2002년 한국정보처리학회 춘계학술발표논문집 제 9 권 제 1 호

EJB 컨테이너를 위한 트랜잭션 관리의 설계

장철수, 김성훈, 노명찬, 서범수, 정승욱, 김중배
한국전자통신연구원
전자거래연구부 모바일응용서버연구팀
e-mail : {jangcs, saint, mcroh, bsseo, swjung, jjkim}@etri.re.kr

A Design of Transaction Management for EJB Container

ChoulSoo Jang, SungHoon Kim, MyungChan Roh, BeomSu Seo, SeungWoog Jung, JoongBae Kim
Mobile Application Server Research Team,
Electronic Commerce Department,
Electronics and Telecommunications Research Institute

요 약

EJB(Enterprise JavaBeans)는 서버 상의 Java 컴포넌트와 그 컴포넌트를 위한 동작 환경을 제공하는 전체 프레임웍(Framework)이다. EJB 에서 동작하는 컴포넌트를 엔터프라이즈 빈(Enterprise Bean)이라고 하고, 그 컴포넌트가 동작하기 위한 환경을 컨테이너(Container)라 한다. EJB 에서는 기존에 개발자가 직접 컴포넌트로 제공해 주어야 하는 트랜잭션, 영속성, 보안문제, 데이터베이스 연결 풀링, 쓰레딩 관리 같은 기능을 컨테이너에서 처리해 줌으로 일반 개발자는 복잡한 하부 로직에 신경쓰지 않으면서 더 쉽고 빠르게 동일한 기능을 구현할 수 있게 된다. 본 논문에서는 이러한 EJB 컨테이너를 설계함에 있어서 하나의 중요한 요소인 트랜잭션 관리에 관한 설계를 논한다..

1. 서론

웹 기반의 전자상거래가 폭발적으로 확산되면서 대규모의 웹 트랜잭션을 안정적으로 처리해 줄 기본 플랫폼의 필요성이 절실히 요구되었고 이에 엔터프라이즈급 웹 어플리케이션 서버(Enterprise Web Application Server)가 등장하게 되었다[4]. 그중 Sun Soft 사에서 제안한 엔터프라이즈 자바빈즈(Enterprise JavaBeans; EJB)[1]는 분산 컴포넌트 트랜잭션 서비스를 지원하기 위한 새로운 플랫폼으로써, 기존에는 컴포넌트로 만들어주지 않으면 개발자들이 일일이 신경써야 할 많은 기능들을 하부 구조에서 지원해주어, 개발자는 하부 시스템 로직에 신경쓰지 않으면서 더 쉽고 빠르게 동일한 기능을 구현할 수 있게 한다[5]. 이러한 기능을 제공하기 위해서 EJB 컨테이너는 네이밍 서비스나 트랜잭션 서비스와 같은 다양한 미들웨어와 연동되어야 한다. 본 논문에서는 EJB 컨테이너의 구현에 있어서 EJB Bean 이 필요로 하는 트랜잭션을 지원할 수 있도록 외부 JTA(Java Transaction API)[2]/JTS(Java Transaction Service)[3]와 연동하기 위한 트랜잭션 관리자의 설계 내용을 설명한다. 이 논문의 구성은 다음과 같다. 2 장에서는 EJB 규약에서 설명하고 있는 트랜잭션 지원의 주요 내용에 대해 살펴보고, 3 장에서는 개발 중인 전체 EJB 컨테이너의 간략한 구조에 대해 설명한다. 그리고, 4 장에서는 EJB 컨테이너의 구현에서 트랜잭션 관리자에 대한 설계 내용을 설명하고, 5 장에서 결론 및 향후 연구 과제를 알아본다.

2. EJB 의 트랜잭션 지원 규약

트랜잭션의 지원은 EJB 구조에서 필수적인 요소이다. EJB 의 트랜잭션은 다중 데이터베이스의 업데이트와 같은 작업을 가능하게 할 뿐 만 아니라, JMS 세션의 메시지 송수신과 다중 데이터베이스의 업데이트 트랜잭션 작업, 다중 EJB 서버 사이의 다중 데이터베이스의 트랜잭션 작업이 가능하다.

