<JSTORM>

Enterprise Java Beans 1부

중앙대학교 컴퓨터공학과
자바 동호회
JSTORM
http://www.jstorm.pe.kr

Document Information

Document title:	Enterprise Java Beans 1부
Document file name:	EJB1_JSTORM.doc
Revision number:	<1.0>
Issued by:	<박지훈> pjh@jstorm.pe.kr
	<남궁윤> yoonng@hanmail.net
	<김종윤> foxkjy@chollian.net
	<전부현> kanaloa@orgio.net
	정리 : <윤준호> junoyoon@orgio.net
Issue Date:	<2000/9/7 >
Status:	final

Content Information

Audience	EJB 프로그래밍을 학습하는 중급 JAVA 프로그래머
Abstract	Enterprise Java Beans가 도대체 무엇인가에 대한 개요와 간단한 Java Beans 프로그래밍을함으로써 Component의 개념을 살펴본다. 또한 각종 J2EE 서버들을 살펴보고 설치방법을 알아본다.
Reference	1) Enterprise Java Bean, Second Edition (Oreill'y)
	2) Mastering EJB (Wiley)
	3) http://www.itplus.co.kr 자료
	4) http://www.bea.com의 자료
	5) http://www.javasoft.com의 자료
	6) http://www.javaworld.com의 자료
	7) Sun사의 EJB 스펙 1.1 PDF파일
	8) J2EE SDK의 example9
	9) J2EE blueprint 및 java pet store example
Benchmark information	

Document Approvals

고문	Signature	date
		
지위	Signature	date

Revision History

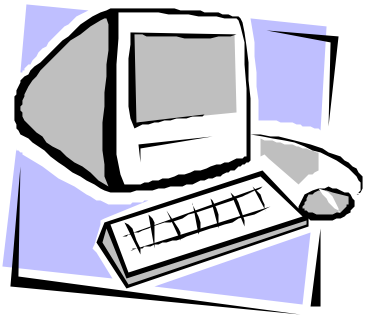
Revision	Date	Author	Description of change

Table of Contents

1 부

Enterprise Java Beans란 무엇인가?

첫 번째 장에서는 Enterprise Java Beans가 도대체 무엇인가에 대한 개요를 살펴볼 것이고 간단한 Java Beans 프로그래밍을 함으로써 Component의 개념을 살펴볼 것이다.
또한 각종 J2EE 서버들을 살펴보고 설치방법을 알아본다.



Chapter 1

Java Beans 와 Enterprise Java Beans

EJB 소개

Sun사가 제안한 자바 서버 컴포넌트 모델

Sun이 내린 EJB 정의

"EJB 아키텍쳐는 객체지향 분산 엔터프라이즈 애플리케이션의 개발 및 분산 배치를 위한 컴포넌트 아키텍쳐이다. 엔터프라이즈 자바빈즈 아키텍쳐를 이용해 만들어진 애플리케이션은 확장성이 있거나 트랜잭션을 보장하며, 다수 사용자 환경에서도 안전하다. 이들 EJB 애플리케이션은 한 번 작성되면 엔터프라이즈 자바빈즈 스펙을 지원하는 어떤 서버 플랫폼에서도 배치되고 운영될 수 있다."

EJB는 제품이 아니라 스펙이다.

- 1998년 3월, EJB 1.0발표
- 1999년 , EJB 1.1발표 (본 교재의 기준)
- 2000년 7월 현재, EJB2.0 Draft

자바빈과는 어떤 관계가 있는가?

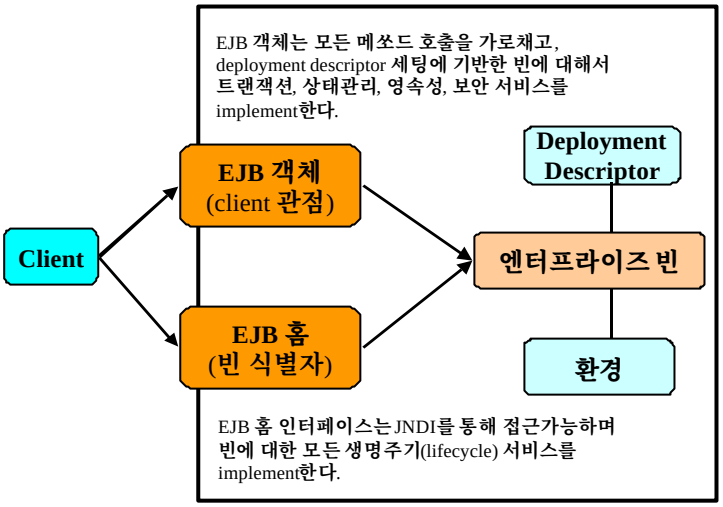
- 자바빈(JavaBean)은 클라이언트 쪽의 컴포넌트 모델이다.
- EJB는 서버 쪽 컴포넌트 모델이며, 이름만 유사할 뿐 직접적인 관계가 없다.

그럼 왜 EJB인가?

- 보안(Security), 패일오버(failover), 트랜잭션(transaction), 등의 시스템 레벨의 기능들을 서버 차원에서 지원함으로 프로그래머는 비즈니스 로직(Business Logic) 자체만 프로그래밍 하면 된다.

- EJB는 자바의 장점을 그대로 물려받고 있다. 즉 Java의 One-Write, Use Anywhere의 특징을 받아들여 "Write Once, Deploy Anywhere", 즉 한번만 작성하면 EJB Server Spec을 구현한 어떠한 CTM(Component Transaction Monitor)에서도 동작할 수 있는 것이다.
- 또한 대부분의 EJB Server는 Container-Managed Entity Bean이라는 것을 지원하여 EJB와 데이터베이스 연동을 아주 쉽게 한다.

EJB 컨테이너 모델 개요



클라이언트는 EJB홈을 통해 EJB객체를 얻고, EJB객체를 통해 EJB 빈을 사용할 수 있게 된다.

대표적인 EJB 컨테이너 제품들

- 웹로직 : BEA
- 웹스피어 : IBM
- 실버스트림 : SilverStream
- 젬스톤
- 기타

EJB 대 JavaBean

JavaBean 이란 무엇인가?

- Sun사의 자바빈즈 정의

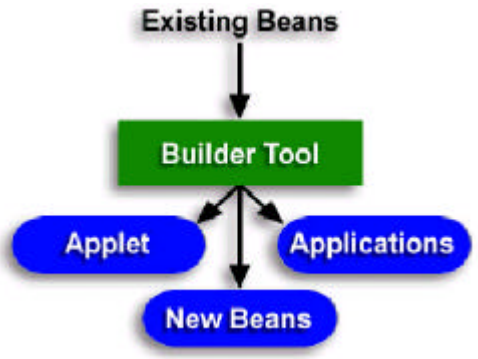
  자바 빈이란 개발자 도구에서 비주얼하게 수행되어질 수 있는 재사용 가능한 소프트웨어 컴포넌트이다. 이식성이 있고 플랫폼 독립적으로 실행되는, 자바언어로 작성된 소프트웨어컴포넌트 모델이다.

자바빈 API

✍✍ 자바 1.1이상 버전에서 자바빈을 만들 수 있는 API를 제공한다. 이 API는 java.beans패키지안의 클래스와 인터페이스로 이루어져 있다.

목적

✍✍ 자바빈의 목적은 어떤 플랫폼에서도 재사용가능한 소프트웨어 컴포넌트를 만드는 것이다. 그래서 개발자가 비주얼한 소프트웨어 컴포넌트인 자바빈을 자신의 어플리케이션에 맞게 적용하거나 수정하여 원하는 어플리케이션을 빨리 만들 수 있게 하는 것이다.



컴포넌트의 규모

✍✍ 컴포넌트들의 크기와 개발하고자 하는 어플리케이션의 관계를 분석하여 적당한 크기의 자바빈을 선택한다.

✍✍ 크기가 작은 컴포넌트

어떤 자바빈은 한 어플리케이션을 만들기 위한 기초 블록(Building Blocke), 즉 건물을 짓기 위해 사용되는 벽돌과 같은 역할을 한다. 즉 기존에 있는 크기가 작은 자바빈 들을 수정하거나 서로 결합하여 새로운 어플리케이션을 만들 수 있는 것이다.예를 들어 GUI컴포넌트의 버튼등이 이런 규모의 자바빈이 될 수 있을 것이다.

✍✍ 크기가 큰 컴포넌트

일반 어플리케이션도 자바빈 컴포넌트가 될 수 있다. 즉 워드프로세스, 스프레드쉬트 등의 일반 어플리케이션도 자바빈이 되어 기존의 복합문서(compound) 등에 삽입할 수 있다. 예를 들어 스프레드쉬트의 역할을 하는 자바빈 컴포넌트를 윈도우 95의 마이크로 소프트 워드나 파워포인트 등의 문서내에 삽입할 수 있는 것이다.

자바 빈과 클래스 라이브러리의 차이점

자바언어로 모든 소프트웨어 모듈들을 자바빈으로 만들 필요는 없다. 자바빈은 비주얼하게 다룰 수 있으며, 특정 목적을 위해 기능이나 데이터를 비주얼하게 변경할 수 있는 소프트웨어 컴포넌트에 적당하다. 반면 클래스는 비주얼하게 다룰 수 없고, 프로그램 소스상에서 특정 기능을 구현하기 위해 사용되는 소프트웨어 모듈에 적당하다.

자바빈의 특징

자바빈은 소프트웨어 컴포넌트이기 때문에 자체적으로 자신만의 데이터(속성)를 가져야 한다. 또한 외부에서 특정기능을 수행하도록 요청(이벤트)이 오면 그 요청을 수행(메쏘드)할 수 있어야 한다. 일반객체와 비슷한 점(속성과 메쏘드를 갖는다)이 있지만, 비주얼하게 다룰 수 있다는 것이 다르다.

자바빈이 가지는 이런 특징들은 크게 아래와 같이 3가지로 구분할 수 있다.

✍✍ **속성**(properties)

컴포넌트내와 연관된 데이터를 의미한다. 속성은 자바빈내에 정의되어 있는 메쏘드에 의해 그 값을 얻거나 변경할 수 있다. 예를 들어 자바빈 컴포넌트가 화면에 보여질 때의 배경색 정보를 담고 있는 속성 " background"를 가질 수 있다. 이 속성은 아마 자바빈내에 정의되어 있는 메쏘드 Color getBackground()" 또는 "void setBackground(Color c)" 같은 것에 의해 값을 얻거나 변경할 수 있을 것이다.

✍✍ **메쏘드**(method)

컴포넌트내에 정의되어 있고, 다른 컴포넌트나 개발환경이 호출할 수 있는 메쏘드를 의미한다. 즉 외부에 제공하는 서비스이다.

