사례 발표

웹 환경에서의 미들웨어 적용

최 종 윤*, 최 종 태*

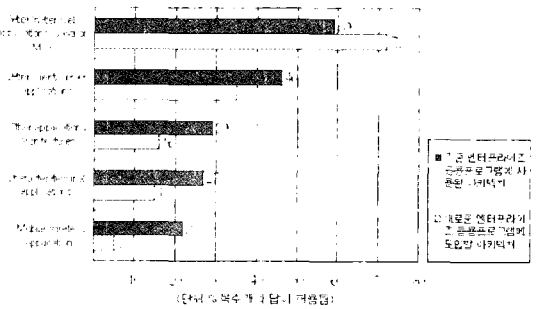
● 목 차 ●

1. 서 론

1990년 초반 클라이언트/서버 컴퓨팅은 적은 비용으로 보다 많은 시스템을 구축할 수 있는 방법으로 많은 사용자들로부터 각광을 받았다. 클라이언트/서버 컴퓨팅에서 필요한 주요기술이 미들웨어이다. 미들웨어는 Scalability, Transaction, Security, Load-Balancing, Fail-Safe 등의 다양한 기능을 Middle-tier에서 제공해 줌으로써 개발자는 비즈니스 로직만 신경 써서 개발할 수 있는 환경을 마련하였다.

그러나 2-tier 모델의 클라이언트/서버 어플리케이션은 유연성이나 확장성 측면에서 사용자들의 요구사항을 만족시키지 못했고, 많은 유지보수 비용을 필요로 하였다. 업체들은 오랫동안 이와 같은 혼란으로부터 빠져 나올 수 있는 방법을 강구하였고, 마침내 찾아낸 돌파구가 3-tier 구조로의 이행과 클라이언트를 가볍게 작성하는 것이었다.

1990년대 중반을 기점으로 비약적으로 발전하기 시작한 인터넷과 웹 환경은 기업 전산 환경을 다시 한번 크게 바꾸어 가고 있다. 기업들은 자신들이 이용하던 기존의 클라이언트/서버 시스템을 웹 환경을 이용하여 보다 편리하게 접근하기를 원했다. 특히 웹 기술은 클라이언트/서버 환경에서 문제가 되었던 클라이언트의 유지보수 문제를 완벽히 해결해 주었다. 웹 어플리케이션 서버(이하 WAS)는 웹 환경에 적합한 미들웨어로 최근 각광을 받고 있다. 클라이언트부터 데이터베이스에 이르는 전체 시스템을 일관적인 모습으로 손쉽게 구축할 수 있는 환경을 제공하여 최근의 IT 시장은 열풍이라고 할 정도로 경쟁적으로 WAS를 도입하고 있다. 현재의 WAS도입 상황은 (그림 1)과 같다.



(그림 1) WAS시장동향
출처: Giga, 2001 [1]

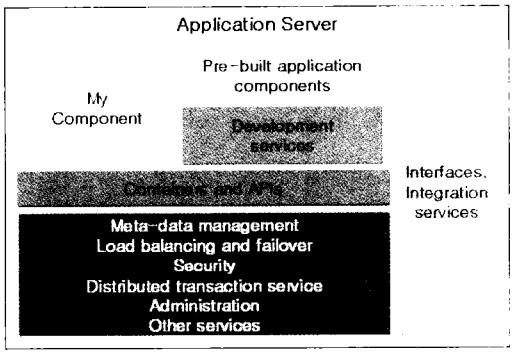
* LG-EDS시스템 기술연구부문 연구원

본 원고에서는 WAS에 대한 시장 동향과 이를 이용한 시스템 구축에 대해 이해를 돕고자 한다. 2장에서는 WAS의 특징 및 시장 동향을 설명하며 3장에서는 WAS를 이용하여 시스템을 구축하는 경우의 유형을 설명하도록 한다.