EJB 에서의 트랜잭션은 nested 트랜잭션과 같이 복잡한 트랜잭션은 고려하지 않고 단지 flat 트랜잭션만을 지원한다. 그리고, EJB 에서 트랜잭션은 크게 2 가지 방법으로 제공되는데, bean-managed transaction

demarcation (BMT) 방법과 container-managed transaction demarcation(CMT) 방법이 그것이다. BMT 에서는 javax.transaction 패키지의 UserTransaction 을 이용하는데, EJB Bean 의 코드 내에서 UserTransaction.begin() 과 UserTransaction.commit() 메쏘드 사이에 있는 자원들이 트랜잭션에 참여한다. 반면에 CMT 의 경우에는 EJB Bean 의 코드가 아닌 배포 명세서에 <표 1> 과 같이 트랜잭션 속성을 지정하면 EJB 컨테이너가 자동적으로 트랜잭션을 관리하도록 한다. 한 예로 트랜잭션의 속성이 Required 속성으로 명시된 메쏘드를 수행하는 경우, 만약 클라이언트가 트랜잭션을 갖고 있지 않은 상태에서 EJB 빈의 메쏘드를 호출한다면, EJB 컨테이너에 의해 자동적으로 새로운 트랜잭션을 생성하고 그 생성된 트랜잭션으로 메쏘드를 실행한다. 반대로 클라이언트가 트랜잭션을 갖고 EJB 빈의 메쏘드를 호출하였다면 클라이언트의 트랜잭션을 넘겨받아 메쏘드를 호출하도록 한다.

Attribute	Client's Transaction	Bean Method's Transaction
NotSupported	none	none
	T1	none
Required	none	T2
	T1	T1
Supports	none	none
	T1	T1
RequiresNew	none	T2
	T1	T2
Mandatory	none	error
	T1	T1
Never	none	none
	T1	error

<표 1> CMT 에서 트랜잭션 속성

<표 1>에서 보는 바와 같이 CMT 방식의 경우에는, 클라이언트 트랜잭션의 존재 유무가 빈의 메쏘드를 수행할 때의 트랜잭션을 결정하는 중요한 요소이지만, BMT 방식에서는 클라이언트의 트랜잭션이 아닌 인스턴스와 연관되어 있는 트랜잭션만이 트랜잭션을 결정하는 요소로 작용한다. 즉, BMT 방식에서는 EJB 컨테이너가 클라이언트의 트랜잭션을 잠시 중단시키고 인스턴스의 트랜잭션을 갖고 비즈니스 메쏘드를 실행한다. 비즈니스 메쏘드가 완료된 후에는 클라이언트의 원래 트랜잭션으로 복구시킨다.

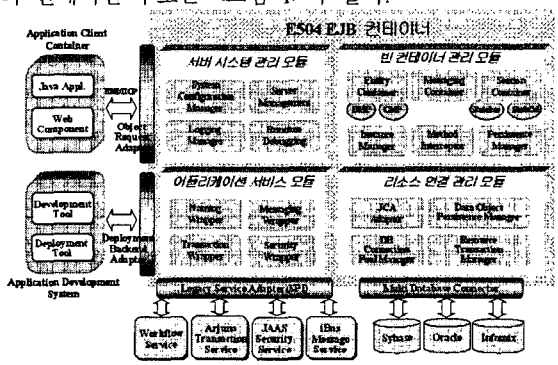
그리고, BMT 방식의 Stateless Session Bean 과 Message-driven Bean 의 경우에는 비즈니스 메쏘드가 완료되기 전에 트랜잭션이 완료되어야 하지만, Stateful Session Bean 의 경우에는 비즈니스 메쏘드가 완료되고 나서도 트랜잭션을 계속 유지할 수 있다. 이 경우에는 트랜잭션이 완료될 때까지 여러 번의 클라이언트 호출동안 EJB 컨테이너가 인스턴스와 트랜잭션의 연관관계를 유지하여야 한다.

Session Bean 이나 Message-driven Bean 의 경우에는 BMT 와 CMT 중 원하는 방법으로 구현하면 되지만, Entity Bean 의 경우에는 CMT 방법만으로 구현하여야 한다.

그리고, CMT 방식의 Stateful Session Bean 의 경우에 SessionSynchronization 인터페이스를 구현하는 경우가 있는데, 이러한 경우에는 트랜잭션의 시작 후에 afterBegin 을,트랜잭션의 완료 전,후에 beforeCompletion 및 afterCompletion 메쏘드를 EJB 컨테이너가 수행시켜 주도록 EJB 규약에서 요구하고 있다.