✍✍ **이벤트**(event)

특정 컴포넌트가 다른 컴포넌트에게 특정 사건이 발생되었음을 알리는 방식을 제공한다. GUI프로그래밍의 이벤트를 생각하면 될 것이다. 자바빈의 이벤트 처리방법은 GUI이벤트 핸들링 방식과 마찬가지이다. 즉 이벤트 소스객체에 이벤트 리스너 객체를 등록하여 이벤트를 처리한다.

JavaBean를 사용하기 위해 필요한 소프트웨어들

자바빈을 만들고 사용하기 위해 필요한 기초 소프트웨어들은 다음과 같다.

✍✍ JDK 1.1 **이상**

자바빈을 작성할 수 있는 API가 java.beans 라는 패키지 안에 있다. 이것을 사용하여 자바빈을 프로그래밍해야 한다. 따라서 JDK가 있어야 한다.

✍✍ BDK(Beans Development Kit)

원래 자바빈은 비주얼 자바개발환경에서 구현되어야 적당하다. 선사는 개발자가 자바빈을 JDK1.1 이상의 API로 만들었을 때, 제대로 동작하는 지 테스트할 수 있는 소프트웨어를 제공하게 되었다. 그 소프트웨어가 BDK이며, 이것은 자바빈을 만들어서 테스트할 수 있는 각종 실행화일과 예제가 포함되어 있다.

그러나 BDK는 자바빈을 개발하기 위한 목적의 소프트웨어는 아니다. 단지 현재 자바빈즈를 지원하는 개발환경이 대중화되어 있지 않기 때문에 등장한, 자바빈 테스트 용 소프트웨어 일뿐이다.

BDK는 현재 http://www.javasoft.com 에서 구할 수 있다.

BDK 설치 및 JavaBean 사용예제

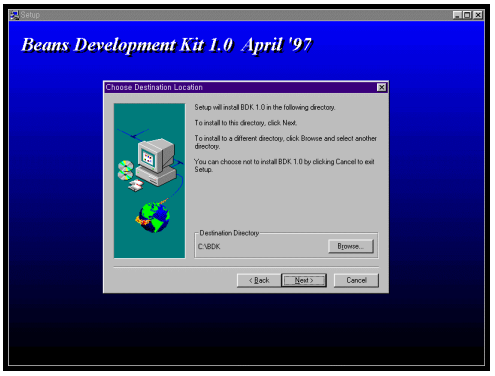
BDK는 순수하게 자바언어로 만들어진 어플리케이션이며, 반드시 JDK1.1이 설치

가 되어있어야 동작한다. 자바빈즈 API 및 자바빈 테스트용 컨테이너 프로그램, 예제 자바빈들을 소스코드와 함께 제공한다.

✍ 설치

윈도우 95환경에서 BDK를 설치하는 과정을 살펴보자.

먼저 윈도우95용 BDK 설치프로그램을 인터넷으로 다운로드 받아서, 실행시키면 다음과 같은 화면이 나타난다.



<그림> BDK 설치 화면

화면에 지시한대로 특정 디렉토리에 BDK를 설치한다.

✍ 내용

BDK가 포함하는 내용물은 다음과 같다.

- 자바빈즈 API 소스 코드 파일

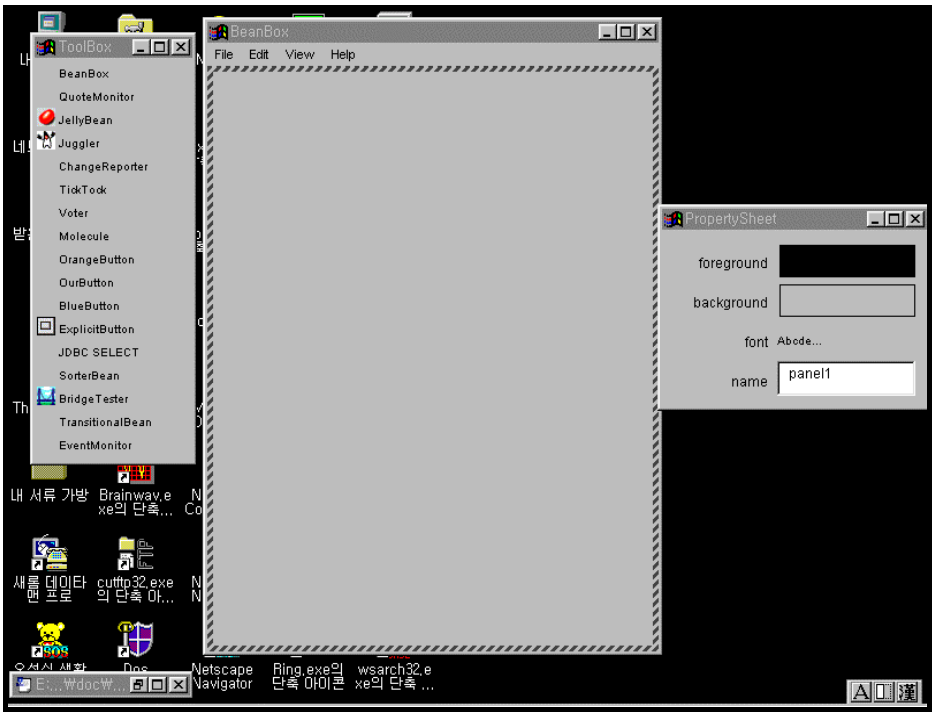
  자바빈을 만들기 위해 사용되는 클래스와 인터페이스의 소스코드를 apis라는 디렉토리에 저장하였다. 이 소스코드는 JDK1.1에도 포함되어 있다.

- 자바빈 컨테이너

  이것은 자바어플리케이션으로서 자바빈을 비주얼하게 화면에 보여주고 개발자가 다룰 수 있는 기능을 제공한다. 이 어플리케이션의 목적은 자바빈즈 API로 작성된 자바빈이 제대로 동작하는 지 테스트하는 것이다. 그래서 만약 비주얼 개발환경이 아닌 JDK1.1로 자바빈을 만들었을 때, 이것이 제대로 동작하는 지 확인하려면 자바빈 컨테이너를 실행하여 자신의 자바빈이 제대로 포함되는지를 살펴보아야 한다.

자바빈 컨테이너를 실행하면 다음 <그림>과 같은 화면이 나타난다.

3개의 윈도우 중에서 가운데 윈도우가 자바빈을 포함하고, 그것을 비주얼하게 다룰 수 있게 사용자에게 보여주는 역할을 한다.



<그림> 자바빈 컨테이너의 실행모습

- 예제 자바빈

자바빈 테스트 컨테이너안에서 동작할 수 있는 예제 자바빈을 제공한다. 이것들을 Jar파일로 압축되어 있다.

- BDK 사용 설명서

자바빈 테스트 컨테이너의 사용법과 자바빈의 개념을 설명하는 매뉴얼이 pdf, ps파일로 저장되어 있다.

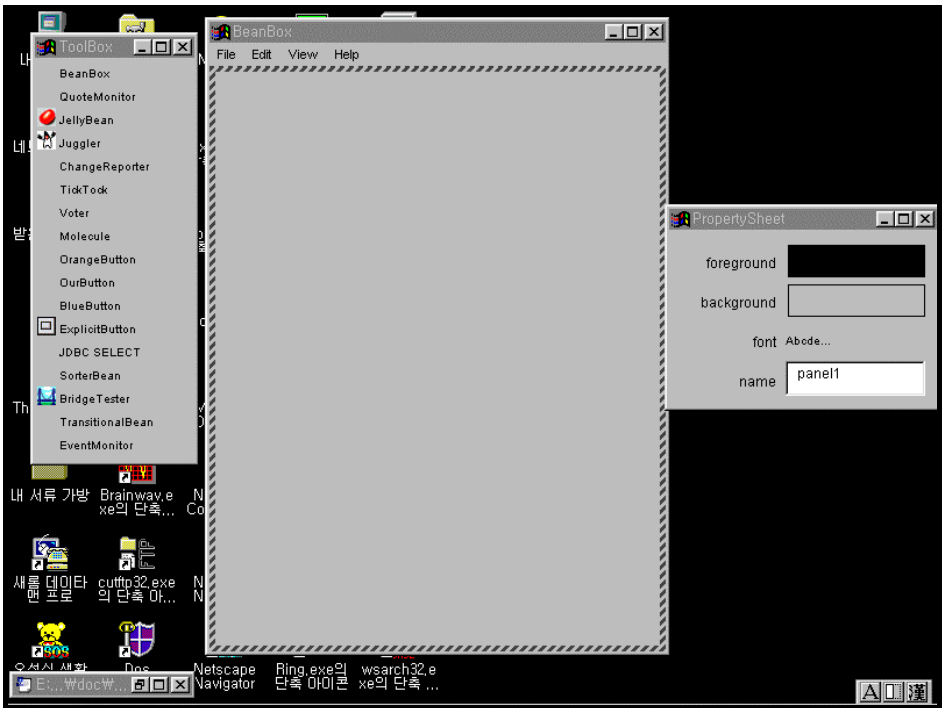
✍ BDK의 자바빈 테스트용 컨테이너의 사용방법

- 실행 방법

자바빈 테스트용 컨테이너 프로그램이름은 sun.beanbox.BeanBoxFrame이다. 이 프로그램을 실행시켜주는 배치화일이 존재하는데, 그것은 BDK가 설치된 디렉토리(본 예에선 d:\bdk)의 하위디렉토리인 beanbox디렉토리에 있는 run.bat이다. 즉 다음과 같이 실행하면 sun.beanbox.BeanBoxFrame 이 실행된다.

```
d:\bdk\beanbox>run.bat
```

그러면 다음 그림과 같은 화면이 생성된다.



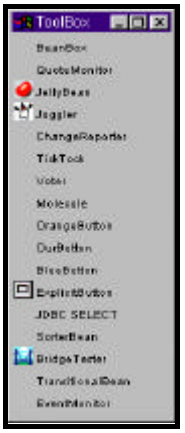
<그림> 자바빈 컨테이너의 실행모습

- 사용방법

총 3개의 윈도우가 생성되는데, 각자의 역할은 다음과 같다.

ㅇ 테스트 자바빈 툴박스(sample Java Bean ToolBox)

위 화면에서 맨 왼쪽에 생성된 윈도우로서, 테스트 할 자바빈들의 이름들이 나열되어 있는 윈도우이다.(다음 <그림> 참조)



<그림> 툴박스

BDK안에는 기본적으로 다음과 같은 자바빈들이 포함되어 있다.