2. 기술현황

2.1 WAS의 특징

WAS는 기존의 미들웨어가 가지고 있던 기능뿐만 아니라 Http상에서 필요한 모든 기능들을 을 수행한다. WAS가 제공해줘야 할 기능들은 (그림 2)와 같다. 반드시 언급된 모든 기능들을 제공해야 한다는 것은 아니다. 하지만, 아래의 모든 기능들에 대한 지원은 점점 더 필수적인 요건이 되어 가고 있다.



(그림 2) WAS의 기능
출처 : Ovum, 2001 [2]

2.1.1 Client's Connectivity

WAS는 클라이언트가 접근할 수 있는 방법을 제공해 줘야 하는데 현재는 분산 객체 기술과 웹을 이용하는 방법이 흔히 이용된다. 분산 객체 기술은 다시 세 가지 기술로 요약될 수 있는데 CORBA의 IIOP (Internet Inter-ORB Protocol), 자바의 RMI (Remote Method Invocation), 마이크로소프트사의 COM/DCOM(Component Object Model/Distributed Component Object Model)이다. 웹을 이용하는 방법은 클라이언트로 웹 브라우저를 이용하는 것이다. WAS는 웹 브라우저가 접근할 수 있도록 하기 위해서 HTTP를 지원해야 한다. WAS가 HTTP를 지원할 경우 웹 서버처럼 단순히 HTML 문서를 전송하는 것 이외에도 HTTP를 통해서 Component들을 이용할 수 있어야 한다. 최근에는 XML기반의 웹 서비스를 위해 SOAP(Simple Object Access Protocol), WSDL(Web Service Description Language), UDDI (Universal Description, Discovery, Integration)을 지원하는 것이 새로운 추세이다.

2.1.2 Database Connectivity

대부분의 경우, Component들은 비즈니스 로직의 처리를 위해서 데이터베이스를 이용하기 때문에 WAS는 Component들이 데이터베이스를 접근할 수 있는 방법을 제공해 줘야 한다. 성능 향상을 위해서 데이터베이스로의 Connection Pool도 제공해 줄 필요가 있다.

2.1.3 Component Model

Component 모델은 어플리케이션의 재사용성을 증가시키기 위해서 필요하다. 현재 Component 모델은 세 가지로 요약될 수 있는데 CCM(Common Component Model), EJB와 마이크로소프트의 COM/DCOM이다. WAS는 이들 중에서 하나 이상을 지원해야 한다.

2.1.4 Component Manager

Component Manager는 Component를 관리해 주는 논리적인 요소로 Component의 비즈니스 로직을 수행할 때 Transaction, Security, Load-Balancing, Fail-Safe 등의 기능을 처리한다. Component Manager에 대한 표준으로 CORBA, EJB, COM+ 등이 있다. 보통 Component 모델과 Component Manager는 하나의 표준으로 정의되어 있는데

CORBA, EJB, MTS 등은 모두 Component 모델인 동시에 Component Manager에 대한 표준이라고 할 수 있다.

2.1.5 Transaction Service

WAS는 Component의 트랜잭션을 처리하기 위해서 트랜잭션 서비스를 제공해야 한다. 트랜잭션 서비스 표준으로 CORBA의 OTS (Object Transaction Service), 자바의 JTS (Java Transaction Service) 등이 있다. 마찬가지로 MS도 나름대로의 트랜잭션 서비스를 제공하고 있다.

2.1.6 Security Service

WAS는 클라이언트에 대한 인증 (Authentication)과 클라이언트의 권한 (Authorization)을 조사할 수 있는 서비스를 제공해야 한다. 뿐만 아니라, 클라이언트와 WAS가 암호화된 데이터를 주고 받을 수 있도록 해야 한다.

2.1.7 Management

WAS는 다양한 Component들을 관리한다. WAS는 이 Component들의 상태를 모니터하고 제어할 수 있는 방법을 시스템 관리자에게 제공해야 한다.

2.1.8 Integrated Development Tool

통합개발환경이 WAS가 가져야 할 기능인가에 대해서 아직 논란이 많다. 그러나 분명한 것은 WAS가 등장한 이유는 시스템 개발에 있어서 통합의 의미가 강하게 부각되어 왔었고, 현재 대부분의 WAS 제품들은 통합개발환경을 제공해 주고 있다는 사실이다. 일부 제품은 다른 Third-Party 통합개발환경을 권장하기도 한다. WAS는 통합개발환경을 이용해서 서버의 Component와 클라이언트를 쉽게 개발할 수 있도록 해야 한다.

