□특집□

Enterprise JavaBeans(EJB) 기반의 컴포넌트 프로그래밍

김 수 동†

◈ 목 차 ◈

1. EJB 개요
2. 세션 빈과 엔티티 빈
3. EJB 컴포넌트의 지속성
4. 트랜잭션
5 Deployment Descriptor
6 결 론

1. EJB 개요

소프트웨어 산업계에서 컴포넌트 기반 프로그래밍에 대한 관심이 높아지고 있다. 컴포넌트 기반 프로그램이 활성화되기 위해서는 컴포넌트가 운용되는 유연하고 확장성 높은 기반 아키텍처가 필요하다. 선(Sun)사에 의해 제안된 엔터프라이즈 자바빈(Enterprise JavaBeans,EJB) 아키텍처는 컴포넌트 기반의 분산 업무 어플리케이션의 개발과 배치를 위한 컴포넌트 아키텍처이다. EJB를 사용함으로써, 복잡한 분산 객체 프레임워크에 대한 작성 없이 확장성과 신뢰성이 높고 안전한 어플리케이션의 작성이 가능하게 된다. EJB는 어떤 회사의 엔터프라이즈 미들웨어에서도 이동성 있고 재사용이 가능한 어플리케이션을 지원하도록 설계되었다.

EJB의 목적에는 여러 가지가 있다. 시스템 수준의 세부적인 기능을 숨김으로써, 어플리케이션의 작성을 용이하게 하며, 한번 작성된 EJB 어플리케이션들은 다른 환경에서도 바로 실행되게 한다. 따라서 구현된 엔터프라이즈 EJB 컴포넌트는 복수의 플랫폼에서 설치되어 사용될 수 있다. 또한 개발과 전개, 그리고 전사적인 어플리케이션의 생성 주기에 관한 실행 특성들을 제시한다. 이를 위해, EJB는 여러 회사에서 구현된 EJB 제품이 연동되도록 하는 계약(Contract)을 제공한다. 한편 EJB는 자바 API와의 호환을 지원할 뿐만 아니라, 엔터프라이즈 자바빈과 자바 외의 언어로 구현된 어플리케이션과의 상호연동 또한 지원한다. 또한 CORBA 프로토콜과의 호환도 지원한다.

EJB의 컴포넌트 아키텍처는 컴포넌트의 표준적인 구축에서부터 관리 및 유지보수까지의 전반적인 컴포넌트 개발을 용이하게 지원하기 위하여 다음의 3가지 요소들로 구성 된다. 먼저 컴포넌트를 개발하기 위한 툴과 배치된 컴포넌트를 관리하기 위한 컨테이너, 그리고 컴포넌트를 배치시키고 유지 관리하는 툴로서, 이들 각 요소들은 컴포넌트 업계에서 필수적인 요구되는 부분이다.

EJB는 분산 트랜잭션 지향 엔터프라이즈 어플리케이션의 컴포넌트로서 한 클래스가 하나의 빈이 된다. 모든 엔터프라이즈 자바빈은 빈의 생성과 관련된 기능을 가지고 있는 홈(Home)과 비즈니스 로직을 가지고 있는 원격(Remote) 인터페이스들을 가지며, 클래스가 지원하는 기능과 상태, 자료의 특징에 따라 세션 빈과 엔티티 빈으로 나뉘게 된다.

† 정회원 : 숭실대학교 정보과학대학 컴퓨터학부 교수

EJB는 클라이언트와 컨테이너 사이, 그리고 엔터프라이즈 빈과 컨테이너 사이에 계약을 가지고 그 계약 내에서 각 역할들을 지원된다. 클라이언트는 엔터프라이즈 자바빈의 홈 및 원격 인터페이스를 접근하고, 객체 식별자를 얻는다. 또한 엔터프라이즈 빈의 동적인 호출을 위한 메타자료 인터페이스가 지원된다. 컴포넌트 계약 차원에서는 엔터프라이즈 자바빈의 인스턴스에 관한 생명주기 등이 컨테이너에 의해 관리되고, 모든 컨테이너의 서비스 목록들이 엔터프라이즈 빈들에게 제공된다.