QuoteMonitor, JellyBean, Juggler, ChangeReporter, TickTock, Voter,

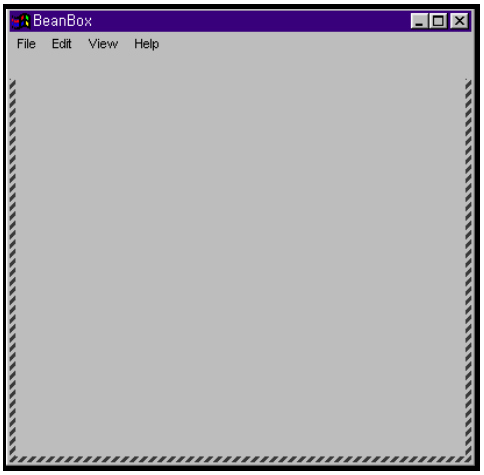
Molecular, OrangeButton, OurButton, BlueButton, ExplicitButton, JDBC SELECT, SorterBean, Bridge Tester, TransitionalBean, EventMonitor

이런 자바빈들은 각각 JAR파일로서 존재한다.

개발자들은 이런 툴박스에 나열되어 있는 자바빈을 마우스로 선택하여, 자바빈 컨테이너안에 포함시킬 수 있다.

- 자바빈 컨테이너(Java Bean containter)

위의 I-10의 <그림>의 가운데에 있는 윈도우로서, 테스트 자바빈 툴박스에서 사용자가 선택한 자바빈을 포함하는 윈도우이다.



<그림> 컨테이너

이 컨테이너의 구체적인 역할은 다음과 같다.

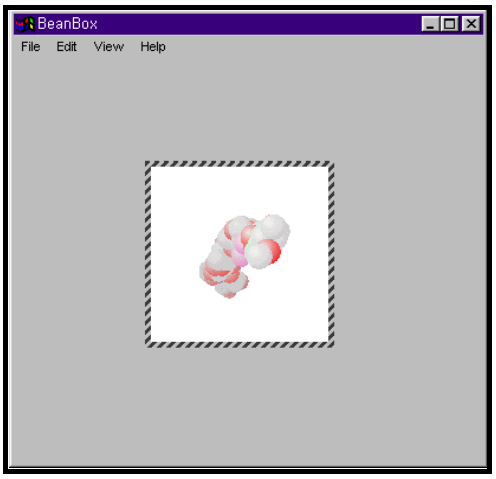
1) 자바빈을 포함하여, 개발자에게 비주얼하게 보여준다.

2) 포함된 자바빈들을 복사/자르기/붙이기 등을 할 수 있다.

3) 변화한 자바빈을 하드디스크에 저장할 수 있다. 자바빈들은 비주얼하게 다룰 수 있기 때문에 속성이나 행위를 변화시킬 수 있다. 예를 들어 버튼역할을 자바빈의 색깔을 변화시킬 수 있다. 이렇게 변화된 자바빈을 그 상태를 유지시키기 위해 하드디스크에 저장할 수 있다.

3) 포함된 자바빈끼리 이벤트를 전달하고 처리할 수 있도록 해준다.

만약 사용자가 툴박스위의 특정 자바빈이름위에 마우스를 누른 다음, 자바빈 컨테이너위에 마우스를 누르면 그 위치에 자바빈이 포함된다. 예를 들어 사용자가 툴박스에서 " molecular" 자바빈(이 자바빈의 컽모양은 분자모형이다)을 마우스로 선택하여, 컨테이너 윈도우에 클릭하면 포함된다. 포함된 자바빈즈의 모양은 다음 <그림>과 같다.

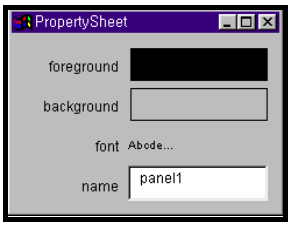


<그림> 분자모양의 자바빈

포함된 자바빈들은 그 위치와 크기를 마우스로 변경할 수 있다.

- 자바빈 속성시트(property sheet)

위 I-10 <그림>의 3번째 윈도우로서, 자바빈의 속성값을 화면에 나타내고 수정할 수 있는 윈도우이다.



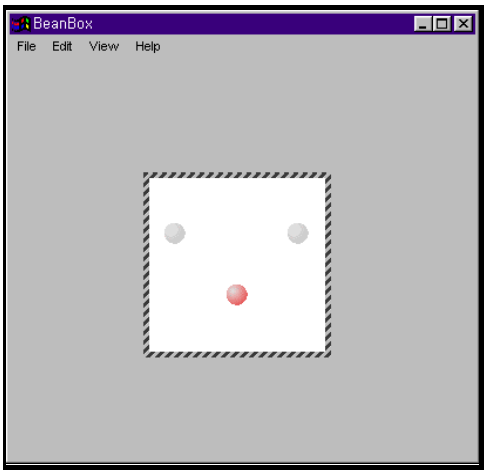
<그림> 속성 시트

자바빈 컨테이너안에서 선택된 자바빈의 속성이름과 해당값을 화면에 보여주고, 사용자가 임의로 속성값을 변경시킬 수 있다. 사용자가 속성값을 변화시키면 그 속성값이 의미하는대로 자바빈즈의 겉모양이나 행위가 변화된다.

예를 들어 " molecular"자바빈은 " moleculeName"이라는 속성이름을 가지고, 처음에 할당되는 값은 " HyaluronicAcid"이다.(I-12의 <그림>참조) 그런데 이 값을 " water"로 변화시키자.(다음 <그림> 참조)

<그림> 변화된 속성

이렇게 속성시트에서 속성값을 변화했을 때, 자바빈의 모양은 다음 <그림> 처럼 변화된다.



<그림> 변화된 자바빈

자바빈의 모양이 분자모형에서 물방울 모양으로 변한 것을 알 수 있다.

✍✍ 테스트용 자바빈들을 이용하여 프로그래밍하는 법

비주얼하게 자바빈들의 속성들을 변화시킴으로 인해서 자바빈의 모습이나 행위를 결정할 수가 있었다. 그런데 자바빈들은 혼자서 독립적으로 동작할 수 있지만, 서로 연결되어서 동작할 수도 있다. 예를 들어 A라는 자바빈위에 마우스이벤트가 발생했을 때, 즉 사용자가 A위에 마우스를 눌렀을 때 또 다른 자바빈 B의 색깔이 바뀌게 할 수 있다는 것이다.

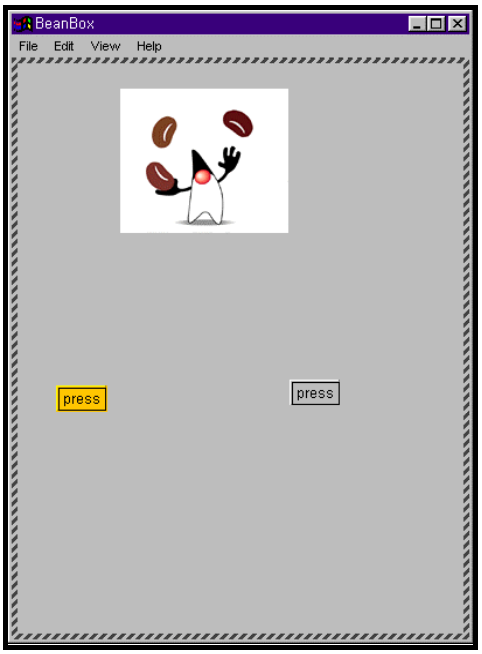
이렇게 테스트 용 자바빈들을 자바빈 컨테이너 안에 포함시키고, 그들을 서로 연결하여 특정 이벤트가 발생할 때 그 이벤트를 처리할 수 있도록 하는 과정을 예제들을 살펴봄으로서 알아본다.

가. 버튼과 애니메이션 자바빈 컴포넌트

두 개의 버튼 자바빈과 한 개의 애니메이션 자바빈을 자바빈 컨테이너안에 포함시킨다. 그리고 그 중 1개의 버튼을 누르면 애니메이션이 실행되고, 나머지 1개의 버튼을 누르면 애니메이션이 중단되는 예제를 만들어보자.

a. 툴박스에서 버튼 기능을 하는 " OrangeButton"과 " OurButton", 애니메이션기능을 수행하는 " Juggler" 자바빈을 컨테이너안에 포함시킨다. " Juggler " 는 콩(bean)을 돌리는 듀크(Duke)를 애니메이션으로 보여주는 자바빈이다.

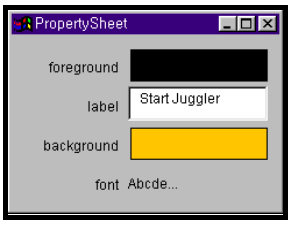
3개의 자바빈을 포함시킨 자바빈 컨테이너의 모습은 다음 <그림>과 같다.



<그림> 두 개의 버튼과 애니메이션 자바빈

b. "OrangeButton"의 이름을 "press"에서 " Start Juggler", "OurButton"의 이름을 "press"에서 "Stop Juggler"로 바꾼다. 이것은 속성 시트의 속성값을 변경하여 이룰 수 있다.

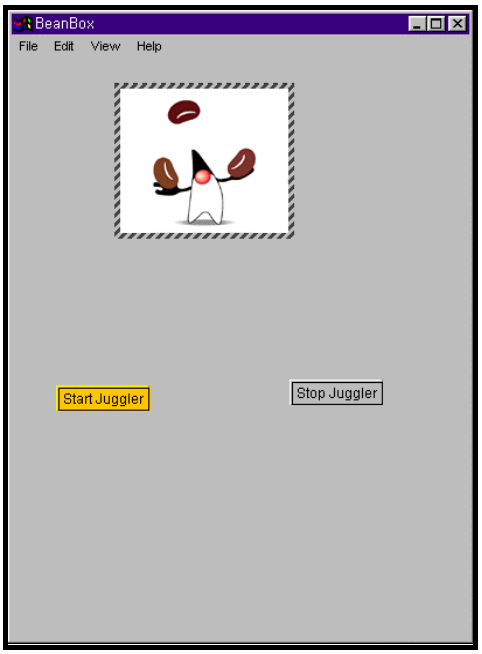
먼저 "OrageButton"을 마우스로 선택한 후, 해당 속성시트안에 " label"이라는 속성의 값을 " Press"에서 "Start Button"으로 변경한다. (다음 <그림>참조)



<그림> 변화된 버튼의 속성

"OurButton"도 마찬가지 방법으로 " Stop Button"으로 "label"속성의 값을 변경시킨다. 그러면 자바빈 컨테이너안의 두 개의 버튼이름이 " Start Juggler", "Stop Juggler"로 바뀌게 된다.

이렇게 이름을 바꾸는 것은 " OrangeButton"을 눌렀을 경우 "Juggler"를 실행시키고 즉 애니메이션을 실행시키고, " OurButton"을 누르면 " Juggler"를 중단시키는 버튼의 역할을 사용자에게 알리기 위함이다. 이 과정을 다 마친 후의 자바빈 컨테이너의 모습은 다음 <그림>과 같다.