2.2 시장 현황

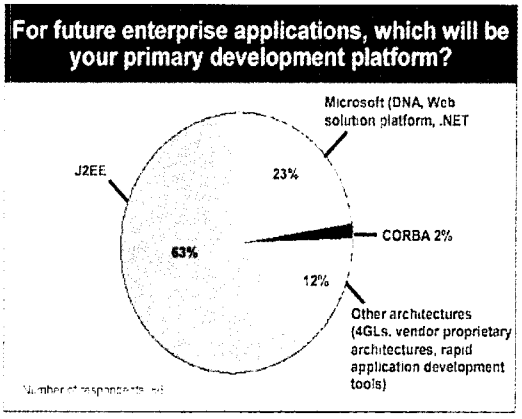
2.2.1 표준 플래폼

업체마다 독자적인 규격의 미들웨어를 제공했던 과거와는 달리 WAS업체들은 표준플래폼에 따르는 제품을 제공하여 시스템의 호환성을 높여주고 있다. 대표적인 표준플래폼이 J2EE와 .NET이다.

Microsoft에서는 COM/COM+, WindowsDNA, ASP 등 기존의 기술에 CLR(Common Library Runtime), XML, WebService 등의 신기술을 접목하여 .NET 플래폼을 발표하였다. 분산 어플리케이션에 필수적인 요소들을 제공하며 자바 플랫폼과 경쟁하고있다.

Sun Microsystems사에서는 대규모 IT환경에 적용되는 자바 기술들을 정리하여 Java 2 Enterprise Edition(이하 J2EE)을 발표하였다. 단일화된 J2EE 표준은 기존의 N-tier 응용프로그램에서 요구되는 데이터베이스 시스템, 트랜잭션 모니터, 디렉토리 서비스 등을 콤포넌트기반의 모델로 제공한다.

.NET과 J2EE의 장단점비교에는 많은 논쟁의 소지가 있다. 이 두 기술은 상당한 차이점을 갖고 있어서 두 가지를 동시에 언급하기에는 어려움이 있다. 개략적인 시장점유율에 대한 예측은 (그림 3)과 같다.

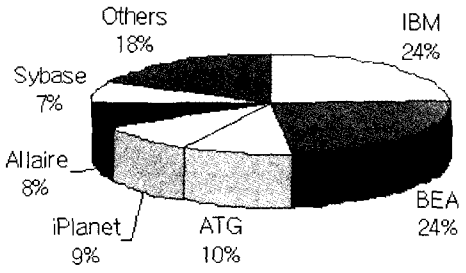


(그림 3) WAS시장 동향
출처 : GigaWeb, 2001 [3]

본 원고에서는 현재 대형 SI프로젝트에서 많이 사용되고 있는 J2EE기반의 WAS에 한정하여 시스템 구축유형을 서술하도록 한다.

2.2.2 J2EE환경

Java기반의 WAS시장은 초기에 CORBA를 기반으로 하였으나 최근 들어 JSP, Servlet, EJB의 지원 중심의 J2EE로 급속히 옮아가고 있다. J2EE지원 WAS 제품군은 매우 다양하게 존재하나 IBM과 BEA가 시장을 선도하고 있다. IBM은 기존의 Legacy와의 연결성에서 BEA는 J2EE 지원에서 강점을 갖고 있다.



2000년 Market Share

(그림 4) WAS 시장동향

출처 : Giga, 2000 [4]

J2EE를 사용하여 시스템을 작성할 때 JSP, Servlet, EJB등을 사용하는데 요소 기술들의 조합에 따른 다양한 경우의 수가 발생할 수 있다. 기술도입의 초기에 개발자들은 다양한 조합의 경우 때문에 비효율적인 시스템 설계에 빠지기 쉽다. 이러한 어려움을 해결하려면 요소 기술들의 특징들에 대한 이해가 필요하며, 개발자의 높은 숙련 도를 요구한다.