2. 세션 빈과 엔티티 빈

EJB에서는 빈을 기본적인 컴포넌트 단위로 정의하고 있다. 빈에는 두 종류가 있는데, 세션 빈(Session Bean)과 엔티티 빈(Entity Bean)이 그것이다. 클래스에 포함된 속성들이 영구적이냐 일시적이냐에 따라서 클래스들의 빈 맵핑이 달라진다. 클래스들이 가지고 있는 자료의 지속성(Persistence)을 고려하여 일시적인 경우는 세션 빈으로 구현하고, 영구적인 자료를 관리하는 경우는 엔티티 빈으로 구현한다.

2.1 세션 빈

세션 빈은 자료의 지속성이 일시적인 성격을 지니는데, 이는 엔티티 빈으로 처리 되지 않는 부분들을 채워주는 역할을 하며, 서로 다른 빈 간의 상호 작용, 즉 워크플로우(Workflow)를 표현해 준다.

엔티티 빈과는 달리 세션 빈은 데이터베이스 베이스내의 공유된 데이터를 표현하는 것이 아니라 공유된 데이터에 대해서 데이터를 읽고, 변경하고, 추가하는 등의 역할을 한다. 세션 빈은 특정 매개변수를 받아 요청된 사항을 수행한 다음, 결과를 반환하는 일련의 프로시져나 배치 프로그램으로 생각하면 된다. 세션 빈은 두 가지의 기본 형태, 즉 무상태(Stateless) 세션 빈과 상태유지(Stateful) 세션 빈으로 나뉘어 진다.

2.1.1 무상태 세션 빈

무상태 세션 빈은 메소드로 대표되는 관련된 서비스들의 집합으로, 빈은 한 번의 메소드 호출이 일어난 후 다음 번 호출까지 상태를 유지하지 않는다. 무상태 세션 빈에 메소드를 호출하면, 빈은 이전에 어떤 요청이 있었는지 아니면 이후에 어떤 요청이 있을지를 상관하지 않고 호출된 메소드를 실행한 다음 결과를 반환한다. 무상태 세션 빈은 소프트웨어 서비스처럼 일반적인 목적에 사용되거나 재사용 가능한 성격을 가지며, 대화 상태가 없기 때문에 활성화와 비활성화를 필요로 하지 않으므로 스와핑으로 인한 부하도 줄일 수 있다. 따라서 무상태 세션 빈은 간단하고 빠르다는 장점을 가진다.

2.1.2 상태유지 세션 빈

상태유지 세션 빈은 클라이언트 어플리케이션의 연장이다. 클라이언트 대신 작업을 수행하고, 그 클라이언트와 관련된 상태를 유지한다. 상태유지 세션 빈과 클라이어트간의 지속적인 대화를 나타낸다고 하여 이 상태를 '대화 상태(Conversational State)'라고 한다. 상태 유지 세션 빈에 호출된 메소드들은 이 대화 상태로부터 데이터를 읽고, 쓸 수 있는데 이 대화 상태는 해당 빈의 모든 메소드들에 의해서 공유된다. 상태유지 세션 빈은 주로 하나의 시나리오에 특화되는 경향이 있다. 즉 상태 유지 세션 빈은 2계층 시스템의 클라이언트 어플리케이션에서 찾아볼 수 있던 로직을 주로 나타낸다.

실용적인 측면에서 보면, 엔티티 빈은 어떤 개념을 나타내는 공유 데이터들의 집합에 대한 안정적이고 일관성 있는 인터페이스를 제공하기 위해 만들어 지며, 세션 빈은 어떤 개념을 확장한

데이터를 액세스하고, 공유되지 않으며, 대개 읽기 전용인 경우가 많다.

2.2 엔티티 빈

엔티티 빈의 경우, 자료의 지속성을 보장하는 방법에는 빈 관리 지속성(Bean-Managed Persistence)와 컨테이너 관리 지속성(Container-Managed Persistence)로 나뉠 수 있다.

2.2.1 빈 관리 지속성

빈 개발자가 직접 소스코드상에서 데이터베이스를 접근하여 자료의 지속성을 보장하며, 빈 인스턴스와 데이터베이스 사이에서 상태를 관리함에 있어 유연성을 준다. 빈 관리 지속성 방법에서의 EJB컨테이너는 빈 인스턴스에게 언제 데이터베이스의 데이터를 추가,변경,삭제를 하는 것이 안전한지에 대한 정보를 줄 뿐, 다른 어떤 도움도 주지 않으므로, 빈 인스턴스는 모든 지속성 작업을 스스로 한다.