<그림> 이름들이 바뀐 두 개의 버튼과 애니메이션 자바빈

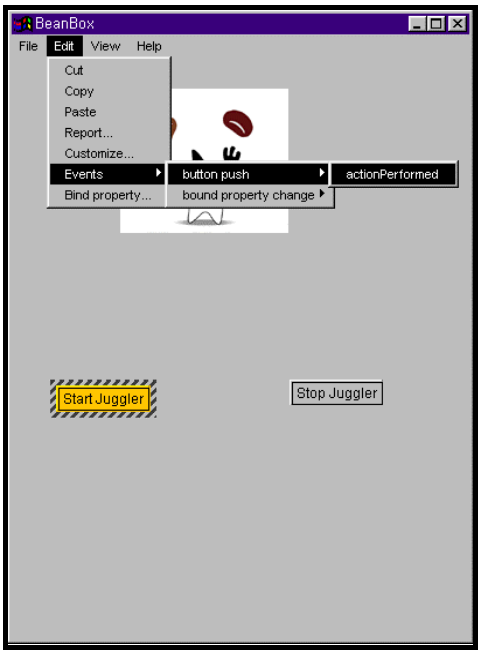
c. “ OrangeButton"버튼을 눌렀을 경우, "Juggler" 애니메이션이 시작되도록 만든다. 자바빈의 실행방식은 GUI컴포넌트들의 그것과 비슷하다. 사용자나 시스템으로부터 발생한 각종 이벤트들을 자바빈 컴포넌트들이 처리하도록 구성되어 있다. 또한 특정 컴포넌트에서 발생한 이벤트가 다른 컴포넌트에게 전달되어, 그 이벤트를 처리하는 서비스(메쏘드)를 실행하도록 연결시켜 주는 것도 GUI컴포넌트의 이벤트 핸들링 방식과 같다.

그래서 ” OrangeButton"버튼위에서 마우스를 눌렀을 때 발생한 이벤트를 "Juggler"에게 전달해 주는 작업이 필요하다. "Juggler"에게 그 이벤트가 전달되었을 때 “ Juggler"는 애니메이션을 시작한다. 만약 ” OurButton"에서 발생한 이벤트를 전달받으면 “ Juggler"는 애니메이션을 중단해야 한다.

이런 작업들은 자바빈 컨테이너의 “ Edit"메뉴의 ” Event"서브메뉴가 담당한다. 이 메뉴는 사용자가 마우스로 선택한 컴포넌트에서 발생가능한 이벤트의 리스트를 사용자에게 보여주는 역할을 한다. 또한 사용자가 이 이벤트 중에서 특정한 것을 마우스로 선택하면, 선택된 컴포넌트에 그 이벤트가 전달되어 직접 처리하거나 아니면 다른 컴포넌트에게 그 이벤트를 전달하여 처리할 수 있도록 해준다.

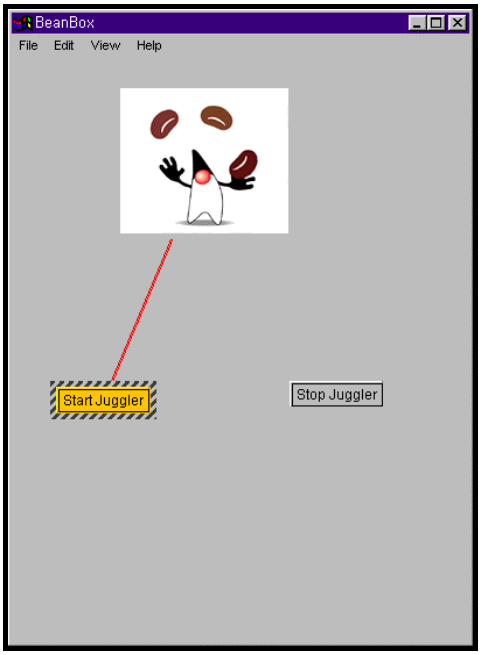
그럼 “ OrangeButton"을 눌렀을 경우 그 이벤트를 "Juggler"에게 전하고, "Juggler"가 애니메이션을 시작하도록 설정해보자.

먼저 자바빈 컨테이너안의 “ OrangeButton"을 마우스 버튼으로 선택한다. 그리고 자바빈 컨테이너의 메뉴에서 ” Edit->Events->buton push->actionPerformed"를 선택한다. (다음 <그림> 참조) 이 이벤트는 사용자가 “ OrangeButton"버튼을 마우스로 눌렀을 때 생성되는 이벤트이다.



<그림> 버튼에서 발생가능한 이벤트중 처리하고자 하는 이벤트를 선택

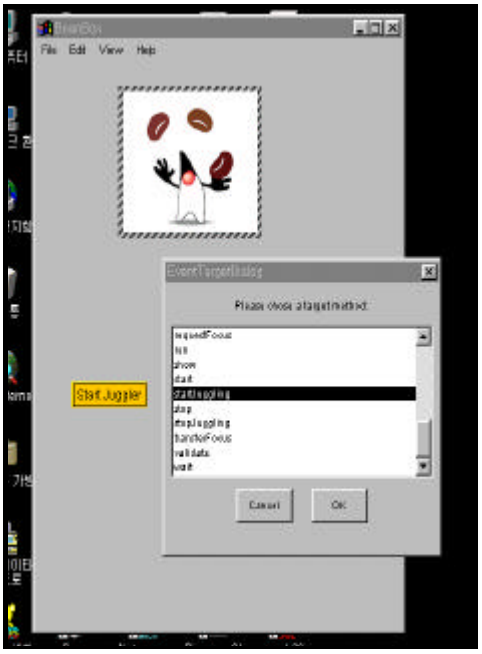
그러면 " OrangeButton"에서 현재 마우스의 위치로 빨간 선이 생성되는 것을 볼 수 있을 것이다. 이 줄은 " OrangeButton"에서 발생한 " button push"이벤트를 전달받아 처리하는 컴포넌트를 가리키는데 사용한다. 본 예에서는 이 이벤트가 " Juggler"에게 전달되어 " Juggler"가 그 이벤트를 받으면 애니메이션을 실행하도록 하기 위해서, 마우스를 " Juggler" 위에 놓는다. (다음 <그림> 참조)



<그림> 버튼에서 발생한 이벤트를 애니메이션 자바빈으로 전달

마우스를 "Juggler"위에 놓는 순간, " EventTargetDialog"라는 새로운 다이얼로그 윈도우가 생성될 것이다. 이 윈도우는 전달받은 이벤트를 처리할 수 있는 메쏘드의 리스트들을 사용자에게 보여준다. 만약 사용자가 특정 메쏘드를 선택하면 이 벤트가 발생했을 때 그 메쏘드가 곧장 실행할 수 있도록 해 준다.

본 예에서는 " OrangeButton"버튼을 마우스로 눌렀을 때 발생하는 이벤트가 " Juggler"에게 전달될 때, " Juggler"가 제공하는 메쏘드들 중에서 " startJuggling" 이라는 메쏘드를 실행하도록 " EventTargetDialog"에서 " startJuggling"이란 메쏘드를 선택한다. (다음 <그림> 참조) 그리고 "Ok"버튼을 누른다. dsstargJuggling"이란 메쏘드는 "Juggler" 컴포넌트안에 정의되어 있는 메쏘드로서 애니메이션을 시작하는 기능을 한다.



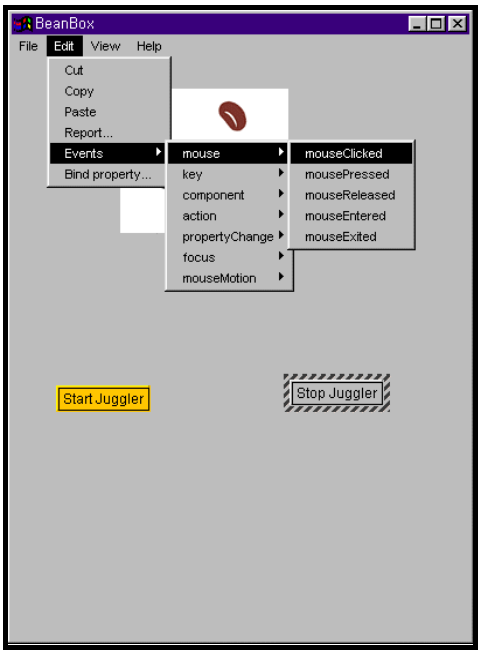
<그림> 이벤트 처리 메쏘드를 선택

이 작업이 끝난 후 "OrangeButton"을 누르면 " Juggler"가 실행된다. 즉 듀크가 콩을 돌리는 애니메이션이 시작된다.

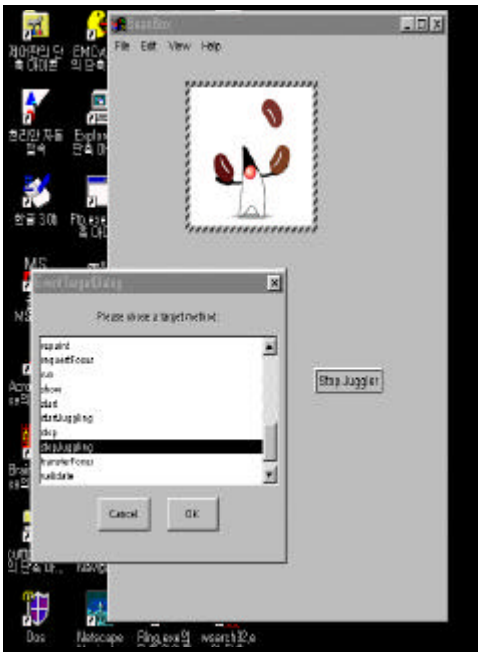
4) " OurButton"버튼을 눌렀을 경우, "Juggler"가 실행을 멈추도록 만든다. 이것은 3)번과 마찬가지 방법으로 하면 된다. 이벤트 처리를 위하여 " " Edit->Events -> mouse->moseClicked"를 선택한다.(다음 <그림> 참조)

그 후 마우스를 " Juggler"위에 놓으면 " EventTargetDialog"가 생성된다.

이 이벤트가 " Juggler"에게 전달되었을 때, 애니메이션이 중단되도록 " Juggler" 컴포넌트안에 정의되어 있는 " stopJuggling"메쏘드를 실행해야 한다. 따라서 " EventTargetDialog"의 " stopJuggling"을 선택한다.(다음 <그림> 참조)



<그림> 또 다른 버튼에서 발생가능한 이벤트 중 처리하고자 하는 이벤트를 선택



<그림> 이벤트 처리 메쏘드 선택

6) 위의 모든 작업이 끝난 후에, “ OrangeButton"을 누르면 "Juggler"가 실행되고 ” OurButton"을 누르면 "Juggler"가 중단되는 것을 테스트할 수 있다. 만약 이렇게 연결된 본 예제를 하드디스크에 저장하려면 “ File"메뉴의 ” Save"를 선택하면 된다.