시스템에 존재하는 불확실 요소들을 사전에 점검하여 이를 반영한 표준 Framework을 개발자에게 제공할 경우, 재사용성이 향상되어 반복되는 오류를 줄일 수 있게 된다. 검증된 Framework의 사용은 안정적인 시스템을 단시일 내에 개발을 가능하게 한다. 실제 J2EE기반의 WAS로 시스템을 구축하는데 있어서 적용할 수 있는 Framework 유형에 대하여 다음 장에서 설명하도록 한다.

3. 시스템 구축

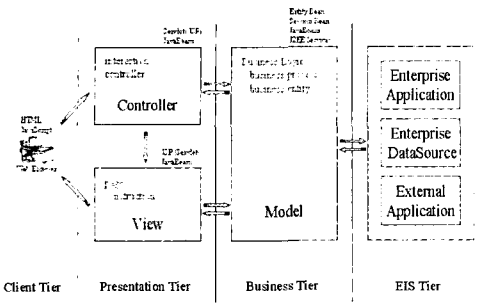
3.1. 주요 구성 요소

요즈음의 분산시스템은 3-Tier또는 n-Tier의 구조를 갖는다. 3-Tier 아키텍쳐는 Presentation Tier, Business Logic Tier, EIS(Enterprise Information System) Tier로 이루어져 있다. 각 Tier는 자신들에게 맡겨진 역할을 서로 독립적으로 수행될 수 있도록 설계되어야 한다. 하나의 Tier에서 일어난 변동이 다른 Tier에게 미치는 영향을 최소화 해야 한다. 각 Tier의 역할과 이에 적용되는 J2EE Technology는 <표 1>과 같다.

<표 1> J2EE 기술분류 출처 : java.sun.com [5]

	responsibility	J2EE technology
Client-side Presentation Tier	• rendering UI • user input handling • user-level validation	• HTML, DHTML • JavaScript
Server-side Presentation Tier	• receive client request • interpret client request • invoke appropriate Business Logic • package data for client • send response	• JSP • Servlet • JavaBeans Component
Business Tier	• Business Process • Representation of Business Entity • interface with EIS Tier	• EJB • JavaBeans Component • J2EE Service (JMS, JTA, JCA, JAF)
EIS Tier	• implement Legacy System	• Database Server • Transaction Monitor • Mainframe Server • ERP, etc

주요 설계사상은 코드와 화면설계의 독립성 유지이다. Enterprise Application 개발 시 Web Page Design과 Layout설계 및 생성, Application Workflow 개발, Business Process 및 Business Entity 개발을 서로 독립적으로 동시에 진행할 수 있도록 한다. MVC(Model - View - Control)구조는 이러한 설계사상을 잘 반영할 수 있는 구조이다. (그림 5)는 MVC 구조를 보여주고 있다.



(그림 5) MVC구조
출처: 기술 내재화, 2000 [8]

Controller의 가장 중요한 역할은 Client Web Browser로부터 요청된 HTTP Request를 해석해서 이를 처리할 해당 Business Logic을 결정하여 호출하고 처리결과에 따라 적절한 Web Page 생성 컴포넌트를 선택하여 처리결과를 넘겨주는 것이다. HTTP Request는 HTTP 프로토콜에 의존적이기 때문에 이를 프로토콜 독립적인 Business Logic이 이해할 수 있는 형태로 변환해야 한다. HTTP Request와 Controller는 보통 다음의 세가지 중 하나의 유형을 사용한다.

