主題

엔터프라이즈 자바 기술과 전자상거래

한국전자통신연구원 김성훈, 장철수, 함호상

차 례

Ⅰ. 서 론

전자상거래 애플리케이션 플랫폼(E-Commerce Application Platform Server: ECAP)은 웹을 이용하는 전자상거래 컴포넌트와 기업의 전산시스템 사이에 위치하면서 웹 기반의 분산시스템 개발을 쉽게 도와주고 안정적인 트랜잭션 처리를 보장해 주는 일종의 전자상거래용 미들웨어이다. 또한 웹 애플리케이션 서버는 CORBA나 COM과 같은 분산객체 기술, TP Monitor와 같은 트랜잭션 처리 기술, 엔터프라이즈 자바빈즈(EJB)와 같은 컴포넌트 기술, JDBC, ODBC와 같은 데이터베이스 액세스 기술, 웹 상태 관리와 같은 웹서버 기술, 보안서비스와 디렉터리서비스 같은 분산처리 기술 등 많은 기술이 집약되어 있는 시스템 소프트웨어라고 할 수 있다. ECAP는 개방형 표준인 썬마이크로시스템즈의 J2EE 명세를 준수하고, 확장성과 안정성을 고려하는 웹 애플리케이션 서버를 하부구조로 하여 애플리케이션 계층에서 전자상거래를 위한 다양한 컴포넌트를 제공하는 일종의 통합 전자상거래용 서버로 정의한다[1].

현재 웹 애플리케이션 서버가 급부상하고 있는 배경에는 무엇보다도 인터넷이 기업 정보시스템의 인프라로 자리를 잡아가고 있다는 점이다. 대표적인 것이 웹 기반의 전자상거래 시스템의 확산이다. 전자상거래가 앞으로 피할 수 없는 대세라면 대규모 트랜잭션 처리를 안정되게 관리해줄 웹 애플리케이션 서버의 필요성이 증대할 것이다. 이와 함께 COM, CORBA로 대표되는 분산 객체기술이 성숙하고 있다는 점도 웹 애플리케이션 서버의 존재가치를 높여주고 있다. 소프트웨어 개발의 용이성과 재사용성, 유지보수 편의성 등으로 객체기술의 효용성은 이미 인정 받고 있다.

그 동안 웹 브라우저상의 애플릿 제작 등 클라이언트 분야에서만 제한적으로 사용돼온 자바 기술이 서버 기반의 객체 표준인 엔터프라이즈 자바빈스(EJB)의 탄생으로 기업용 대형 애플리케이션 개발에도 적용될 수 있게 되었다. 또한 운용체계에 상관

없이 운용되는 소프트웨어를 개발할 수 있는 언어인 자바가 서버용 소프트웨어 개발환경을 제공하게 됨으로써, 웹 기반의 분산 시스템 개발언어로 자바의 채용이 증가할 전망이고 이에 따라 분산시스템의 미들웨어인 웹 애플리케이션 서버의 필요성도 커지고 있다[2, 12].

본 논문에서는 현재 업계의 ECAP의 주요 핵심 기술이자 엔터프라이즈 애플리케이션 서버 시장의 표준으로 자리를 잡고 있는 엔터프라이즈 자바 기술[4]과 엔터프라이즈 자바의 주요 서비스 기술인 엔터프라이즈 자바 빈즈에[9] 대해 소개하고 ECAP의 전자상거래 적용 예로써 엔터프라이즈 자바 빈즈를 이용하여 구축한 EJB 컴포넌트[3]에 대해 살펴보겠다.

Ⅱ. 엔터프라이즈 자바 기술

1. 기술 개요

기업 내부 전산 시스템이 인터넷과 접목되면서 기존의 애플리케이션들은 보다 복잡한 형태를 띄게 되었고, 기능적인 측면에서도 비즈니스 업무가 외부 비즈니스 파트너 또는 고객들과의 연동할 수 있는 기능들이 추가되었다. 즉, 기존의 애플리케이션들이 기업 대 고객 또는 기업 대 기업간의 전자 상거래를 위한 기능을 추가적으로 수행하게 되었다. 또한 그동안 예측 가능했던 고객들의 요청이 웹과 연동되면서 예측 불가능한 상태로 변화되었으며, 대응할 수 있는 고객의 수도 그만큼 증가하게 되었다. 이러한 변화를 수용하기 위한 엔터프라이즈 애플리케이션 시스템의 요구 조건은 다음과 같다.

- 확장성(Scalability) : 과거 기업 내부의 전산 시스템은 동시에 접속할 수 있는 사용자의 수를 예측할 수 있었다. 그러나 웹과 연동되면서 동시 접속할 수 있는 사용자의 수는 예측 불가능하게 되었다. 따라서 서버 시스템은 접속 사용자의 수에 대해 유연하게 대처할 수 있는 확장성을 갖추어야 한다.
- 안정성(Availability) : 웹 사용자가 애플리케이션 시스템에 접속할 수 있는 시간은 예측 불가능하다. 따라서 서버 시스템은 언제든지 고객에게 응답할 수 있도록 안정성을 갖추어야 한다.
- 보안성(Security) : 기업의 전산 시스템이 웹과의 연동을 부담스럽게 생각하는 부분이 바로 보안성 때문이다. 인터넷은 장벽이 없다. 따라서 서버시스템은 개인의 정보와 기업의 시스템을 보호하기 위한 보안성을 갖추어야 한다.

썬 마이크로시스템즈의 엔터프라이즈 자바 기술은 기존의 엔터프라이즈 애플리케이션이 e-비즈니스 환경에 접목될 수 있도록 컴포넌트 기반의 서버 중심적인 다계층 방식의 애플리케이션 아키텍춰이다[1]. 엔터프라이즈 자바는 서버측에 여러 종류의 미들웨어 컴포넌트들을 포함하고 있으며, 멀티 서버 플랫폼에 동작할 수 있도록 설계되었다. 엔터프라이즈 자바 플랫폼이 다중 도메인 애플리케이션의 구조를 가짐으로써 확장성을 지원 하고 있으며, 네이밍, 보안, 트랜잭션, 메세징, 데이터베이스와 같은 미들웨어 시스템을 포함함으로써 안정성과 보안성을 갖추고 있다[11].