✍ EJB와 JavaBeans

	JavaBeans	EJB
만든 곳	Sun	Sun
목적	비주얼 개발환경에서 사용되는 재사용가능한 컴포넌트 모델	미션 크리티컬한 서버쪽 비즈니스 어플리케이션에서 사용될 수 있는 분산객체 컴포넌트 모델
주로 동작하는 컴퓨터 위치	클라이언트 > 서버	오직 서버
개발 난이도	상대적으로 쉬움	상대적으로 어려움
트랜잭션 및 패일오버, 동시 동작성 보장	불가능	가능
패키지	java.bean.*	javax.ejb.*
동작 프로세스	Intra-process	Inter-process

분산객체 아키텍처

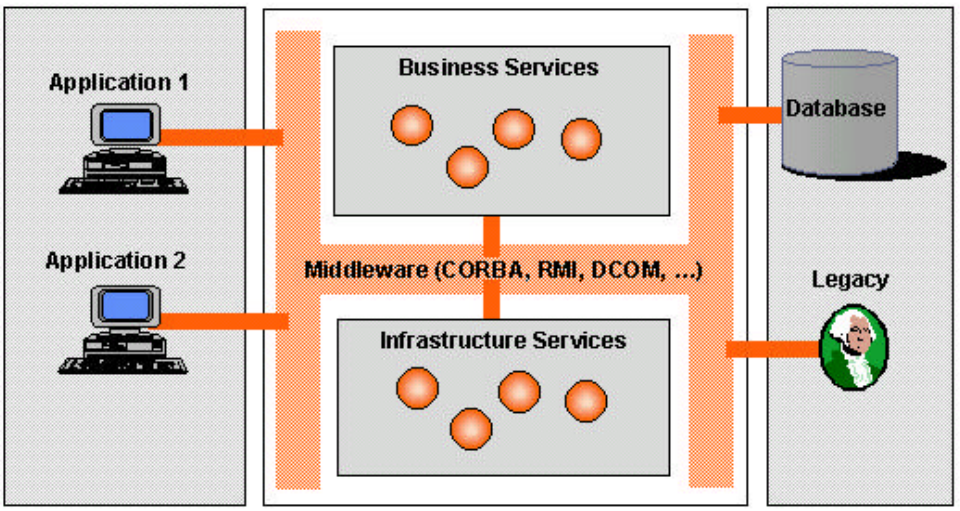
개요

✍ RPC(Remote Procedure Call)을 객체 레벨로 적용함.

✍ 같은 컴퓨터 또는 네트웍으로 연결된 다른 컴퓨터안에 있는 객체의 메쏘드를 호출하여 그 결과값을 얻을 수 있음.

✍ CORBA, RMI, DCOM 등의 프로토콜이 존재

✍ 분산객체 아키텍처 개요



<그림> 분산객체 아키텍처

CORBA(Common Object Request Broker Architecture) : OMG

✍✍ 특징

a. 분산객체들의 메시지 패싱을 위한 표준안

b. 다른 언어로 작성된 객체 지원

c. 다른 플랫폼 지원

d. 분산 트랜잭션 지원

✍✍ CORBA는 표준 ORB(Object Request Broker)를 제시

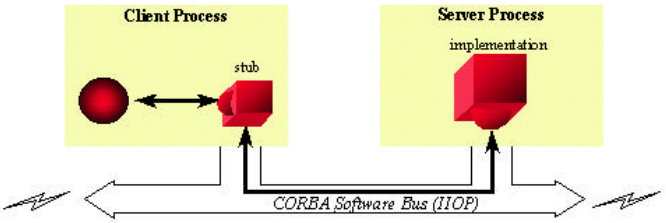
a. 표준 언어 바인딩 : IDL(Interface Definition Language)

b. 표준 통신 프로토콜 : IIOP(Internet Inter-ORB Protocol)

✍✍ OMG그룹에 의해 정의됨 : 1991년 CORBA1.0 이후로 계속 발전되어 옴.

✍✍ 동작과정

Stub, Skeleton을 사용하여 리모트 객체를 로컬 객체처럼 사용할 수 있게 함.



<그림> 코바 ORB

RMI(Remote Method Invocation) : Java

✍✍ 특징

a. CORBA와 유사함.

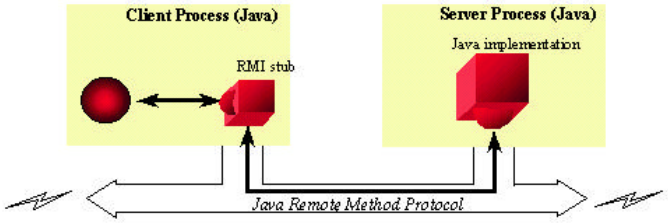
b. 자바언어만 지원

c. CORBA보다 훨씬 사용이 간단함

d. JDK 1.1이상의 자바가상머신에 ORB가 내장되어 있음.

✍✍ 동작과정

Stub, Skeleton을 사용하여 리모트객체를 로컬객체처럼 사용할 수 있게 한다.

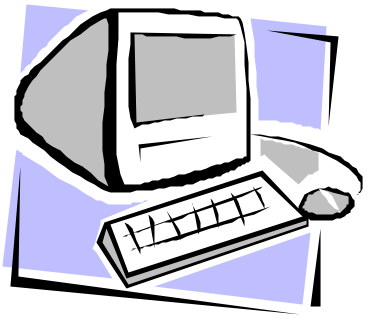


<그림> RMI 프로토콜

✍✍ 미래

a. CORBA 클라이언트가 RMI 리모트객체를 호출할 수 있다.

b. RMI 클라이언트가 CORBA서버 객체를 호출할 수 있다.

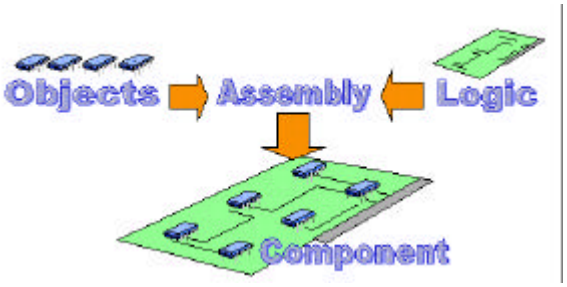


Chapter 2

EJB 컴포넌트 모델

컴포넌트란?

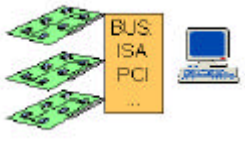
컴포넌트들은 조립된다.



<그림> 컴포넌트와 조립

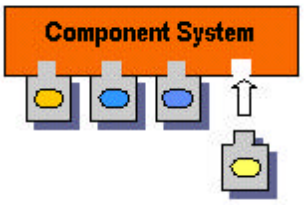
하드웨어 컴포넌트 시스템

- 컴퓨터 부품을 조립할 때 부품을 꽂는 마더보드나 버스 등이 소프트웨어 컴포넌트 시스템과 대응된다.
- 소프트웨어 컴포넌트는 그런 컴포넌트 시스템에서 제공하는 인터페이스를 준수해야 조립될 수 있다.



<그림> 컴포넌트와 버스

컴포넌트 시스템



- 제공해야 하는 표준
  - 컴포넌트가 무엇을 수행할 수 있는지 발견하는 것
  - 컴포넌트끼리 서로 통신하는 방법
  - 컴포넌트들을 생성하고 파괴, 찾는 메커니즘
  - 기타 기반 서비스들
- 컴포넌트시스템은 컴포넌트들을 조립하고 관리하기 위해 가장 기본이 되는것이다.

컴포넌트들의 특징

- 컴포넌트들은 재사용가능해야 한다.
  - 컴포넌트들은 다른 컴포넌트들과 연관성이 적다.
- 컴포넌트들을 생성할 때가 아닌 조립할 때 연관성이 결정된다.
  - 따라서 컴포넌트의 재사용성이 증가한다.
- 컴포넌트는 잘 정의된 인터페이스를 가지고 있다.
  - 컴포넌트 사용자는 내부구현을 모른다.
- 컴포넌트의 크기는 매우 클 수도 있으며 작을 수도 있다.
  - 컴포넌트들은 다른 컴포넌트들로부터 만들어질 수 있다.

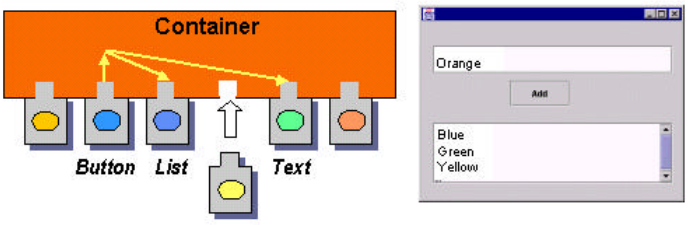
객체와 컴포넌트의 관계

- 컴포넌트는 반드시 객체 일 필요는 없다.
  - 컴포넌트는 Functional(non-OO)기술이나 언어로 개발될 수 있다.
  - 한 컴포넌트는 수 많은 객체들로 이루어질 수 있다.
  - 그렇지만 객체개념은 컴포넌트개념과 유사한 부분이 많다. (속성, 메쏘드, 클래스, 캡슐화, 추상화)
- 객체는 코드레벨의 재사용/컴포넌트는 아키텍처 레벨의 재사용이 가능하다.

클라이언트 컴포넌트 대 서버 컴포넌트

클라이언트 컴포넌트

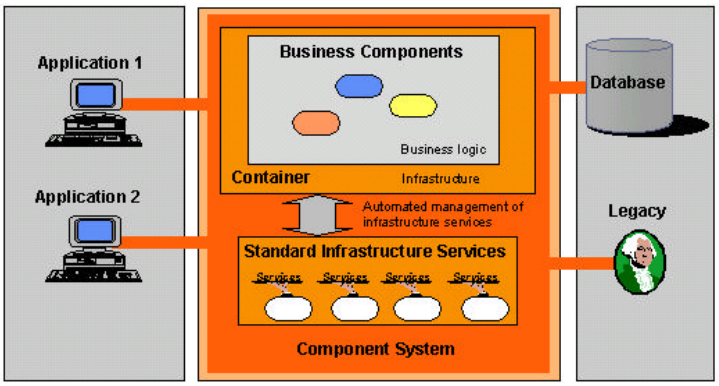
- 컴포넌트 = 버튼, 리스트, 스프레드 쉬트 등
- 컨테이너 = 윈도우, 폼 등
- 컴포넌트시스템은 GUI에 맞게 이벤트와 속성들에 기반한다.
- 툴들은 코딩없이 컴포넌트들을 조립하는데 사용된다.
- 예제 : JavaBeans, ActiveX(OCX)
- 컴포넌트 조립



<그림> GUI 컴포넌트와 컨테이너

서버 컴포넌트

- 통합 서버 컴포넌트 시스템



<그림> 서버 컴포넌트 시스템

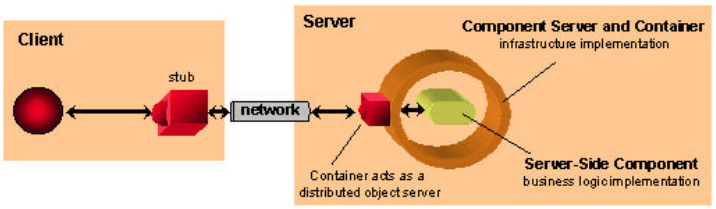
- 컴포넌트와 컨테이너

  컴포넌트

  - 인터페이스와 구현이 분리되어 있다.
  - 컨테이너안에서 구현물이 초기화 된다.