- One HTTP request to One Controller
- One HTTP request Group to One Controller
- All HTTP request to One Controller

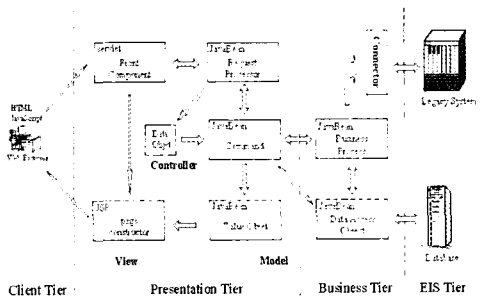
View는 Client Web Browser에 표시되는 HTML Page를 생성하는 역할을 한다. Page 생성에 필요한 동적 데이터는 해당 Page 생성시 직접 Business Logic에 접근하여 가져올 수도 있으나, Controller가 Business Logic을 호출한 후 그 처리 결과중의 하나로 동적데이터를 생성하여 Page 생성 컴포넌트에 넘겨주도록 설계하는 것이 바람직하다. Page 생성 컴포넌트는 동적데이터를 적절한 형태로 변환시켜 사용자에게 보여주는 방식을 구현하는 Page Design Logic이 주를 이룬다. 이는 Page Desinger의 작업이 쉬워지고 차후에 Page Design 상의 변동에 대한 요구가 있을 경우 유지보수가 쉬워진다는 장점이 있다. J2EE Application 모델에서는 HTML page 생성시 JSP를 사용하도록 제안하고 있다.

Model은 Enterprise Application의 핵심이라 할 수 있는 Business Logic이 구현되는 부분이다. 모든 Business Entity들과 이를 가지고 구체적인 업무를 수행하는 Business Process들이 Model로서 구현된다. Model은 기업의 Mission Critical한 업무를 포함하기 때문에 트랜잭션, 보안, 기존 시스템과의 통합 등을 고려하여 설계되어야 한다. J2EE Application 모델에서는 Enterprise JavaBeans를 제안하고 있으며 Business Entity나 Business Process 컴포넌트 작성시 이를 사용하도록 권장하고 있다.

EJB가 Enterprise 환경의 Application을 개발하는데 유용한 기술이나 컴퓨팅 자원을 많이 소모하고 Network이나 서버 용량 등에 대한 고 수준의 스펙을 요구하기 때문에 현실적으로 많은 제약이 존재한다. HTTP가 기존의 미들웨어가 담당하였던 분산의 역할을 대행할 수 있어서 상당수의 시스템이 EJB없이 Servlet/ JSP 기술만을 사용하여 구축하고 있는 실정이다. EJB를 도입하는 경우에도 객체설계와 관계형 데이터베이스와의 연계성 때문에 대다수의 시스템이 Session Bean만을 사용하여 구축되고 있다. 시스템의 설계는 크게 EJB를 사용하지 않는 경우와 사용하는 경우로 분류할 수 있다.

3.2 웹 UI만을 적용하는 구현 모델

Servlet과 JSP 기술만을 사용한 아키텍처는 아래와 같이 MVC 아키텍처를 바탕으로 한다.



(그림 6) JSP/Servlet만의 모델
출처: 기술 내재화, 2000 [8]

아키텍쳐를 이루는 각 컴포넌트가 맡은 기능과 역할은 다음과 같다.

3.2.1 Front Component

Client Tier로부터 요청되는 HTTP request를 Request Processor에 넘겨주고 Request Processor로부터 Page Constructor에 대한 정보를 받아서 해당 Page Constructor에 프로그램 제어를 넘겨준다. 분석 단계에서 도출된 Use case별로 작성하도록 하며 Servlet으로 구현한다.

3.2.2 Request Processor

Request Processor가 담당하는 첫번째 작업은 사용자의 인증 여부나 권한 검사 등의 Application 전체에서 공통적으로 수행되어야 할 작업들이다. 또한 HTTP request에 대하여 문법적인 타당성을 확인하고 request Data 형식상의 전환작업을 거친 후 적절한 Command를 결정하여 해당 request를 수행하도록 한다. 마지막으로 Command 수행 결과에 따라 적절한 Page Constructor를 선택하여 이에 대한 정보를 Front Component에 넘겨준다. 일반 자바 클래스나 자바빈즈로 구현한다.