1997년에 엔터프라이즈를 위한 플랫폼 개발을 시작한 썬 마이크로시스템즈는 엔터프라이즈 자바 API로 불리는 표준 자바 확장 API들을 발표했으며, 엔터프라이즈 애플리케이션을 구축하기 위한 다양한 종류의 미들웨어 표준 인터페이스를 제공하였다. 대표적인 예로 EJB (Enterprise Java-Beans)가 있는데, EJB는 서버 제공자에 관계없이 동일한 자바 인터페이스를 제공하는 서버측 자바 컴포넌트 프로그래밍 모델이다. 그러나 이러한 EJB

와 확장 API만으로는 엔터프라이즈 애플리케이션의 모든 요구 조건을 수용할 수 없었다. 엔터프라이즈 자바 플랫폼은 컴포넌트에 대한 표준 프로그래밍 모델인 EJB기술에 통신이나 트랜잭션 등을 위한 상호 운용 가능한 프로토콜을 지원할 수 있도록 자바의 확장 API들을 모두 포함하는 표준 자바의 확장 플랫폼이다. 즉, EJB가 가지고 있었던 상호 운용성과 플랫폼 일관성, 이식성 등의 단점을 보완한 서버 아키텍춰라 할 수 있다.

본 절에서는 엔터프라이즈 자바 기술의 컴포넌트 애플리케이션 모델과 플랫폼이 제공하는 다양한 종류의 서비스, 그리고 다른 컴포넌트 기반의 미들웨어 플랫폼과의 비교를 하겠다

2. 계층형 분산 애플리케이션 모델

엔터프라이즈 자바 플랫폼은 계층형 분산 애플리케이션 개발을 위하여 엔터프라이즈 자바 빈 컴포넌트[9], 서블릿[8] 및 JSP(Java Server Page) 컴포넌트[8] 그리고 애플리케이션 클라이언트 컴포넌트 및 애플릿 등의 컴포넌트에 대한 명세를 정의한다. 엔터프라이즈 자바의 계층형 분산 애플리케이션 모델이란, 애플리케이션 로직이 각 계층의 환경에 종속됨으로써, 기능별로 분리된다는 것을 의미하고, 서로 다른 애플리케션 컴포넌트를 다른 서버(또는 같은 서버)에 배포하여 운용할 수 있다는 것을 의미한다. 즉, 서블릿과 JSP페이지와 같은 웹 컴포넌트와 비즈니스 로직이 구현된 엔터프라이즈 자바빈즈 컴포넌트를 서로 다른 서버 혹은 같은 서버 내에 서로 다른 가상 머신에서 운용할 수 있다. 구체적으로 엔터프라이즈에서 정의하는 애플리케이션 모델은 그림 1과 같이 사용자(클라이언트) 계층, 웹 계층, 비즈니스 계층, 엔터프라이즈 정보 시스템 계층으로 나뉘어 진다.

• 사용자 계층

엔터프라이즈 자바의 애플리케이션은 다음의 두

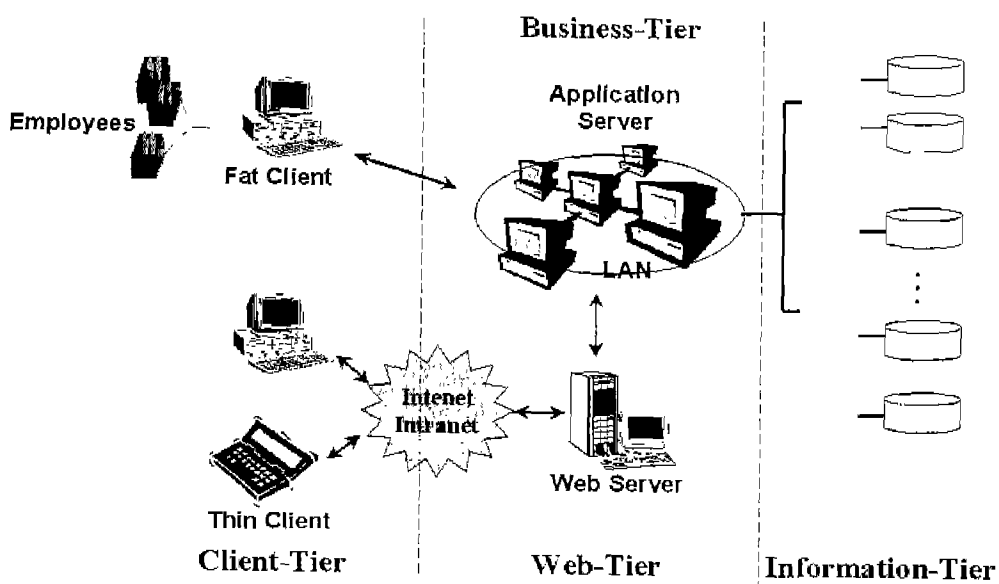

그림 1. 분산 계층형 애플리케이션 서비스 모델(출처 : [12])

가지 형태로 존재한다. 첫번째는 웹을 기반으로 하는 애플리케이션으로써 사용자의 웹 브라우저가 사용자 계층에서 동작하고, 웹 서버의 웹 계층에서 JSP나 서블릿에 의해 작성되는 동적인 웹 페이지나 정적 HTML 웹 페이지를 통하여 동작한다. 두 번째로 웹 기반이 아닌 경우에는, 단독 사용자 애플리케이션이나 이동 디바이스 또는 네트워 기반의 전자제품 등에 탑재된 에플릿이 사용자 계층에서 동작한다. 이 경우에 클라이언트는 웹 계층을 사용하지 않고 RMI/IIOP와 같은 별도의 통신 방법으로 비즈니스 계층의 엔터프라이즈 빈을 이용한다. 웹 기반이 아닌 사용자 계층은 사용자의 입력을 받아들여 비즈니스 계층의 엔터프라이즈 빈에게 전달하는 자바 빈 클래스도 포함한다.