  컨테이너

  - 클라이언트와 컴포넌트들 사이의 메시지패싱을 관리하며 트랜잭션, 보안 등의 기반 서비스를 자동으로 수행한다.
  - 컴포넌트들의 callback함수를 통해 직접 통신한다.



<그림> 컨테이너와 컴포넌트

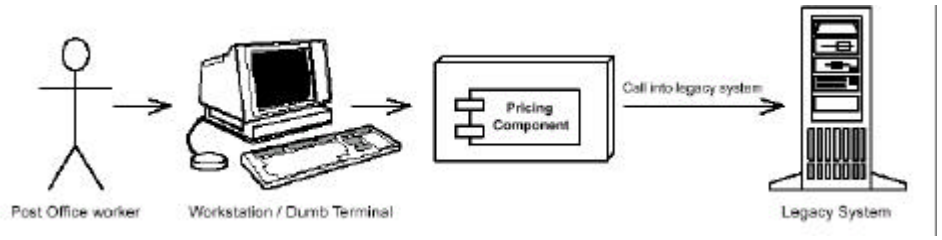
✍ 컴포넌트 서버의 기본 인프라 서비스들

- 디렉토리 서비스
- 분산트랜잭션 서비스
- 보안 관리
- 동시 접속 및 실행 관리 (멀티쓰레드 관리)
- 영속성 관리(persistence management)
- 자원 풀링(Resource Pooling)
- 관리자 인터페이스
- 로드 밸런싱
- fault tolerance

컴포넌트 재사용 사례

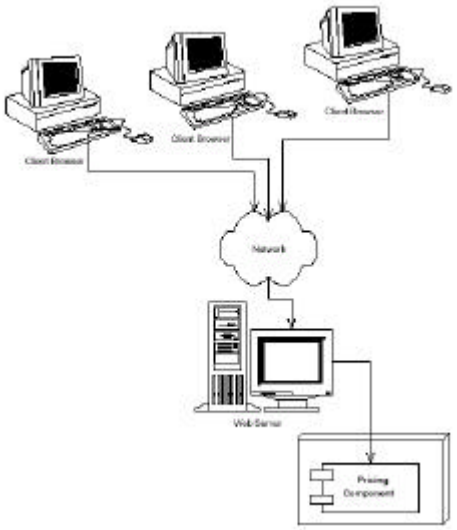
✍ 다음은 가격산정 컴포넌트를 여러 어플리케이션 환경에서 재사용하는 사례이다.

a. 우체국



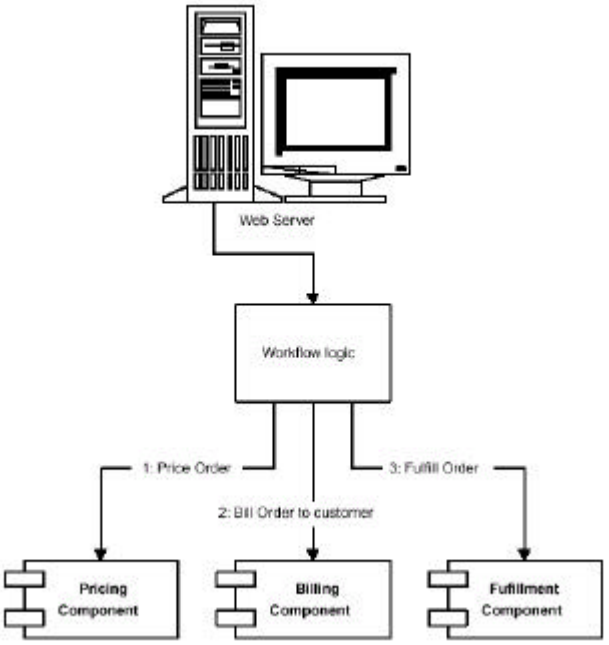
<그림> 우체국 시스템

b. 인터넷 자동차 판매 시스템



<그림> 자동차 판매 웹 전자상거래 시스템

c. 전자상거래 워크플로우



<그림> 일반 전자상거래 워크플로우 시스템

컴포넌트 트랜잭션 모니터

TP 모니터 (Transaction Processing Monitor)

- 많은 클라이언트들이 서버에 동시에 접속하여 서비스를 수행해도 시스템 무리가 없으며 또한 트랜잭션을 보장하도록 도와주는 시스템
- 은행업무, 전자상거래 등 미션 크리티컬한 업무에 주로 적용
- 메인프레임 급 서버 시스템에 주로 채택되어 옴.

ORB(Object Request Broker)

- 분산객체 메시지 패싱을 위한 미들웨어
- 다양한 플랫폼, 복잡한 네트웍 등의 등장으로 인해 필요성 증대
- 그렇지만 현재로는 통신백본을 제공할 뿐 동시성, 트랜잭션, 자원관리, fault tolerance 등의 구현은 상당부분 개발자에게 부담.

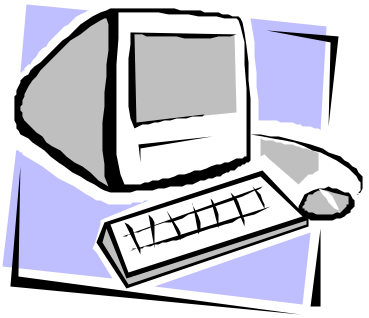
CTMs : ORB와 TP 모니터의 결합

- 분산객체 미들웨어 + TP 모니터 + 서버 컴포넌트 모델 = CTM
- OMG

  OTS(Object Transaction Service) + CORBA
- Microsoft

  MTS(Microsoft Transaction Server) + DCOM
- Sun

  JTS(Java Transaction Service) + EJB



Chapter 3

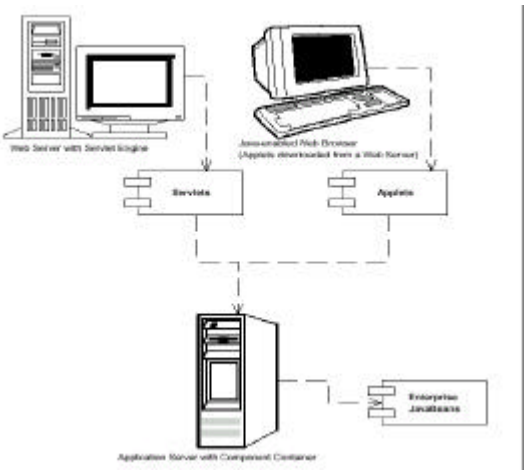
EJB로 무엇을 할 수 있는가?

EJB 어플리케이션 서버가 제공하는 기능

일반적으로 EJB어플리케이션 서버들은 J2EE(Java 2 Enterprise Edition : 엔터프라이즈 환경에 필요한 프로그래밍 API집합)를 지원하며 구체적으로 다음과 같은 기능을 공통적으로 제공한다.

- EJB 컴포넌트 배치 및 실행
- JDBC, RMI, JNDI 등 EJB 컴포넌트 사용을 위한 기본적인 API 지원
- 보안
- 동시동작, 패일오버(fail over), 트랜잭션 및 무결성 보장
- 멀티쓰레드 안정성 보장
- HTTP 지원
- 서블릿, JSP 등 웹개발 환경 지원

EJB 컴포넌트 사용 예



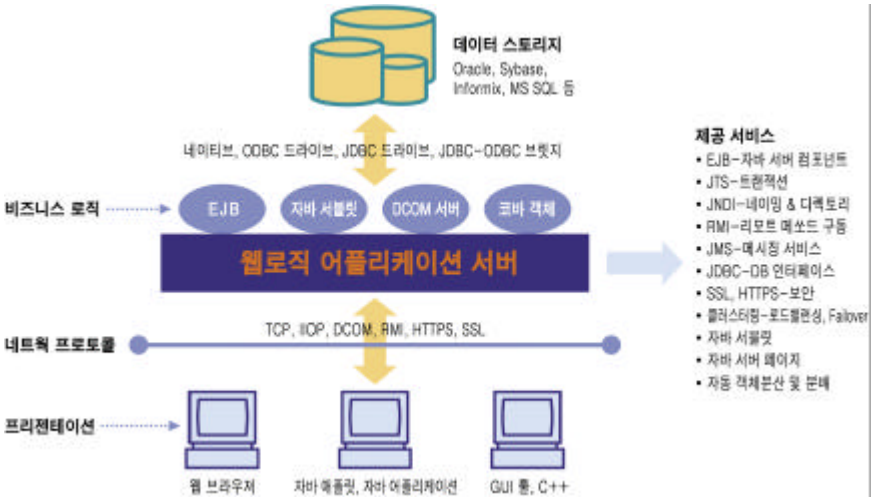
<그림> 엔터프라이즈 빈의 다양한 사용 예

어플리케이션 서버의 종류

웹로직(Weblogic) : BEA사

✍ 개요

웹로직 애플리케이션 서버를 한마디로 정의하면 ‘ 분산된 자바 애플리케이션들을 취합, 분산 배치하고 관리할 수 있는 표준 기반의 확장성 있는 개방형 플랫폼 ’ 이라 할 수 있다. 웹로직 애플리케이션 서버는 자바 비즈니스 컴포넌트들 뿐만 아니라 이종의 데이터베이스 그리고 다른 네트웍 자원들을 상호 연결해주는 역할을 한다. 웹로직은 그것이 관할하는 애플리케이션 객체들 간의 트랜잭션 무결성, 보안성 그리고 확장성을 제공한다. 나아가 데이터베이스 드라이버 같은 것을 클라이언트 데스크탑들마다 설치할 필요없이 서버측에 한 번만 설치하면 되므로 소프트웨어 비용과 유지관리에 드는 노력을 줄여준다. 웹로직 애플리케이션들은 데이터베이스 연결, 이벤트 처리, 트랜잭션 컨트롤(JTS), 원격 메쏘드 호출(RMI), 네이밍 및 디렉토리 서비스(JNDI), 보안, 글로벌 접속, 기기사용 및 자원관리와 같은 서비스에의 접속을 공유할 수 있다.