3.2.3 Data Object

대부분의 Business Process는 클라이언트로부터 넘어온 정보에 근거하여 작업을 수행한다. 이렇게 작업 수행을 위한 데이터로서 넘어갈 파라미터들이 많아지면 이를 하나하나 인자로 선언하는 것보다 하나의 Object로 만들어 넘겨주는 것이 더 효율적이며 유지보수하기에도 편리하다. 이러한 Object들을 Data Object 또는 Value Object라고 하며 Request Processor가 생성하여 적절한 Command에 넘겨주게 된다. Value Object는 Method의 인자로서 전달 가능하도록 serialization이 가능해야 하며 일반 자바 클래스나 자바빈즈로 구현한다.

3.2.4 Command

각각의 HTTP request별로 작성되며 해당 request를 수행할 Business Process를 호출하거나 특별한 Business Process가 필요 없는 작업 에 대해서는 직접 Data Access Object에 접근하기도 한다. 또한 Business Process나 Data Access Object의 작업 수행 결과 생성된 Value Object를 Page Constructor가 참조할 수 있도록 해당 Object를 저장한다. 일반 자바 클래스나 자바빈즈로 구현한다.

3.2.5 Business Process

Application의 Business Logic이 구현되는 부분이다. Front Component와 마찬가지로 각 Use case별로 작성한다. 일반적으로 business process 만을 구현하도록 하며 business entity와 관련된 DB 관련 작업은 모두 Data Access Object에 이관하도록 한다. 일반 자바 클래스나 자바빈즈로 구현한다.

3.2.6 Data Access Object

Application의 모든 DB 관련 작업을 담당한다. DB 작업을 Business Process에서 분리시켜 Data Access Object에서 처리함으로써 차후에 Database 서버 환경이 변경될 경우 유연하게 대처할 수 있게 되며 Application이 특정 DB에 종속되지 않는다. 일반 자바 클래스나 자바빈즈로 구현한다.

3.2.7 Connector

CICS나 틱시도 등 Legacy System과의 연결을 구현하는 부분이다. Legacy 시스템에 대한 Wrapping을 담당한다.

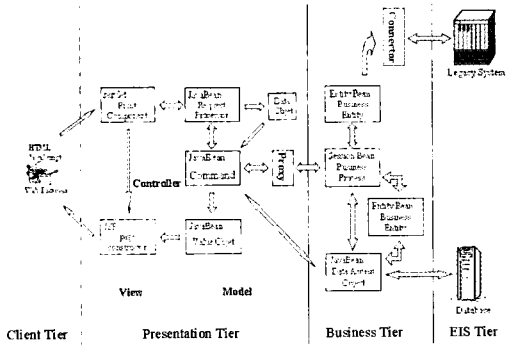
3.2.8 Page Constructor

사용자에게 표시되는 HTML page를 생성하는 JSP page이다. Value Object를 참조하여 동적 데이터를 얻어내며 이를 적절한 형식으로 변환시켜 HTML을 생성해 낸다. 일반적으로 JSP page는 프로

그래머보다는 Web Page 디자이너에 의해 더 많이 다루어지므로 최소한의 코딩 만을 사용하도록 한다.

3.3 EJB를 적용하는 구현 모델

EJB를 사용한 아키텍쳐는 3.2.절에서 제시된 Servlet/JSP 만을 이용한 아키텍쳐를 바탕으로 하여 설계되었다. 두 아키텍쳐를 비교해 보면 알 수 있겠지만 Proxy를 제외하면 Presentation Tier는 동일한 구조를 가지고 있다. Presentation Tier내의 각 컴포넌트들의 기능과 역할도 역시 동일하다. Business Tier를 보면 Business Process 컴포넌트가 Session Bean으로 구현 방식이 바뀌었고 Business Entity를 나타내는 Entity Bean이 새로이 추가 되었다.



(그림 7) EJB사용 모델
출처: 기술 내재화, 2000 [8]

다음은 EJB를 사용한 아키텍쳐에서 변경되거나 추가된 컴포넌트들에 대한 설명이다.

3.3.1 Business Process

Servlet/JSP 만을 이용한 아키텍쳐의 Business Process와 동일한 역할을 담당하나 Session Bean으로 구현된다. 분석 단계에서 도출된 Use case별로 작성되며 해당 Use case에서 각 단위 업무를 수행하기 위하여 필요한 속성(attribute)들을 member 변수로 저장할 필요가 있을 경우에는 Stateful Session Bean을 사용한다.