2.2.2 컨테이너 관리 지속성

컨테이너 관리 지속성의 경우는 컨테이너에서 프라이머리 키(Primary Key) 클래스를 이용하여 자동적으로 관리해 줌으로서, 빈 개발자가 별도의 지속성 관리를 위한 코딩을 해 주지 않아도 된다. 두 가지 방법들은 각각 장·단점을 가지고 있으므로 업무의 성격에 따라 결정해 주어야 한다. 이러한 엔티티 빈은 재사용성을 높여 주고, 개발 비용을 줄여 줌으로써 개발 생산성을 높여 준다.

3. EJB 컴포넌트의 지속성

EJB 엔티티빈은 빈의 속성을 저장하는 방식과 어떤 조건에 맞는 빈을 검색하는 방법에 따라 빈 관리(Bean-Managed) 지속성과 컨테이너 관리(Container-Managed) 지속성으로 나뉜다. 두 가지 방식의 엔티티빈에서 모두 빈들을 식별하는 수단으로서 프라이머리 키 클래스(Pramary Key Class)를 정의해서 각각의 빈을 식별한다.

3.1 빈 관리 지속성

빈 관리 지속성 방식의 빈 개발에서는 개발자가 직접 엔티티빈의 속성들을 데이터베이스에 저장하기 위한 코드와 빈을 검색하는 코드를 직접 작성 해주어야 한다. 빈 관리 지속성 방식의 빈은 개발자가javax.ejb.EntityBean 인터페이스에 포함된 ejbLoad() 함수, ejbStore() 함수, ejbCreate() 함수, ejbRemove() 함수를 빈의 속성들을 읽어들이고 저장, 레코드 생성, 레코드 삭제하는 코드로 각각 오버라이드해서 개발한다.

```
...
public void setEntityContext(EntityContext ctx) throws RemoteException {
        this.ctx = ctx;
}
...
public void ejbLoad() throws RemoteException {
        AccountPK pk = (AccountPK) ctx.getPrimaryKey();
        String id = pk.accountID;

        Connection conn = getConnection();
        PreparedStatement stmt = conn.prepareStatement(
                "select ownerName, balance from accounts where id = ?");
        stmt.setString(1,id);
        ResultSet rs = stmt.executeQuery();
        rs.next();
        ownerName = rs.getString("ownerName");
        balance = rs.getDouble("balance");

}
```

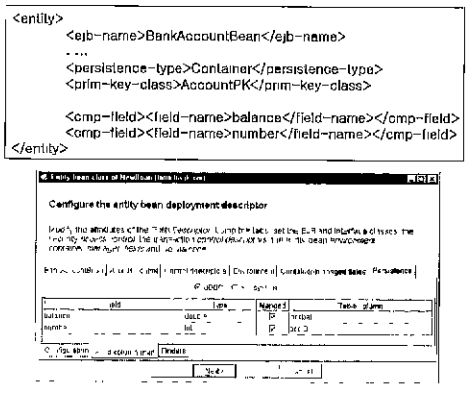
(그림 1) ejbLoad() 함수의 예

위 ejbLoad() 함수의 예는 빈 개발자가 빈의 속성에 데이터베이스에 있는 특정 레코드의 필드값을 대입하는 코드를 보여주고 있다. 위 예에서 setEntityContext()함수는 빈 인스탄스가 생성된 직후 호출되는 함수로서 EntityContext 객체를 호출시 전달한다. 이렇게 전달된 EntityContext객체는 ejbLoad() 함수에서 나타난바와 같이 현재 로드되고 있는 빈 인스탄스의 프라이머리 키를 얻기 위해 사용된다. 프라이머리 키를 얻어온 후 JDBC API를 이용하여 해당 프라이머리 키를 갖는 레코드를 가져온 후 빈 인스탄스의 속성에 레코드의 각각의 필드값을 대입함으로써 빈 인스탄스를 로

딩한다. 빈의 저장, 생성, 삭제 또한 이와 비슷한 방식으로 구현된다.