• 웹 계층

엔터프라이즈 자바의 웹 컴포넌트는 JSP 페이지나 서블릿으로 구성된다. 서블릿과 JSP 페이지를 호출하는 HTML 페이지는 애플리케이션 결합 시 웹 컴포넌트와 같이 배포되지만 엔터프라이즈 자바의 웹 컴포넌트로는 고려되지 않는다. 웹 계층의 웹 컨테이너는 웹 컴포넌트의 인스턴스를 생성, 리소스의 할당, 사용자 요청에 대한 응답, 엔터프라이즈 빈 호출등과 같은 기능을 지원한다.

• 비즈니스 계층

은행, 재정 등 특정 비즈니스 분야의 로직 코드는 비즈니스 계층에서 동작하는 엔터프라이즈 빈에 의해 처리된다. 엔터프라이즈 빈은 클라이언트 프로그램으로부터 데이터를 받아 처리하거나, 저장을 위해 엔터프라이즈 정보 시스템으로 전달한다. 엔터프라이즈 빈은 또한 데이터 저장소로부터 데이터를 받아 처리를 하고 클라이언트 프로그램으로 되돌린다. 비즈니스 계층의 컨테이너는 엔터프라이즈 빈의 트랜잭션, 생명주기관리, 상태 관리, 멀티쓰레딩, 보안 그리고 자원관리 등과 같이 매우 복잡한 시스템 관련 기능을 수행한다. 3장에서 다루는 EJB 표준은 바로 이 비즈니스 계층의 엔터프라이즈 빈과 그것을 적재 및 관리하는 컨테이너에 대한 표준으로 엔터프라이즈 자바의 핵심을 이루고 있다.

• 엔터프라이즈 정보 시스템 계층

엔터프라이즈 정보 시스템 계층은 엔터프라이즈 정보 시스템 소프트웨어를 다루고, 전사적 자원 관리(Enterprise Resource Planning, ERP), 메인 프레임 트랜잭션 처리, 데이터베이스 시스템, 그리고 다른 레거시 정보시스템 등과 같은 엔트프라이즈 인프라 시스템을 포함한다. 엔터프라이즈 응용 컴포넌트는 엔터프라이즈 정보 시스템을 통해 데이터베이스 접근과 같은 기능을 이용한다.

4. 엔터프라이즈 자바 표준 서비스

엔터프라이즈 자바에서는 네이밍(naming), 보안, 트랜잭션, 메세징(messaging), 데이터베이스를 위한 분산 미들웨어 서비스를 제공한다. 엔터프라이즈 자바 서비스는 표준 인터페이스를 제공함으로써 서비스 제공자에 따라 상이한 구현을 일관된 형태로 이용할 수 있다. 그림 2는 표준 서비스의 블록 다이어그램이다.

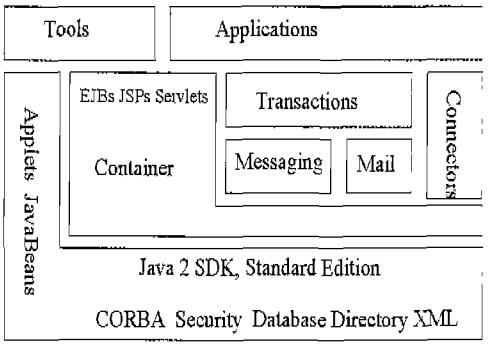

그림 2. 엔터프라이즈 자바 표준 서비스(출처:[12])

• RMI(Remote Method Invocation)/IIOP

자바 대 자바 환경에서 CORBA의 표준 IIOP 프로토콜을 사용하여 원격지 객체의 메쏘드를 호출하는 메커니즘을 제공한다.

- JNDI API(Java Naming and Directory Interface API)
  DNS, NDS, LDAP과 CORBA Naming 과 같은 네이밍/디렉터리 서비스를 이용할 수 있는 자바의 네이밍/디렉터리 인터페이스를 제공한다[6].
- JDBC API (Java Database Connection API)
  DB2, Informix, Oracle, SQL Server 그리고 Sybase등과 같은 관계형 데이터베이스를 이용할 수 있는 일관된 형태의 데이터베이스 접근 인터페이스를 제공한다[10].
- JavaMail API
  메일 또는 메시지 애플리케이션을 구축할 수 있는 프로토콜에 무관한 프레임웍을 제공한다.
- JTA(Java Transaction API)
  JTA는 트랜잭션 관리자와 리소스 관리자, 애플리케이션 서버, 클라이언트 애플리케이션 사이의 인터페이스를 제공한다[5].
- JTS(Java Transaction Service API)
  JTS는 CORBA의 OTS에 대한 자바 구현과 JTA 표준에 대한 구현 클래스를 제공한다[5].
- JMS(Java Messaging Service)
  JMS는 메시지 큐를 이용하거나 publish/subscribe 프로그래밍 모델을 사용하여 비동기적인 통신 방식을 제공한다.
- Servlet/JS
  PServlet/JSP는 동적인 웹 페이지 생성과 클라이언트 브라우저의 세션 관리를 위한 웹 서버측의 컴포넌트를 제공한다[7].
- Java IDL
  Java IDL은 Java-To-CORBA통신을 위한 리모트 인터페이스를 생성하며, IIOP를 지원하는 ORB와 IDL-To-Java 컴파일러를 포함하고 있다.
- EJB(Enterprise JavaBeans)
  애플리케이션 서버에 관계없이 동작할 수 있는 표준 컴포넌트 프로그래밍 모델을 제공하며, 컴포넌트가 구동 될 수 있도록 컨테이너의 규약을 제공한다[9].