3.3.2 Proxy

EJB의 설계 사상 중 하나가 분산 컴포넌트의 지원이므로 특정 EJB를 참조하기 위해서는 본질적으로 Remote Access 방식을 취한다. 이는 EJB 클라이언트가 EJB 컴포넌트와 동일한 JVM내에 존재하더라도 마찬가지이다. 따라서 EJB를 참조하는 클라이언트는 필요한 시점에서만 EJB 객체에 대한 Reference를 획득하고 일단 참조된 EJB 객체 Reference는 차후 사용을 대비하여 Caching하는 것이 Application 전체적인 성능 향상을 가져온다. 이를 가능하게 하는 것이 Proxy 이며 EJB 클라이언트로부터의 EJB method 호출 시 이를 중간에서 가로채어 해당 EJB 객체 reference를 얻어내서 EJB method 호출을 대행하는 기능을 한다.

3.3.3 Business Entity

DB 서버의 레코드를 Application 서버에 캐싱하기 위하여 Business Entity를 사용한다. DB에 저장된 레코드를 접근하기 위하여 매번 DB 서버와 연결을 맺어 레코드를 조회하지 않고 Application 서버에 캐싱된 데이터를 사용하면 Application의 성능은 물론이고 유연성이나 재사용성이 크게 향상된다. 그러나 DB 서버의 레코드와의 동기화, Transaction, Concurrency, Life Cycle 관리 등을 모두 고려하여야 하므로 Business Entity를 일반 자바 클래스로 구현하는 것은 너무나 많은 시간과 비용을 초래한다. 이를 위한 솔루션으로 J2EE Application 모델에서는 Entity Bean을 사용하도록 제안하고 있다.

4. 결 론

현재 너무나 많은 벤더들이 WAS 시장을 두고 경쟁을 벌이고 있고 이 벤더들 사이의 인수, 합병

도 자주 발생하고 있다. WAS 시장은 IT 산업에서 중요한 시장이 될 것이며 많은 시스템들이 WAS를 이용해서 개발될 것이다. 많은 e-business 응용프로그램이 Portal, CRM, SCM, Mobile등의 다양한 솔루션을 통합 지원하는 구조로 발전하고 있으며 이를 위한 인프라로서 WAS가 중요하게 사용되고 있다. 새로이 떠오르는 웹서비스(SOAP, WSDL, UDDI)의 기반 시스템으로도 저변을 넓혀가고 있다. 그러나 WAS는 기존 기술을 많이 통합시키기도 했지만 새로운 기술들도 많이 적용되었기 때문에 아직은 기술의 안정화와 검증 과정이 필요하다.

참고문헌

[1] Giga Information Group, N-Tier Architectures Dominate, but Others Still in Use, Randy Heffner, 2001/2/23

[2] Ovum, Ovum Evaluates Application Servers/ Introducing Application Server, Christine Axton, Gary Barnett, Neil Ward-Dutton and Bola Rotibi, 2001.1

[3] Gigaweb, J2EE and Server-Based Components Showing Strong User Support, Randy Heffner, 2001/2/21

[4] Giga, 2000 Forecast for the EJB Application Server Market, Mike Gilpin and Carl Zetie, 2000/6/19

[5] http://java.sun.com/j2ee/

[6] J2EE Technology in Practice, Rick Cattel, Addison Wesley, 2001,6

[7] Applying Enterprise JavaBeans, Matena, Addison Wesley, 2001

[8] 2000년 Application Architecture 기술내재화 보고서, LG-EDS시스템, 2000

저자약력

최 종 윤

1995년 성균관대학교 산업공학과(공학사)
1998년 광주과학기술원 기전공학과(공학석사)
1998년-현재 LG-EDS시스템 기술연구부문
관심분야 : OO, CBD, WAS

최 종 태

1989년 충남대학교 기계공학과(공학사)
1990년-1995년 대우정보시스템
1995년-현재 LG-EDS시스템 기술연구부문
관심분야 : OO, CBD