3.2 컨테이너 관리 지속성

컨테이너 관리 지속성 방식의 빈 개발에서는 개발자가 엔티티빈의 속성들이 데이터베이스에 저장되는 방식에 대한 코드를 작성하는 것이 아니라 Deployment Descriptor(DD)에 엔티티빈의 각각 속성과 데이터베이스 테이블의 필드를 연관시켜주는 내용을 기술함으로써 빈의 속성들이 데이터베이스에 저장되는 방법과 빈을 검색하는 방법을 정의할 수 있다. 컨테이너 관리 지속성 방식의 빈에서는 빈 관리 지속성 방식과는 다르게 개발자가 직접 빈의 속성들을 로딩하고 저장하는 코드를 작성할 필요가 없다. 대신에 DD에 빈의 속성들과 DB 테이블의 필드들을 연관짓는 내용을 기술함으로써 컨테이너에게 빈의 속성등과 DB 테이블사이의 관계 정보를 제공한다.



(그림 2) 컨테이너 관리 지속성 빈의 Deployment Descriptor 예

(그림 2)는 컨테이너 관리 지속성 빈의 속성들이 어떻게 DD에 기술되는지와 DD에 명시된 빈의 속성들이 Container에 의해 데이터베이스의 테이블과 어떻게 맵핑이 이루어지는지를 보여주고 있다.

4. 트랜잭션

트랜잭션은 어플리케이션 프로그래밍을 단순화시키는 입증된 기술이다. 어플리케이션 개발자로 하여금 프로그램 실행의 실패에 대한 복구와 다중 사용자 프로그래밍에 대한 복잡한 문제에 대한 처리를 해결 시켜준다. 그러나, 이러한 문제를 해결하기 위한 트랜잭션을 프로그래밍하는 것은 어려운 일이다. EJB에서는 트랜잭션이 기본적인 구성요소로 지원되므로 빈 구현자나 사용자 어플리케이션 개발자는 복잡한 분산 트랜잭션을 고려하지 않아도 된다. EJB의 트랜잭션은 단일 형태로 내부에 다른 트랜잭션을 포함하지 않으며, 크게 두 가지로 분류된다.

4.1 컨테이너 관리 트랜잭션

개발자가 트랜잭션을 위한 프로그래밍을 고려하지 않고, DD에 트랜잭션 속성을 기술함으로써 컨테이너에 의해 트랜잭션 처리가 이루어지도록 한다. 엔티티 빈의 경우는 컨테이너 관리 트랜잭션만을 이용할 수 있다. 컨테이너 관리 지속성으로 구현된 엔티티 빈의 경우는 컨테이너 제공자의 툴을 통해서 생성된 데이터 접근 클래스에 의해 관리된다.

컨테이너 관리 트랜잭션으로 구현된 빈의 경우는 DD를 통해서 트랜잭션 속성을 정의한다. 속성에는 6가지 종류가 있으며, 사용자 쪽의 트랜잭션(T1)과 빈 내의 트랜잭션(T2)을 고려하여 사용한다.

<표 1>은 6가지 트랜잭션 속성에 따라서 T1에 의하여 비즈니스 함수와 자원에 걸리는 트랜잭션의 범위를 보여준다. NotSupported는 T1에 상관없이 트랜잭션을 반영하지 않고, Required는 T1이 있으면 T1을 수용하고, 없으면 자신의 트랜잭션(T2)을 반영한다. Supports는 T1이 있으면 T1을 수용하지만 없으면 트랜잭션을 반영하지 않는다. RequiresNew는 T1에 상관없이 자신의 트랜잭션

〈표 1〉 트랜잭션 속성

속 성	사용자 트랜잭션	비즈니스 함수에 관련된 트랜잭션	자원 관리자와 관련된 트랜잭션
NotSupported	None	None	None
	T1	None	None
Required	None	T2	T2
	T1	T1	T1
Supports	None	None	None
	T1	T1	T1
RequiresNew	None	T2	T2
	T1	T2	T2
Mandatory	None	Error	N/A
	T1	T1	T1
Never	None	None	None
	T1	Error	N/A

(T2)을 실행한다. Mandatory는 T1이 없으면 비즈니스 함수에서 오류가 발생하고 자원은 유효하지 않게 된다. Never는 T1을 사용하지 않는다는 의미로 T1이 있으면 Mandatory의 T1이 없는 것과 같은 결과로 실행된다.

