 세션 빈은

비즈니스 프로세서로 관리하는 방식에 따라 상태유지 세션빈과 무상태 세션빈으로 나누어진다. 엔티티 빈(Entity Bean)은 데이터베이스와 같은 영속적인 저장 메커니즘에 저장되는 비즈니스 객체이며, 이 자료의 영속성을 보장하는 방법에는 빈 관리 영속성(Bean-Managed Persistence)과 컨테이너 관리 영속성(Container Managed Persistence)으로 나눌 수 있다. 메시지 드라이브 빈은 JMS의 메시지 처리 기능을 표현하기 위해서 EJB 2.0에서부터 포함되었다.

2.3 JMS (Java Messaging Service)

JMS는 각 어플리케이션간에 메시지 서버의 메시지 큐(Message Queue)에 의해서 비동기 메시지를 전달할 수 있는 어플리케이션 개발에 사용하는 API이며, 모든 J2EE 컴포넌트나 웹 컴포넌트간의 메시지를 보내거나 받을 수 있다[4]. JMS 통신 모델은 다음 2가지가 있다. 하나는 point-to-point 모델로 클라이언트 메시지가 메시지 큐를 통해 오직 하나의 클라이언트에게 메시지를 전달하며, Publish/Subscribe 모델은 하나의 주제로 메시지를 publish하면 그 주제와 관련된 클라이언트 모두에게 메시지 전달된다. 또한 JMS는 영속적 메시지(Persistent Messages) 전달을 보증하며 트랜잭션을 지원한다.

EAI에서 통합을 하는 방법은 두 가지 즉, Data 통합과 Process 통합으로 나누어진다. 본 논문에서는 Process 통합 방법을 위한 메시지 패싱에 관하여 연구한다. 재사용성을 높이기 위해서 컴포넌트 화 할 수 있도록 컴포넌트 기반 개발 방법으로 EJB와 JMS를 이용하여 Messenger Adapter를 구현한다.

3. Integration Messenger Adapter

본 논문에서는 EAI 기법 중에서 어플리케이션의 변화를 최소화하고, 이 기종간의 데이터 재사용을 높일 수 있도록 하기 위한 방법으로 메시지 패싱 기법을 제시한다.

클라이언트와 서버의 사이에서 클라이언트에서 보내지는 메시지를 받아서 해당 서버로 연결시켜주는 작업을 하는 메시지 브로커를 설계한다. 이때 트랜잭션을 해결하기 위한 방법으로, 메시지를 받고 보내는 작업을 할 때 메시지 큐를 이용해서 그것을 순차적으로 보낼 수 있도록 한다.

3.1 Integration Messenger Passing

Integration Messenger Adapter를 살펴보기 전에 먼저 기존의 Integration Messenger passing 방법에 대하여 알아보기로 한다.

Integration adapter는 기존의 어플리케이션 인터페이스를 공통의 인터페이스로 변환한다. 클라이언트의 어플리케이션과 서버가 통신할 때 메시지 어댑터라는 표준 인터페이스를 제공함으로써 쉽게 통신을 할 수 있도록 도와준다.

Integration messenger는 어플리케이션간의 통신 의존성을 최소화한다. 클라이언트 어플리케이션이 서버 측 어플리케이션에 어떤 요청을 하면 그 메시지를 통합 메신저(Integration Messenger)가 중간에서 해석해주는 역할을 한다.

아래 [그림 1]과 같이 클라이언트에서 이벤트가 발생하면 메시지를 Integration Messenger(broker)에게 보내고 그 메시지를 해석하여 해당 Integration adapter로 보내준다. 그러면 Integration adapter는 받은 메시지를 해석하여 각 어플리케이션에 맞는 작업을 수행한 후 결과를 리턴 한다. 이처럼 Integration messenger는 클라이언트에서 보낸 메시지를 해석하여 원하는 어플리케이션의 통합 어댑터에 전달하여 각 서버의 어플리케이션간의 통신 의존성을 최소화 할 수 있다.

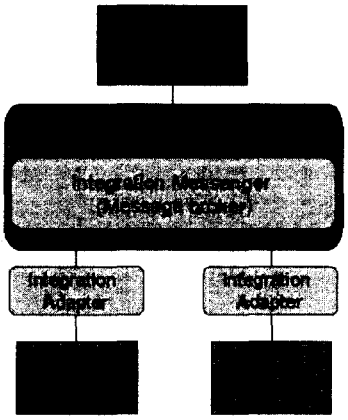

그림 1 Integration with Integration messenger

3.2 Integration Messenger Adapter

Integration Messenger Adapter는 클라이언트와 서버 사이에서 메시지를 패싱해 주는 작업을 한다. 클라이언트에서 어떤 이벤트가 발생하면, 요청이나 요구하는 이벤트, 메신저 브로커가 이 메시지를 받아서 JMS를 통해서 해당 어플리케이션에 요구사항을 보낸다. 서버에서는 그 요구사항을 처리하여 다시 메시지 어댑터로 넘겨준다. 메시지 어댑터는 받은 메시지를 다시 요구했던 클라이언트에게 넘겨준다. 이때 메시지를 처리할 때 메시지 큐를 이용하여 모든 일을 처리한다. 아래의 [그림 2]는 이러한 Message Driven Bean과 JMS, 클라이언트와 서버의 관계에 대해서 설명하고 있다. 메시지 어댑터는 EJB의 Message Driven Bean과 JMS를 이용한다. 메시지를 받고 넘겨줄 때 JMS의 메시지 큐를 이용함으로써 트랙잭션을 보장할 수 있고, 페일 오버(failover)를 해결 할 수 있다.