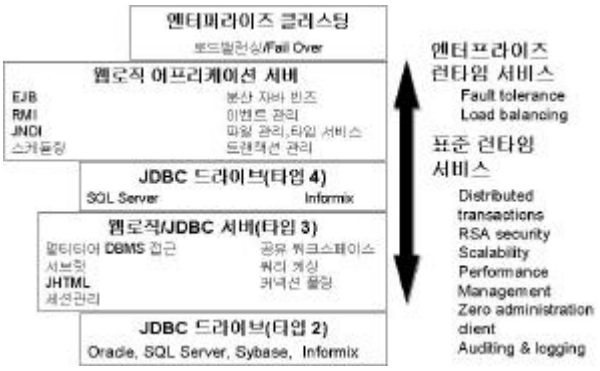


<그림> 웹로직 애플리케이션 서버 개요

컴포넌트들을 레고 형식으로 바로 가져다 쓸 수 있도록 하는 서버와 통신의 완벽한 하부구조를 제공하기 때문에 고객이 직접 개발할 때와 비교해 볼 때 웹로직을 쓰면 자바 구축 및 배치에 드는 노력이 75% 까지로 줄 수 있다. 더구나 이 하부구조는 표준에 기반하여 신뢰성 및 확장성을 제공하며 지원 또한 충분히 받을 수 있는 장점이 있고 또한 컴파일을 다시 하지 않고도 어느 플랫폼에나 이식이 가능하다.

✍ 제품 모듈 구성

웹로직의 제품라인은 JDBC 드라이버에서 출발하여 웹로직 애플리케이션 서버에서 완결됩니다.J DBC 같은 것들은 전체 애플리케이션 서버를 구성하는 서브셋들이다. 웹로직은 내장 가능한 하부구조 형태로 패키징되며 자바클래스 모음 형태로 제공된다. 웹로직 애플리케이션 서버는 현재 5.1 버전이다.



<그림> 웹로직이 지원하는 JDBC 드라이버

✍✍ 제품 기능

웹로직이 제공하는 기능을 크게 개발과 분산 배치의 두가지 범주로 나누어 보면 다음과 같다.

- 개발을 위해 제공하는 서비스

제공서비스	내 역
EJB	서버측의 비즈니스 로직을 안전하고 트랜잭션 가능한 컴포넌트로 캡슐화
JDBC	자바 프로그램이 다수의 데이터베이스에 접근하고 갱신할 수 있도록 해 줌
RMI	애플리케이션이 특정 자바 플랫폼에 있는 객체의 메쏘드 호출을 할 수 있도록 함
JMS	특정 조건이 변할 때 관련된 모든 애플리케이션들에게 변경된 정보를 전달할 수 있는 서버 푸시 이벤트 모델을 제공. Publish/Subscribe 기능을 제공하며 특히 서버측의 필터링을 통해 네트워과 피전달자의 부하를 경감
JNDI	글로벌 디렉토리 기술과의 통합
파일/프린트	파일에 대한 리모트 접근을 가능하게 하며 자바 애플리케이션을 위한 프린트 서비스도 제공
서블릿	HTTP를 통한 자바 비즈니스 로직의 디스패칭과 동적으로 구성된 HTML 응답을 가능케 함

- 분산 배치를 위해 제공하는 서비스

제공서비스	내 역
보안	X.509를 통해서 인증을, SSL를 통해 통신 무결성과 개인 정보 보호를, ACL을 통해 권한 제한이 가능
트랜잭션 무결성	DBMS 트랜잭션 뿐만 아니라 여러 개의 EJB를 하나의 트랜잭션으로 묶어 주는 기능을 제공
다중프로토콜 지원	방화벽의 터널링 뿐만 아니라, TCP, HTTP, HTTPS, SSL, CORBA IIOP 등과 같은 통신 프로토콜 위에서 통신이 가능
서버 클러스터링	서버측 실행을 위한 프락시 기능 뿐만 아니라 서버들 사이의 리퀘스트 포워딩 기능을 제공
성능 및 확장성	모든 클라이언트 요구가 하나의 커넥션으로 멀티플렉스 되며 ODBC보다는 네이티브 드라이브가 제공. DBMS 커넥션 풀링과 질의 캐싱 기능이 있으며 지속적인 클라이언트/서버 연결과 애플리케이션 컴포넌트간 동적 리로케이션을 통해 부하조절 기능을 제공
그래픽 형태의 콘솔	애플리케이션과 시스템 컴포넌트의 상태를 종합적으로 감지하고 관리 할 수 있는 기능 제공
통합 로깅	진단 정보 및 시스템 상태를 로깅

웹스피어(websphere) : IBM

✍✍ 개요

IBM WebSphere Application Server는 기존의 리소스를 사용하고 개발에 필요한 주기를 단축시켜주며 관리 부담을 덜어주는 재사용이 가능한 공개 기술로 구성된다. Standard Edition을 사용하면 사용자는 Java servlets, JavaServer Pages 및 XML을 사용하여 정적인 웹 사이트를 동적인 웹 컨텐트의 소스로 빠르게 변환할 수 있다. Advanced Edition는 비즈니스 로직을 통합하는 EJB 구성 요소를 실행하는 실행이 뛰어난 EJB 서버이다. Enterprise Edition은 EJB와 CORBA 구성 요소를 통합하여 높은 수준의 트랜잭션과 용량이 큰 e-business 응용 프로그램을 구성한다.

✍✍ 기능

ㅇ JavaServer™ Pages 지원에는 다음 사항이 포함됩니다.

- .91 과 1.0 사양 지원
- 쿼리와 연결 관리를 위한 태크 지원 확대
- JSPs를 위한 XML-compliant DTD

ㅇ 자동 사용자 세션 및 사용자 상태 관리를 포함하는 Java™ servlet

ㅇ 데이터베이스 지원

- JDBC for DB2R Universal Database™, Oracle 및 MicrosoftR SQL Server 를 사용하는 고속 풀 데이터베이스 액세스

ㅇ 파서와 데이터 변경 도구를 포함하는 XML 서버 도구

ㅇ 웹 사이트의 성능과 효율성을 향상하는데 도움을 주는 트래픽 관리 개발을 위한 웹 사이트 분석 도구

ㅇ 웹 페이지 내용을 동적인 언어로 번역하는 기계 번역

ㅇ 다음을 포함하는 IBM HTTP 서버

- 새로운 관리 GUI
- LDAP와 SNMP 연결 지원
- Tivol Ready™ 모듈
- IBM VisualAgeR for Java와 통합하여 개발자들이 웹 기반 응용 프로그램을 원격적으로 테스트하고 디버그할 수 있게 해주어서 개발 시간을 줄여줍니다

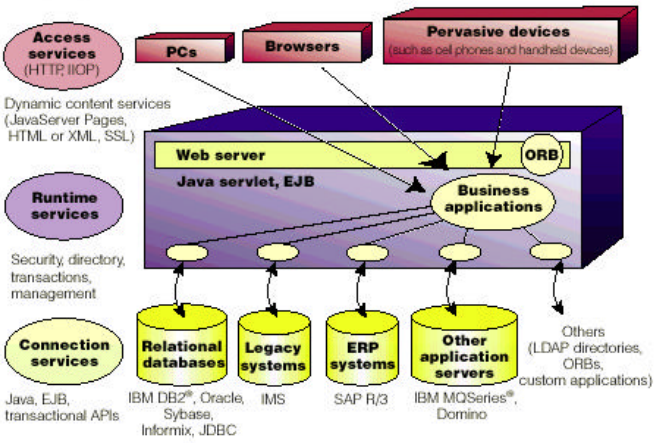
ㅇ Enterprise JavaBeans™ (EJB) 1.0 사양 지원

ㅇ EJBs, Java servlets 및 JSPs의 성능과 스케일을 향상시키는 전개 지원에는 다음이 포함됩니다.

- 애플릿 수준의 분할
- 로드 밸런싱
- 분산 트랜잭션과 트랜잭션 처리 지원 강화
- 관리 및 보안 조절 향상에는 다음이 포함됩니다.
- 사용자 및 그룹 수준 설정
- 메소드 수준 정책 및 조절

ㅇ CORBA 지원으로 bean-managed지속과 container-managed지속을 제공

✍✍ 구조



<그림> 웹스피어의 내부구조

실버 스트림 (silverstream) : SilverStream

✍✍ 구조

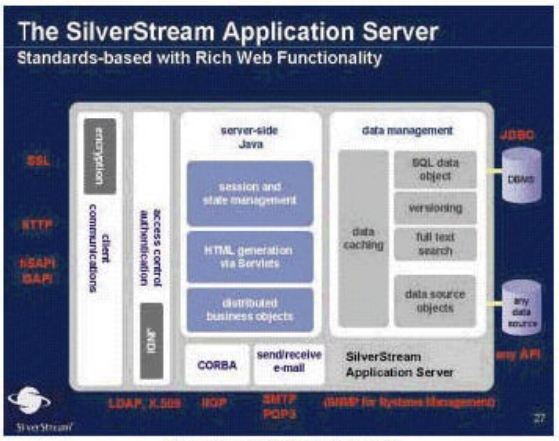


Figure 1: SilverStream's server architecture

<그림> 실버스트림의 내부구조도

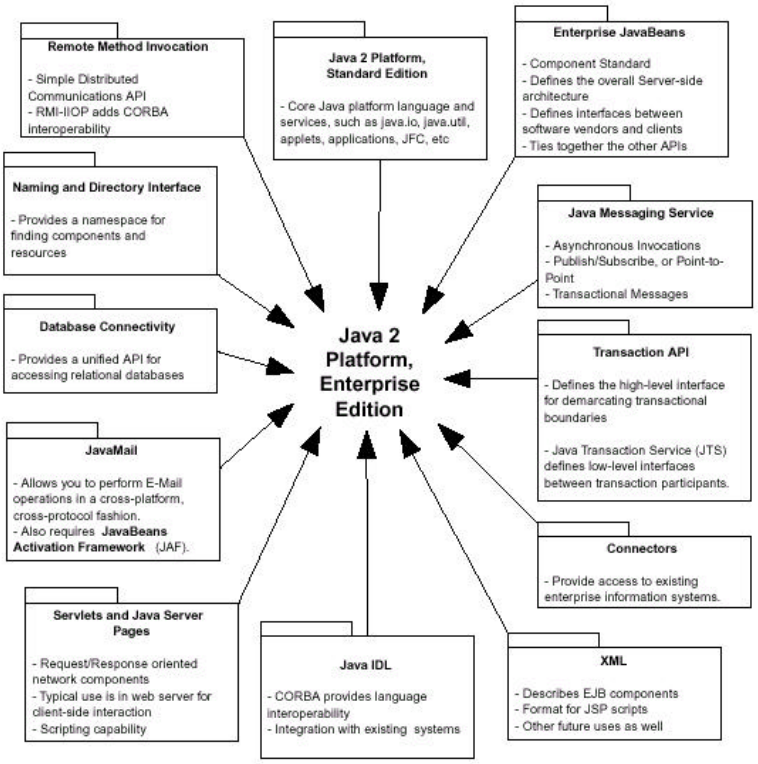
젬스톤

✍✍ http://www.gemstone.com/products/j/index.html

EJB 개발 방법

J2EE(The Java 2 Platform, Enterprise Edition)