4.2 빈 관리 트랜잭션

개발자가 직접 트랜잭션 구현을 함으로써 컨테이너의 지원을 받지 않는다. 트랜잭션은 하나의 작업 단위가 완전히 수행되거나 동시에 수행되는 다중 작업이 가능한 여러 트랜잭션 간의 작업 단위가 분리되어 수행되어야 한다. 분리 수준을 관리하기 위한 API는 자원 관리자에게 특화되어 있기 때문에 EJB 구조에서는 분리 수준을 관리하기 위한 API를 정의하지 않는다.

5. Deployment Descriptor

DD는 ejb-jar 파일 생산자와 소비자 사이 간 계약의 일부이다. 빈 제공자에 의해 만들어진 하나의 ejb-jar 파일은 하나 이상의 엔터프라이즈 빈들을 포함하며 보통 어플리케이션 조립에 관련된 지시사항들은 포함하지 않는다.

DD의 역할은 ejb-jar 파일의 소비자를 위한 선언적 정보(예를 들면 엔터프라이즈 빈들의 코드에 직접 포함되지 않는 정보)들을 명시하는 것이다. DD에는 데이터베이스, 데이터 저장, 트랜잭션 관련 정보들을 포함하여 크게 다음과 같은 두 가지 정보들이 들어갈 수 있다.

5.1 엔터프라이즈 빈의 구조적 정보

구조적 정보는 하나의 엔터프라이즈 빈의 구조를 설명하고 그 엔터프라이즈 빈의 외부에 대한 의존성들을 선언한다. ejb-jar 파일 생산자는 DD에서 반드시 구조적 정보를 제공해야 한다. 구조적 정보가 변경되면 변경되지 않은 엔터프라이즈 빈의 평션들과 불일치가 일어나기 때문에 보통 구조적 정보는 변경될 수 없다.

5.2 어플리케이션 조립 정보

어플리케이션 조립 정보는 ejb-jar 파일 안의 엔터프라이즈 빈(또는 빈들)이 어떻게 더 큰 어플리케이션 배치 단위로 조립되는지 설명한다. 조립 수준의 정보는 엔터프라이즈 빈의 평션을 깨뜨리지 않고 변경될 수 있다. 하지만 그러한 변경으로 조립된 어플리케이션의 행위는 변화시킬 수 있다.

6. 결 론

본 논문에서는 EJB의 전체적인 구조에 대하여 소개하고, 세션 빈과 엔티티 빈의 차이, 지속성, 트랜잭션, DD에 대하여 간략하게 소개하였다. 현재 EJB는 버전 1.1이 공개되었으며 새로운 기능이 계속 추가되고 있다. 추가되고 있는 새로운 기술은 컴포넌트에서 이벤트를 받고 처리하는 기능과 컴포넌트 내의 데이타를 세밀하게 조회하는 기능 등이다. 현재 EJB는 전 세계적으로 컴포넌트의 표준으로 자리잡아가고 있으며 국내에서도 EJB를 기반으로 정보 시스템을 구축하는 사례들이 늘어나고 있다. 따라서 EJB의 기능과 이를 기반으로 한 컴포넌트 시스템 구축에 대한 연구 및 사례들이 앞으로 증가할 것이며 EJB 기반의 컴포넌트 시장도 활성화 될 것으로 예상된다.

참고문헌

[1] http://www.javasoft.com/ejb, "EJB 1.1 specification"

[2] E. Roman, Mastering Enterprise Java Beans, John Wily & Sons, 1999

[3] R. Monson, Enterprise Java Bean, O'Reilly, 2000

[4] 민현기, 김수동, 효율적인 EJB 트랜잭션 설계 기법, 한국정보과학회 소프트웨어공학 학술대회 논문집 Vol.2 No.1, p.42-50 2000년 2월

김 수 동

1984년 전산학 학사, Northeast Missouri State University
1988년 전산학 석사, The University of Iowa
1991년 전산학 박사, The University of Iowa
1991년-1993년 한국통신 연구개발단 연구실장/선임연구원
1993년-1994년 The University of Iowa, 교환교수
1994년-1995년 현대전자 S/W연구소 책임연구원
1995년-현재 숭실대학교 컴퓨터학부 교수
관심분야 : 객체지향 개발방법론, 객체지향 재사용 기법(Framework, Component-Based Software Engineering), 웹 기반 분산 객체 컴퓨팅(CORBA, Internet, Client-Server Web, Web Agents), 인터넷 상거래 시스템 기술