5. 다른 플랫폼과의 관계

엔터프라이즈 자바 플랫폼과 비교될 수 있는 컴포넌트 플랫폼 모델은 OMG(Object Management Group)의 CORBA[14]와 마이크로소프트의 윈도우 DNA(Distributed interNet Applications Architecture)이다.

CORBA는 ORB(Object Request Broker)에 기반을 둔 분산 객체 인프라 구조로 정의되며, CORBA 컴포넌트 모델의 각종 서버측 미들웨어 서비스들을 정의하고 있다. CORBA와 EJB는 썬 마이크로시스템즈의 엔터프라이즈 자바의 정책처럼 상호 보완적인 관계이다. 즉, EJB를 포함한 엔터프라이즈 자바 플랫폼은 IIOP와 OTS와 같은 CORBA를 기반 구조로 설계되었다. 또한 CORBA의 컴포넌트 모델은 구현 언어에 독립적으로 구현된 EJB의 상위개념으로 정의된다. CORBA의 컴포넌트 모델을 지원하는 애플리케이션 서버들은 향후 EJB 컴포넌트를 지원할 것으로 알려져 있다. 현재 대부분의 CORBA 서버들은 CORBA와 EJB를 동시에 지원하고 있으며, 엔터프라이즈 애플리케이션 서버를 채용하려는 경향이 있다. 즉, CORBA는 엔터프라이즈 자바와 경쟁적인 측면보다 상호 보완적인 측면에서 서로를 수용하려는 방향으로 발전하고 있다[11].

이에 비해 마이크로소프트의 윈도우 DNA는 썬 마이크로시스템즈와 경쟁적인 관계로써 엔터프라이즈 애플리케이션 플랫폼을 제공하고 있다. 윈도우

표 1. 윈도우 DNA와 엔터프라이즈 자바 플랫폼과 비교

플랫폼 서비스	윈도우 DNA	엔터프라이즈 자바 플랫폼
Operation Systems	윈도우 CE/95/98/NT/2000	Any
Browser	Internet Explorer	Any
Browser Component	ActiveX Component	Applets
Web Server	Internet Information Server	Any
Web Server Component	Active Server Page	
Application Server	Microsoft Transaction Server	Any EJB application Server
Server Component	MTS Component	Enterprise JavaBeans
Communication Protocol	DCOM	IIOP
Database Access	ADO and OLE DB	JDBC and SQLJ
Transaction Management	Microsoft Distributed Transaction Coordinator	Any transaction service through JTA
Security	Microsoft security service	Java security services
Directory	Microsoft Registry	Any directory through JNDI

DNA는 인터넷을 위한 엔터프라이즈 애플리케이션 개발을 위한 환경을 제공하며, 다양한 종류의 클라이언트를 포함할 수 있다. 그러나 윈도우 DNA의 단점은 마이크로소프트 윈도우에 종속적이라는 점이다. 따라서 윈도우 DNA는 윈도우에서 제공하는 미들웨어 서비스들과 분산 컴포넌트 모델(COM)을 사용하고 있다. 윈도우에서 제공하는 미들웨어 서비스는 MTS(Microsoft Transaction Server), DTC(Distributed Transaction Coordinate), 윈도우 레지스트리, 윈도우 보안 서비스 등이다. 윈도우 DNA는 엔터프라이즈 자바와 마찬가지로 다 계층 분산 애플리케이션 모델을 기반으로 하고 있으며 기능적으로는 엔터프라이즈 자바와 유사하다.

윈도우 DNA와 썬 마이크로시스템즈의 엔터프라이즈 자바 플랫폼과의 근본적인 차이점은 개발언어 독립적이라는 측면과 개발 플랫폼 독립적이라는 측면이다. 엔터프라이즈 자바 플랫폼은 벤더 독립적인 플랫폼이다. 사용자는 개발 환경과 배포 제품, 배포 환경을 고려하여 서로 다른 제품들을 선택할 수 있다. 즉 서로 같은 표준을 공유하고 있기 때문에 특정 제품에 제한되지 않는 장점이 있다. 단, 반드시 자바로 구현 되어 있어야 한다. 윈도우 DNA는 구현언어에 독립적이다. 즉 윈도우 DNA는 자바, C++, 비주얼베이직, 델파이, 파워빌더 등과 같은 거의 대부분의 언어를 이용하여 애플리케이션을 개발할 수 있다. 표 1은 윈도우 DNA와 엔터프라이즈 자바 플랫폼과의 비교이다.

Ⅲ. 엔터프라이즈 자바 빈즈 (EJB:Enterprise JavaBeans)

EJB는 서버 상의 Java 컴포넌트와 그 컴포넌트를 위한 동작 환경을 제공하는 프레임웍으로, 보다 더 큰 프레임웍인 엔터프라이즈 자바 플랫폼의 일부분이다. EJB에서 동작하는 컴포넌트를 엔터프라이즈 빈(Enterprise Bean)이라고 하고, 그 컴포넌

트가 동작하기 위한 환경을 컨테이너(Container)라 한다. EJB는 CORBA가 언어에 독립적인 특성을 가지고 있는데 반해 Java로만 구현되어 언어 독립성의 장점을 잃지만, EJB에서는 기존에 개발자가 직접 컴포넌트로 제공해 주어야 하는 데이터베이스 트랜잭션, 보안문제, 데이터베이스 연결 풀링, 쓰레딩 관리 같은 기능을 컨테이너에서 처리해 줌으로 일반 개발자는 시스템 로직에 신경 쓰지 않으면서 더 쉽고 빠르게 동일한 기능을 구현할 수 있게 된다[15].