3. E504 EJB 컨테이너

현재 개발 중인 EJB 컨테이너인 E504 는 기존 시스템과 비교하여 여러가지 장점을 지니고 있다. 우선, 미들웨어 독립적인 구조를 지향하고 있어서 트랜잭션, 보안, 메시지 서비스와 같은 다양한 외부 미들웨어 시스템과 손쉽게 통합할 수 있는 구조를 제공하고 있다. 외부 미들웨어 중 트랜잭션 서비스는 Hanuri JTS, Arjuna JTS[6], OpenORB JTS, JBoss[7] 등을 지원하며, 인증 서비스는 Kerberos, X.509 를, 메시지 서비스는 Ibus, OpenJMS 등을 지원한다. 또한, E504 는 프로토콜 중립적인 구조로써 RMI-IIOP, JRMP 등의 프로토콜을 지원할 수 있는 구조를 제공한다. 또한 EJB 2.0 규약의 CMP (Container Managed Persistence) 및 CMR (Container Managed Relationship)을 지원하고 있다. E504 의 전체적인 구조는 <그림 1>과 같다.



<그림 1> E504 EJB 컨테이너 구조

E504 는 크게 4 개의 모듈로 구성되어 있는데 각 모듈은 다음과 같다.

- 서버 시스템 관리 모듈

  개발자 클라이언트로부터 배포된 어플리케이션의 탑재, 컨테이너의 생성, 서버에서 구동되는 각 관리자의 상태 정보의 모니터링 및 시스템 로그 정보를 수집하는 기능을 수행
- 빈 컨테이너 관리 모듈

  배포된 빈의 속성 정보 관리, 인스턴스 생명주기 관리, 빈 메소드에 설정된 미들웨어 서비스 적용, 빈 메소드를 실행하고, 예외 처리 기능을 수행
- 어플리케이션 서비스 모듈

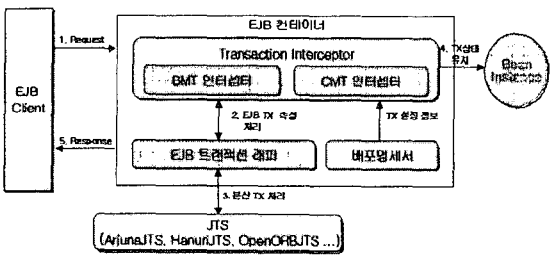
  개발자가 설정한 각종 미들웨어 서비스를 적용하기 위해서, 인증 및 접근제어 관리, 분산 트랜잭션 관리, 메시지 서버 연동 관리의 기능을 수행
- 리소스 연결 관리 모듈

빈과 영속성 관리자에서 사용하는 데이터베이스 연결 관리, JCA 기반 레거시 정보 시스템 연동 기능, 데이터베이스 연결의 트랜잭션 연동 관리의 기능을 수행.

E504 는 컨테이너가 실행하여야 할 중요한 기능들을 효율적으로 개발하기 위해서 오픈 소스로 공개되어 있는 EJB 컨테이너중 하나인 JBoss[7]의 인터셉터의 개념을 확장하여 수용하고 있다. 인터셉터의 개념을 통해서 컨테이너의 핵심적 기능을 분리하여 클라이언트의 메쏘드 호출이 여러 인터셉터를 거치면서 순차적으로 처리되도록 한다. 또한, 사용자 정의 인터셉터를 추가할 수 있는 구조를 제공하여 컨테이너의 기능을 쉽게 확장하여 커스터마이징 할 수 있도록 하고 있다.

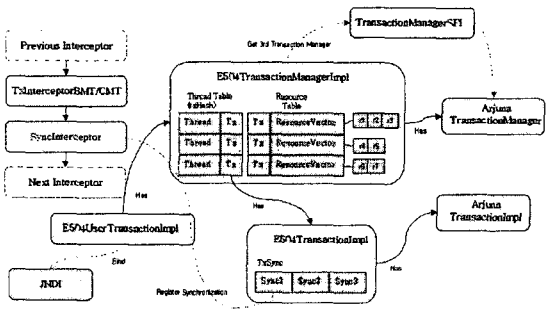
4. 트랜잭션 관리자

E504 EJB 컨테이너를 위한 트랜잭션 관리는 <그림 2>에서 보는 바와 같이 크게 2 가지의 모듈로 구성되어 있는데, 그 중 하나가 트랜잭션 래퍼(Wrapper) 이고, 다른 하나가 트랜잭션 인터셉터이다.