✍✍ 기술구조도



<그림> J2EE API 및 지원 기술

✍✍ Sun에서 제안한, 엔터프라이즈 환경에서 필요한 프로그래밍 API 집합

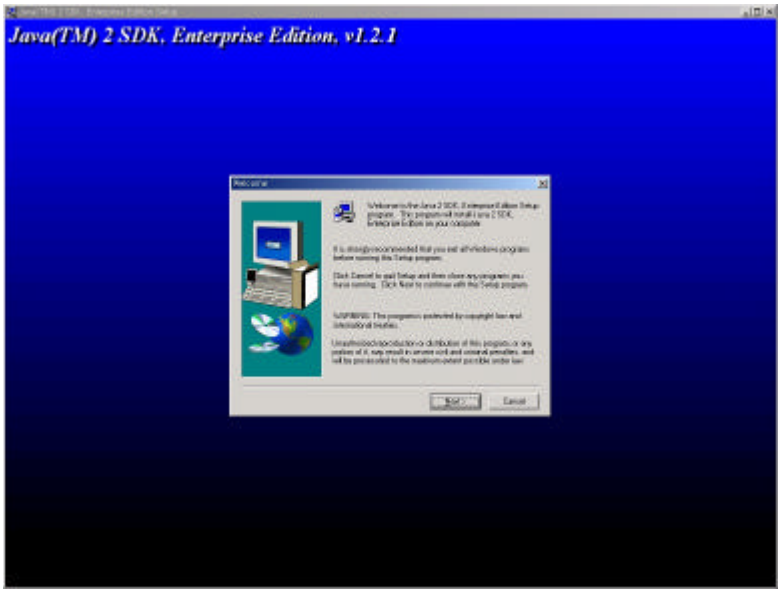
- The Java 2 Platform, Micro Edition
- The Java 2 Platform, Standard Edition (J2SE)
- The Java 2 Platform, Enterprise Edition (J2EE)

✍✍ EJB는 J2EE안에 포함된 스펙이며 현재 웹어플리케이션 서버들은 J2EE를 통합적으로 지원하는 방향으로 바뀌고 있다. 따라서 먼저 J2EE를 습득하고 J2EE SDK로 EJB 컴포넌트 개발을 테스트하는 것이 낫다.

✍✍ J2EE 용 SDK

- 다운로드 : http://www.javasoft.com/j2ee/index.html

- 설치



<그림> J2EE SDK 설치 화면

EJB 지원 웹어플리케이션 서버

상업용 EJB 지원 웹 어플리케이션 서버들 중에 자신의 어플리케이션 운영환경에 최적인 것을 선택한다.

각 EJB를 지원하는 어플리케이션 서버마다 EJB를 Deploy하는 방법이 약간씩 다르지만 EJB가 지원하는 서비스는 어떤 서비스도 동일하게 사용할 수 있다.

- 주의할 것은 웹 어플리케이션 서버가 지원하는 EJB 버전이다. EJB1.0을 지원하는 웹 어플리케이션 서버의 EJB들은 EJB1.1을 지원하는 웹 어플리케이션 서버에서는 그대로 동작하지 않을 수 있다.

- 현재 1.1이 최신버전이며 2.0이 Draft로 제시되어 있는 상황이다.

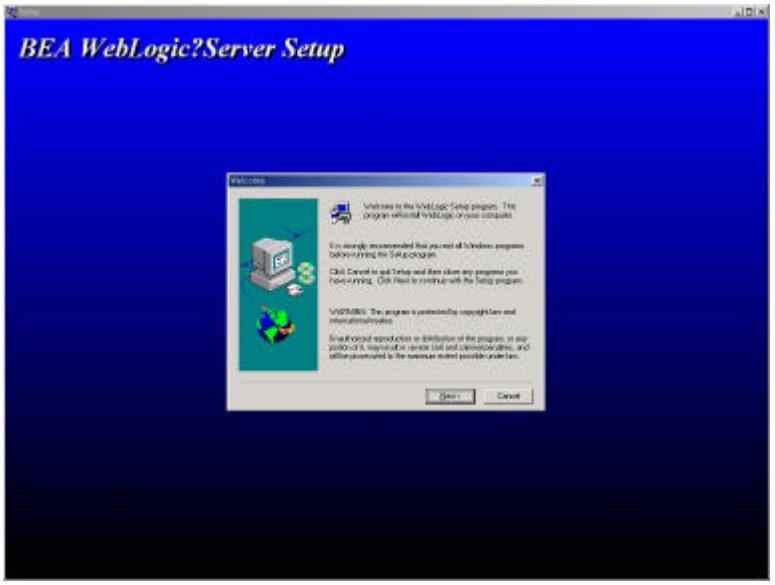
- 비상업용인 오픈소스 EJB서버도 테스트용으로 참고할만하다. (http://www.ejboss.org)

웹로직 어플리케이션 서버 설치 과정

✍✍ http://www.bea.com에서 트라이얼 버전 다운로드 가능

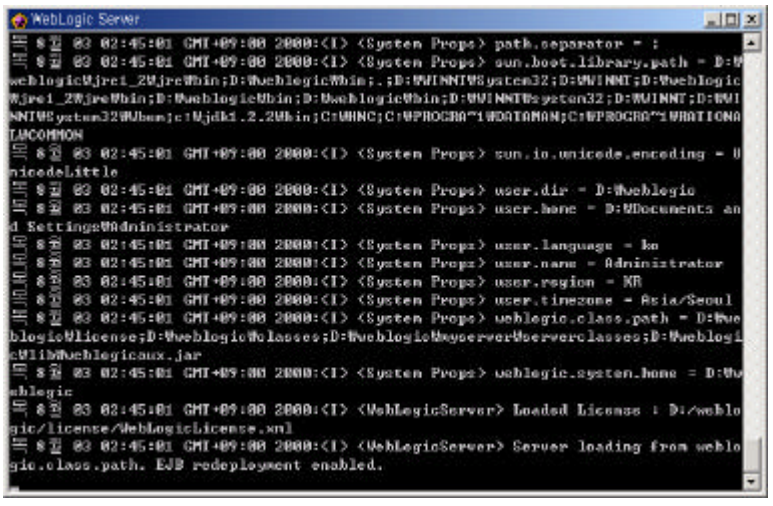
설치과정

✍✍ 윈도우 2000에서 웹로직 5.1버전을 설치한다.



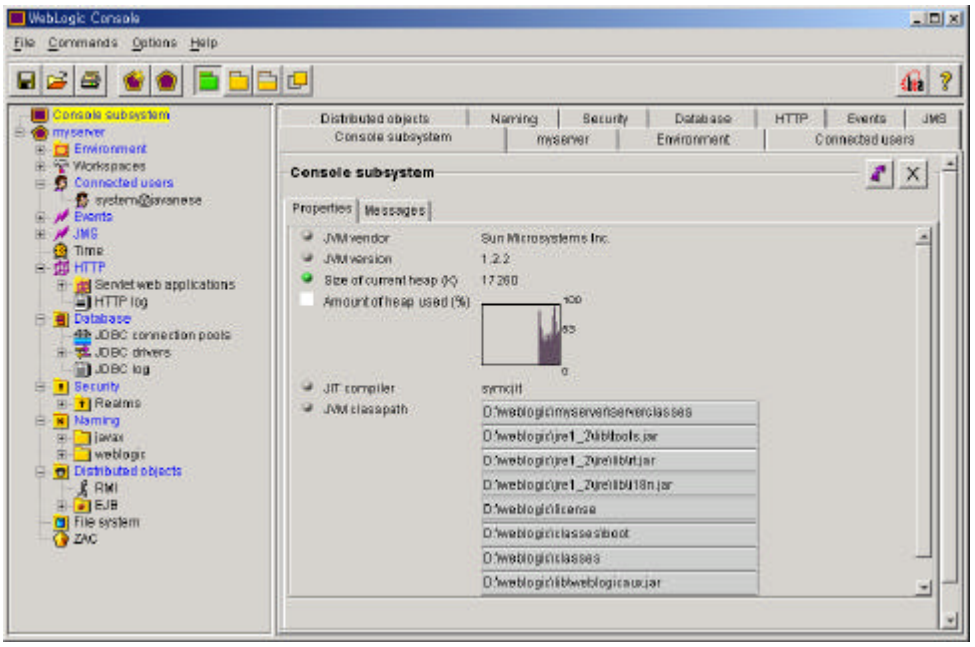
<그림> 웹로직 설치 과정

✍✍ 설치후 서버실행



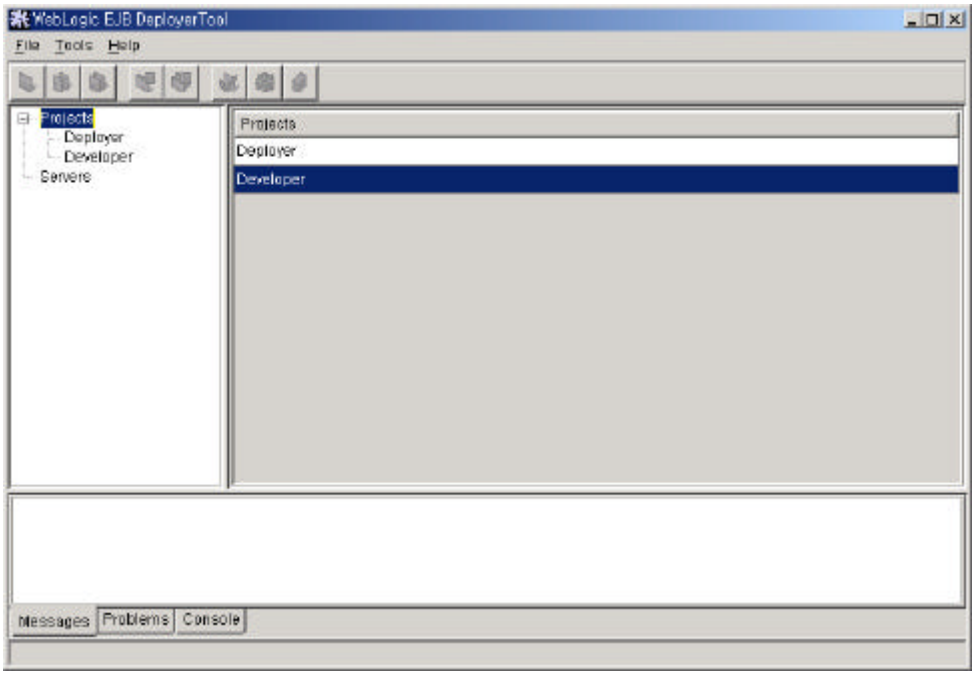
<그림> 웹로직 서버 실행

클라이언트 콘솔



<그림> 웹로직 관리 콘솔

EJB 디플로이먼트(Deployment) 툴



<그림> EJB 디플로이먼트 툴