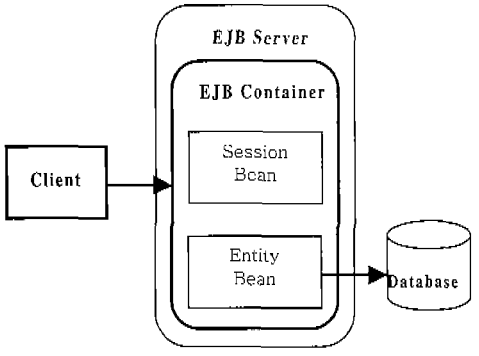

그림 3. EJB의 구조(출처:[9])

EJB를 구성하고 있는 각각의 부분은 그림 3과 같이 EJB Server, EJB Container, 그리고 2가지 형태의 엔터프라이즈 빈으로 나뉘어 지며, 각각의 역할은 다음과 같다[1, 2].

1) EJB Server: 컨테이너가 필요로 하는 서비스들을 제공하는 개념적인 정의이며 EJB Container나 데이터베이스 등이 EJB 서버에 해당한다.
2) EJB Container : 엔터프라이즈 빈들을 관리하며, 엔터프라이즈 빈들이 운영되는데 필요한 여러 기능들을 제공한다.
3) Session Bean : 세션 빈은 비즈니스 로직을 처리하기 위한 엔터프라이즈 빈이다.
4) Entity Bean : 지속적인 저장장치와 연결되어 데이터의 검색이나 조작 등을 하기 위한 엔터프라이즈 빈이다.

EJB는 세션 빈(Session Bean)과 엔터티 빈(Entity Bean)이라는 두 가지 형태의 엔터프라이즈 빈을 제공한다. 세션 빈은 CORBA 등에서 말하는 컴포넌트와 같은 개념이며, 일반 비즈니스 로직을 갖는다. 세션 빈은 데이터베이스와 같은 지속적인 저장장치에 연결되는 빈이 아니기 때문에 컨테이너가 재시작 되면 이전에 있던 세션 빈은 사라진다. 반면에 엔터티 빈은 데이터베이스의 데이터를 의미하고 있으며, 간단한 형태의 엔터티 빈은 데이터베이스 테이블의 한 레코드를 나타낸다. 그리고, 복잡한 형태의 엔터티 빈은 특정한 테이블 간의 조인(Join)된 형태를 나타낼 수도 있다. 엔터티 빈은 데이터베이스의 레코드에 연결되기 때문에 컨테이너가 재시작 되더라도 이전에 작업했던 내용을 그대로 유지한다. 이 엔터티 빈은 CORBA와 같은 경우에는 개발자가 일일이 만들어 주어야 했던 컴포넌트였으며, EJB의 가장 큰 특징 중의 하나라고 할 수 있다. 컨테이너는 하나의 엔터티 빈에 대한 여러 사용자의 동시 접근을 허용하고, 이때 발생되는 동기화 처리 등을 알아서 해결해 준다. 또한, 엔터티 빈은 세션 빈과 달리 특정 데이터를 의미하기 때문에 지속적으로 보존하여야 할 필요가 있다. EJB 명세는 이를 위한 두 가지 방법을 제공하고 있는데 그것이 컨테이너 관리 지속성 방법(Container Managed Persistence)와 빈 관리 지속성 방법(Bean Managed Persistence)이다. 컨테이너 관리 지속성은 전개 명세 (Deployment Descriptor) 파일 상에 보존하여야 할 대상을 지정하면 개발자의 추가적인 작업 없이 빈의 보존이 이루어 진다. 빈 관리 지속성의 경우에는 엔터티 빈의 저장에 관련된 일련의 코딩 작업을 빈상에서 개발자가 직접 수행하여 한다. 개발자의 입장에서 보

면 컨테이너 관리 지속성이 편리하나 복잡한 질의문이나 개발자가 빈에 대한 강한 핸들링 권한이 필요한 경우에는 빈 관리 지속성 방법이 사용된다.엔터프라이즈 빈 개발자 입장에서 보면 각 빈들은 그림 4와 같이 EJB Home, EJB Object, EJB 클래스만을 고려하면 된다. 여기서 EJB Home과 EJBObject는 각각 Home 인터페이스(Interface)와 리모트 인터페이스를 구현한 오브젝트이며, 실제 비즈니스 로직은 엔터프라이즈 빈 Class에서 구현된다. Home 인터페이스와 Home 오브젝트는 엔터프라이즈 빈 Class를 생성하고 제거하는 Factory 메쏘드(method)를 열거한다. 즉, Home 인터페이스는 클라이언트들이 요구하는 빈의 인스턴스(Instance)를 생성할 수 있는 하나 혹은 그 이상의 생성자 메쏘드들을 내제 한다. 또한, 엔터티 빈은 데이터베이스의 레코드와 연계되기 때문에 Finder 메쏘드를 가지고 있는데 이를 이용하여 클라이언트는 특정한 엔터티 빈을 접근할 수 있다.

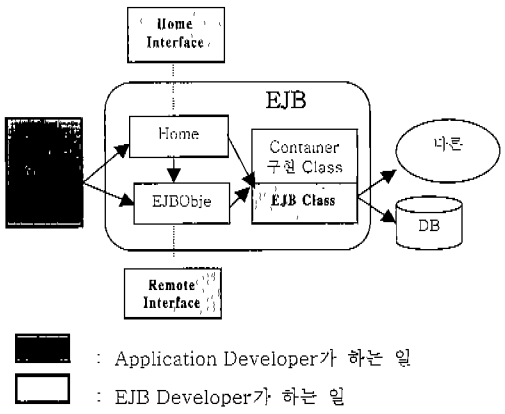

그림 4. EJB 컴포넌트의 동작(출처:[12])

리모트 인터페이스는 빈 클래스의 비즈니스 메쏘드를 나열하고 있으며, 이의 구현인 EJBObject는 빈의 대리자(Proxy)역할을 수행하는, 네트워크상에 보여지는 객체로써, 클라이언트가 EJB 인스턴스의 비즈니스 로직을 접근하기 위해서는 이 EJBObject를 사용해야 한다. 즉, 클라이언트는 EJB 인스턴스의 참조를 얻는 것이 아니라 단지 EJBObject의 인스턴스를 얻는다. 클라이언트가 메쏘드를 호출하면 EJBObject가 받아들여 그것을 EJB 인스턴스에 전달하여 실제 로직을 처리한다.