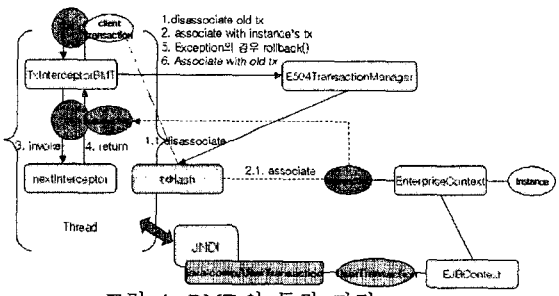
<그림 2> 트랜잭션 관리자의 개괄도

트랜잭션 래퍼는 분산 트랜잭션을 지원하기 위하여 외부 JTS(Java Transaction Service)를 사용할 수 있도록 하기 위한 래퍼(Wrapper)로써, 각 트랜잭션 인터셉터의 요청을 받아 외부 JTS 와의 연동을 통해 2 Phase Commit 이나 Synchronization 등을 지원한다. 트랜잭션 래퍼의 중요 클래스로는 <그림 3>과 같이 E504TransactionImpl, E504UserTransactionImpl, E504TransactionManagerImpl 등이 있다. E504TransactionManagerImpl 은 트랜잭션에 참여할 자원들을 관리하며 EJB bean 과 JTA/JTS 사이에 존재하여 bean 의 트랜잭션 관련 메쏘드에 대해 묵시적으로 관여하는 역할을 한다. 반면에, E504UserTransactionImpl 은 bean 의 트랜잭션 관련 메쏘드에 대해 명시적으로 관여하며, E504TransactionManagerImpl 에 메쏘드를 전달하여 수행하도록 한다. E504TransactionImpl 은 외부 JTA/JTS 의 트랜잭션을 래핑한 객체로써, 다중 synchronization 을 위해서 내부에 sync 벡터를 관리한다. 트랜잭션래퍼에는 이밖에 TransactionManagerSPI (Service Provider Interface)를 두어 외부 JTA/JTS 의 트랜잭션 관리자를 얻어 올 수 있는 일관된 방법을 제공한다.



<그림 3> E504 트랜잭션 관리자의 클래스

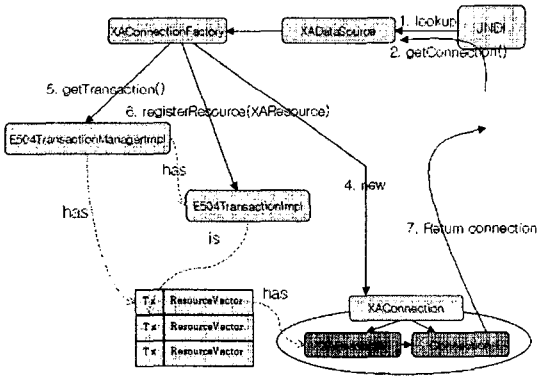
트랜잭션 인터셉터는 CMT 인터셉터와 BMT 인터셉터 각각이 존재하는데, CMT 인터셉터는 배포하려는 EJB 빈이 어떤 트랜잭션 속성으로 동작하여야 할지 배포명세서에 지정만 하면, 지정된 속성에 따라서 클라이언트의 트랜잭션을 계속 이어서 실행시키거나 새로운 트랜잭션을 생성하여 실행시키는 등, EJB 빈이 사용하려는 트랜잭션을 컨테이너가 자동적으로 관리한다. 반면 BMT 인터셉터는 EJB 빈 내에서 좀 더 세밀하게 트랜잭션의 경계를 관리할 수 있도록 <그림 4>와 같이 EJBContext 나 JNDI 를 통해 UserTransaction 을 제공하며, 클라이언트의 트랜잭션을 잠시 중단시키고 EJB 빈을 위한 새로운 트랜잭션을 생성하고 관리할 수 있게 한다.