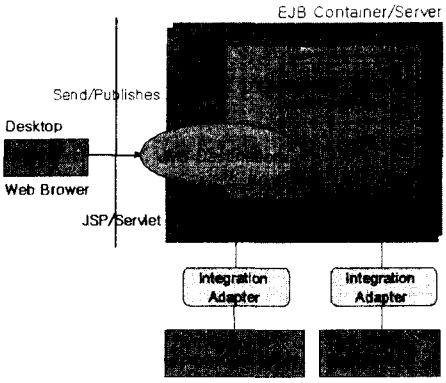

그림 2 Client view of message driven bean

4. 적용 및 사례연구

앞 절에서 설명한 메시지 브로커를 Banking System에 적용 시켰다. Customer 클래스에서 변경이 있을 경우, Customer enevt에서 변경을 하고, 처리 사항을 그것을 eventBean에 알려 주면, eventBean에서는 Customer클래스에 다시 알려준다. 아래 [그림 3]은 eventBean과 다른 클래스들과의 관계를 클래스 다이어그램으로 표현한 것이다.

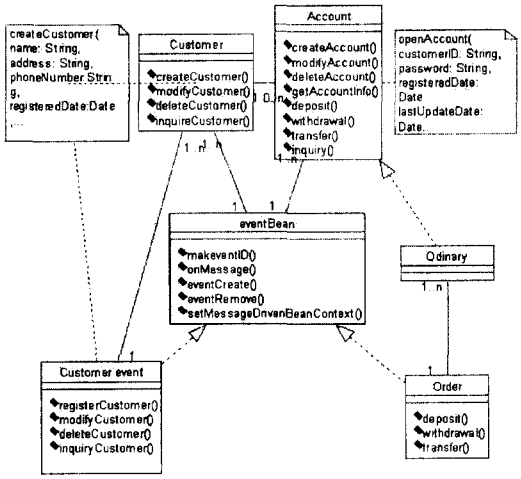

그림 3 EventBean과 클래스들 간의 연관 관계

JMS로 만들어진 컨테이너와 JMS 큐의 메시지는 Message Driven Bean을 통해서 전달된다. 아래의 [표 1]은 enevtBean에서 JMS와 EJB에 메시지를 생성하고 삭제하는데 사용되는 주요 메소드와 기능의 설명을 나타낸 것이다..

표 1 Method implemented in message drive beans

메소드	설명
onMessage()	클라이언트에게 메시지를 전송시키는 메소드.
eventCreate()	컨테이너에 bean instance를 추가시키는 메소드.
eventRemove()	컨테이너에 bean instance를 제거하는 메소드.
setMessageDrivenBeanContext()	트랜잭션과 관련된 메소드로 bean이 실행될 때 그 bean의 정보를 제공하는 메소드.

5. 결론 및 향후 연구 과제

본 논문에서는 기존의 Message Passing을 이용하여 어플리케이션의 변화를 최소화하고 재사용을 높이기 위한 방법으로 Integration Messenger Adapter를 제시하였다.

Integration Messenger Adapter는 클라이언트와 서버의 사이에서 eventBean 브로커가 보내지는 메시지를 받아서 처리한다. 여기에서 메시지를 처리할 때 메시지 큐를 이용함으로써 트랙잭션을 보장할 수 있고, 페일 오버(failover)를 해결 할 수 있다. 또한 메시지 기반 통합이기 때문에 약간의 변화로 어플리케이션의 통합이 가능하였다.

본 논문에서는 기존 EAI의 기법 중에서 메시지 기반의 Integration Messenger Adapter에 대하여 정의하였다.

향후 연구과제로 어플리케이션 통합에 대한 다른 방법인 data 기반에서 통합에 대해 연구하고자 한다.

[참고문헌]

[1] Francis X. Maginnis, William A. Ruh, "Enterprise Application Integration", John Wiley & Sons, 2000

[2] JEFFREY C. LUTZ, "EAI Architectural Patterns", EAI Journal, 2000

[3] Adatia, Arni, Gabhart, Griffin, Juric, Ltoo, McAllister, Mulder, Nagarajan, O'Connor, Osborne, Sarang, Tost, Young, Berr, "Professional EJB", Wrox, 2001

[4] Michael Kovacs, Paul Giotta, Scott Grant, "Professional JMS", Wrox, 2001

[5] eai Journal, "http://www.eaijournal.com"

[6] Intelligent EAI, "http://www.intelligenteai.com"