Ⅳ. 엔터프라이즈 자바 빈즈의 인터넷 쇼핑몰 시스템 적용 예

1. 쇼핑몰 빈의 종류

인터넷 쇼핑 몰을 위한 EJB 빈들은 고객 빈, 상품 아이템 빈, 상품 카타로그 빈, 주문 빈, 재고 빈, 환불 빈, 지불 빈, 배송 빈 등으로 구성되어 있으며 각각의 빈들의 관계는 그림 5와 같다. 그림 5에서 ItemCatalogBean이나 CustomerBean과 같이 빈으로 끝나는 EJB 빈들은 엔터티 빈을 의미하며, 그렇지 않은 EJB 빈들은 세션 빈을 의미하고 있다.

EJB의 세션 빈과 엔터티 빈을 접근하는 방식에는 서로 차이가 있다. 즉, 클라이언트에서 세션 빈을 참조하기 위해서는 이름 해석을 통해 EJB 홈 인터페이스를 얻고, EJB 홈 인터페이스의 create() 메쏘드를 통해 빈의 대리자 역할을 수행하는 EJB 오브젝트를 얻어, 이 오브젝트를 통해 세션 빈의 메쏘드를 수행하면 된다. 반면, 엔터티 빈은 세션 빈과 같이 EJB 홈 인터페이스의 create() 메쏘드를 통해 새로운 엔터티 빈을 만들 수 있을 뿐만 아니라, EJB 홈 인터페이스의 FindByPrimaryKey() 등의 메쏘드를 이용하여 기존에 만들어진 엔터티 빈에 접근할 수 있다. 엔터티 빈은 데이터베이스의 한 레코드와 같으므로, create() 메쏘드를 이용해 새로운 엔터티 빈을 만드는 것은 데이터베이스에 한 개의 레코드가 삽입되는 것이며, FindBy-

PrimaryKey() 등의 메쏘드를 이용한 방법은 데이터베이스의 검색 질의 (select query)를 수행한 것이다.

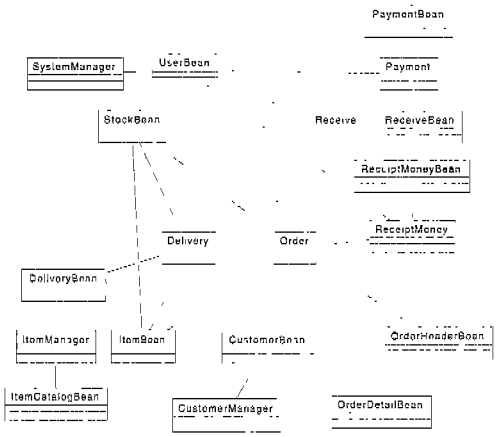

그림 5. 쇼핑 몰의 EJB 빈 들

각각의 로직을 담당하는 세션 빈들은 내부적으로 데이터베이스에 접근할 필요가 있는 경우에는 그림 6과 같이 엔터티 빈을 이용하도록 하고 있다.

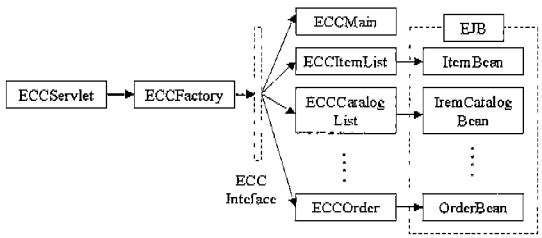

그림 6. 세션 빈에서의 엔터티 빈 이용

2. 쇼핑몰 빈의 트랜잭션 속성

쇼핑 몰에서 사용하는 각종 엔터프라이즈 빈들은 컨테이너 관리(Container Managed) 트랜잭션 모드 중 TX_REQUIRED 속성을 갖도록 설계되었다. 컨테이너 관리 모드에서 엔터프라이즈 빈의 개발시에는 트랜잭션의 처리에 거의 신경 쓸 필요가 없으며, 단지 전개 명세(Deployment Descriptor)에서 트랜잭션 속성 항목에 컨테이너 관리 트랜잭션 속성 중 하나로 지정만 하면 컨테이너가 알아서 트랜잭션 처리에 대한 보장을 해준다. 컨테이너 관리 트랜잭션 속성 중 본 구현에서 사용된 TX_REQUIRED 속성은, 트랜잭션 상태인 클라이언트에서 호출된 엔터프라이즈 빈이 클라이언트의 트랜잭션 컨텍스트(Transaction Context)를 이어 받아 실행 될 수 있도록 한다. 또한, 엔터프라이즈 빈이 TX_REQUIRED 속성을 가지고 있을 때, 트랜잭션 상태에 있지 않은 클라이언트에서 호출된 엔터프라이즈 빈은 새로운 트랜잭션 컨텍스트를 만들어 실행된다

3. 서버 시스템 구성

고객으로부터 요구 메시지를 받은 웹 브라우저는 웹 서버를 통해 ECCServlet이라는 서블릿(Servlet)을 호출하도록 구성하였는데, 이 서블릿은 그림 7과 같이 비즈니스 로직에 해당하는 EJB 빈의 클라이언트가 되어 서비스를 처리한다.

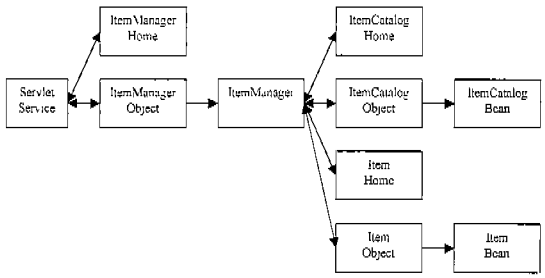

그림 7. 서블릿 구조

서블릿은 요구 메시지의 분석과 HTML 문서 생성 및 요구에 따른 비즈니스 로직을 수행한다. 즉, ECCServlet은 ECCFactory 클래스를 통해 고객의 요구 메시지를 분석하고, 각기 해당되는 비즈니스 로직을 로드(load)하여, 분석된 메시지를

ECC Interface를 통해 비즈니스 로직에 전달시켜 서비스를 수행한다. 해당되는 비즈니스 로직들은 주어진 요구 메시지에 따라 세션 빈이나 엔터티 빈 등을 호출하여 실제 서비스를 이루며, 그 결과를 HTML 문서로 생성하여 웹 브라우저에게 전달한다.

4. 쇼핑 몰 엔터프라이즈 빈의 구현

컨테이너 관리 엔터티 빈의 경우에는 컨테이너를 통해 관리되는 데이터베이스 테이블의 필드명을 전개 명세(Deployment Descriptor)에 지정해 주면, 컨테이너는 이 명세 내용을 참조해서 자동적으로 JDBC 구현 코드를 생성시켜준다. 그리고, 모든 엔터티 빈에는 Primary Key를 이용해 엔터티 빈을 찾도록 하기 위한 FindByPrimaryKey() 메쏘드가 존재하도록 EJB 1.1 명세에 정의되어 있는데, 이 메쏘드 또한 전개 명세에서 Primary Key로 사용되는 필드들 만을 지정해 주면 컨테이너에서 자동적으로 코드를 생성해 준다. FindByPrimaryKey() 이외의 검색 메쏘들 또한 전개 명세에서 검색 조건만 지정해주면 JDBC 구현 코드를 자동적으로 생성시켜준다. 이 외에도, 대리자 역할을 수행하는 EJBObject와 실제 구현 클래스 사이의 통신 역할을 담당하는 모듈 또한 자동으로 생성된다. 이와 같이 많은 구현 코드등이 자동적으로 생성되므로 엔터프라이즈 빈의 구현 시간은 급격히 감소되고 개발시에는 거의 비즈니스 로직에만 신경쓰면 된다.

5. 관리자측 컴포넌트

사용자로 등록된 관리자는 그림 8의 ItemManager 세션 빈을 이용하여 상품 배치를 위한 상품분류를 작성하여야 한다. 상품 분류는 ItemManager 세션 빈내에서 ItemCatalogBean 엔터티 빈을 이용하여 최상위 레벨에서 임의의 단계인 하위레벨 까지 자유롭게 분류될 수 있다. 또한, 분류된 각 위치에 ItemManager 세션 빈에서 ItemBean 엔터티 빈을 이용해 상품을 배치하며 재고를 설정한다. 상품분류 등록이나 상품 배치시의 이미지 등록은 웹 브라우저로부터 클라이언트의 이미지를 서버가 HTTP 프로토콜로 받아 들여 서버의 이미지 디렉토리에 유일한(unique) 이름으로 저장한 후, 해당 상품의 선택 시 브라우저에게 전송된다.

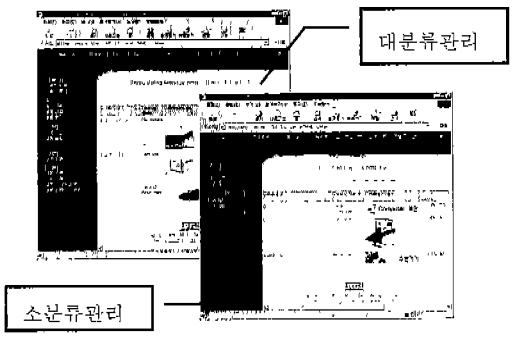

그림 8. 관리자 화면

이밖에 관리자는 StockBean 엔터티 빈을 이용해 상품의 재고관리를 하며, Order 세션 빈 등을 이용해 주문 내용을 확인 후, Delivery 세션 빈과

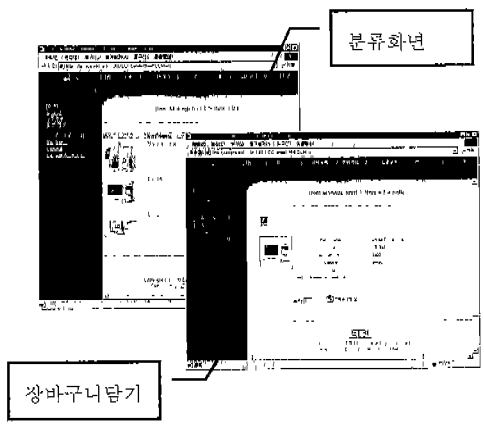

그림 9. 고객측 화면

DeliveryBean엔터티 빈등으로 상품의 배송 관리를 하며, ReceiptMoney와 ReceiptMoney-Bean으로 입금처리 등을 할 수 있다.

6. 고객측 컴포넌트

고객이 쇼핑 몰에 접속을 하면 초기화면으로 ItemCatalogBean엔터티 빈의 FindAll() 메쏘드를 수행한 결과인 상품 대분류가 나타난다. 이 대분류 중 구매하고자 하는 상품의 분류를 선택하면, ItemCatalogBean에 FindByPrimayKey() 메쏘드에 의해 관리자가 설정한대로 그림 9와 같이 중분류/소분류로 들어갈 수 있다. 이렇게 분류를 검색해 나가서 최종적으로 상품을 선택하면 상품에 대한 자세한 설명을 볼 수 있고, 물건을 장바구니에 담을 수 있다.

상품을 담아놓는 장바구니 정보는 고객측 브라우저에 쿠키[16]정보로써 저장이 되며, 상점에 회원으로 로그인 하지 않은 경우에는 손님의 자격으로 장바구니 정보를 저장하고 있다. 실제 주문을 내리기 위해서는 상점에 로그인을 해야하는데, 그림 10과 같이 상점에 로그인을 하면 로그인 정보를 쿠키에 설정하여 손님자격의 장바구니가 회원자격의 장바구니로 바뀌며, OrderHeaderBean과 OrderDetailBean 엔터티 빈을 이용해 실제 주문을 내릴 수 있게 된다.

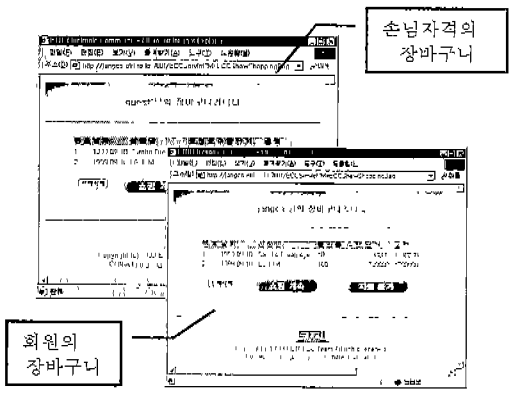

그림 10. 고객의 장바구니

V. 결 론

본 논문에서는 현재 업계의 웹 애플리케이션 서버 시장의 표준으로 자리를 잡아가고 있는 엔터프라이즈 자바 기술과 이 기술의 핵심이라고 할 수 있는 엔터프라이즈 자바 빈즈에 관해 살펴 보았다. 또한 엔터프라이즈 자바 빈즈를 간단한 인터넷 쇼핑몰 시스템에 적용한 예에 대해서 살펴 보았다.

기업의 전산 환경은 기술의 발전과 사용자의 요구에 따라 점차 복잡해지고 다양한 종류의 서버측 서비스를 요구하고 있다. 선의 엔터프라이즈 자바 기술은 이러한 기술적 요구 추세에 부응할 수 있는 하나의 대안으로 생각되며, 앞으로 기업의 전산화에 많은 기여를 할 것으로 생각된다.

그러나 엔터프라이즈 자바 기술은 가장 기본이 될 수 있는 분산 객체 플랫폼 만을 제시하였으며, B2B 전자상거래 분야와 같은 대형 인터넷 정보 시스템에 적용하기 위해서는 아직도 많은 부분이 충족되어야 할 것이다. 특히, 웹과 EJB 컨테이너의 성능 및 안정성을 위한 지원과 웹 관련 자원들의 실시간 부하 분산 기능과 같이 다중 서버 플랫폼에 대한 시스템 차원의 지원과 이를 위한 서버측의 구현 방법론이 정립되어야 할 것이다.

※참고 문헌

[1] 한재선 외 3인, "웹 애플리케이션 서버 : 인터넷 전자상거래를 위한 공용 플랫폼", EC/CALS 기술 워크샵 논문지, pp. 63~69, 1999.

[2] 박대연, 한재선 외 2인, "엔터프라이즈급 웹 애플리케이션 서버 하부구조 개발에 관한 연구 중

간보고서", 한국과학기술연구원, 한국전자통신연구원, 1999
[3] 장철수, 김성훈 외 2인, "엔터프라이즈 자바 빈즈를 기반으로하는 인터넷 쇼핑몰의 UML에 의한 설계," 한국산업공학회 추계학술 대회 논문집, pp247~251, 1999
[4] "Java 2 Platform Enterprise Edition Specification, V1.2," Sun Microsystems , 1999.
[5] "Java Transaction API, V.1.0.1," Sun Microsystems, 1999
[6] "Java Naming and Directory Interface V1.1," Sun Microsystems, 1998
[7] "Java Sevlet Specification, V2.2," Sun Microsystems, 1999
[8] "JavaServer Pages Specification V1.1," Sun Microsystems, 1999
[9] "Enterprise JavaBeans Specification V1.1," Sun Mirosystems, 1999
[10] "JDBC 2.0 API," Sun Microsystems, 1998
[11] Anne Thomas, "Java 2 Platform Enterprise Edition," Sun Microsystems, 6 1999
[12] 민덕기, "비즈니스 서버 컴포넌트의 재사용 지원플랫폼의 수립 및 프로토타입 개발에 관한 연구 중간보고서," 건국대, 한국전자통신연구원
[13] 박재현, "전자상거래용 분산컴포넌트 개발에 관한 연구 중간연구보고서," 에이젠텍, 한국전자통신연구원, 1999.
[14] Terry Quatrani, "Visual Modeling With Rational Rose and UML," Addison Wesley, 1998.
[15] Object Management Group, "The Common Object Request Broker Architecture and Specification Revision 2.2," 1998.
[16] "Persistent Client State HTTP Cookies", Netscape Communications Corporation,

김 성 훈

1997년 광운대학교 (석사-멀티미디어 정보처리)
1995년 광운대학교 (학사-전자공학)
1996~1998년 시스템 공학연구소 연구원
1998~현재 한국전자통신연구원 연구원
관심분야 : 전자상거래, 분산시스템, 데이터마이닝

장 철 수

1997년 광주과학기술원 정보통신공학과 (공학석사)
1995년 인하대학교 전자계산공학과 (공학사)
1997년~1998년 시스템공학연구소 연구원
1998년~현재 한국전자통신연구원 연구원
관심분야 : 데이터베이스, 분산시스템, 전자상거래

함 호 상

1995년 고려대학교 박사
1983년 고려대학교 석사
1977년 고려대학교 산업공학과 학사
관심분야 : CALS/EC분야, ERP분야, 콤포넌트기술분야 등