<그림 4> BMT 의 동작 과정

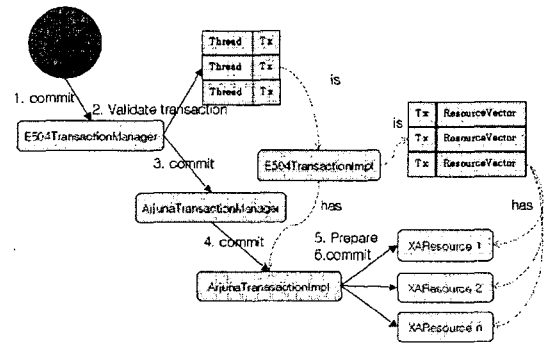
트랜잭션에 관련된 인터셉터 중 또 하나의 중요한 인터셉터가 있는데 바로 SyncInterceptor 이다. SyncInterceptor 는 SessionSynchronization 를 구현한 Session Bean 과 모든 Entity Bean 에 대해 callback 기능을 위해 Synchronization 처리를 해주는 인터셉터로써, 이미 트랜잭션에 참여한 Bean 인스턴스에 대해 또 다른 요청의 허용 여부를 결정하는 Re-enterant 처리도 담당한다.

트랜잭션에 참여할 자원은 트랜잭션 래퍼를 이용하여 관리하는데, <그림 5>는 트랜잭션에 참여할 자원의 등록 과정을 나타내고 있다.



<그림 5> 자원의 등록 과정

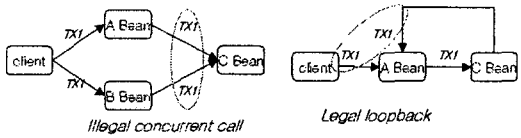
EJB 컨테이너나 EJB Bean 에서 트랜잭션에 참여할 자원이 필요한 경우 JNDI (Java Naming and Directory Interface)를 통해서 XADataSource 를 얻은 후, XADataSource 에게 Connection 을 얻는 메쏘드인 getConnection()을 호출하면, XADataSouce 는 XAConnectionFactory 를 호출하여 새로운 XAConnection 을 생성한 후, XAConnection 의 XAResource 를 트랜잭션 관리자에게 등록하고, XAConnection 의 Connection 을 돌려주게 된다. Connection 을 돌려받은 클라이언트는 Connection 을 통해 원하는 작업을 할 수 있으며 수행했던 작업에 대해서 트랜잭션 처리를 보장 받는다. 특히 다중 데이터베이스와 같은 분산 트랜잭션에서의 2 Phase Commit 을 보장받을 수 있다. 2 Phase Commit 을 하는 과정은 <그림 6>에 설명하고 있다.



<그림 6> 2 Phase Commit 과정

규모가 큰 비즈니스 환경에서는 다중 데이터베이스를 이용한 분산 트랜잭션이 필수적이다. 하지만, 소규모인 비즈니스 환경에서는 다중 데이터베이스가 아닌 단일 데이터베이스로 작업을 하는 경우가 많으며, 이런 경우에는 Prepare 과정과 Commit 과정으로 나뉘어지는 2 Phase Commit 을 하지 않고 바로 Commit 을 하는 것이 효율적이다. 이것을 위해서 E504 의 트랜잭션 관리자는 Resource Vector 를 관리하고 있어서, 트랜잭션에 참여하는 자원이 둘 이상이 아닌 경우에는 JTS 를 통한 2 Phase Commit 을 하지 않고 DBMS 에게 직접 Commit 을 요청하여 처리하도록 한다.

EJB 규약에서는 이미 트랜잭션에 참여하고 있는 하나의 entity 객체에 대해서 같은 트랜잭션 컨텍스트를 가진 클라이언트의 요청의 허용 여부 (re-entrant)를 배포 명세에서 지정하도록 하고 있으며 EJB 컨테이너는 지정된 대로 처리한다. 즉, Non-reentrant 로 설정되어 있는 경우에는 다른 request 를 처리하고 있는 동안, 같은 트랜잭션 컨텍스트를 가진 다른 client request 가 도착하면, exception 을 발생시킨다 그러나, 컨테이너는 <그림 7>과 같이 illegal concurrent call 과 legal loopback 을 구별하지 못하므로, bean 개발자는 loopback 을 가급적 쓰지 않아야 하며, callback 이 필요하지 않은 entity bean 은 배포 명세에 non-reentrant 로 설정하여야 한다.



<그림 7> Re-entrant 의 종류

5. 결론 및 향후과제

본 논문에서는 현재 개발 중인 EJB 컨테이너인 E504 에 대해 간략히 살펴 보았으며, E504 컨테이너에서의 트랜잭션 관리자의 설계내용을 설명하였다. 설계된 트랜잭션 관리자는 EJB Bean 이 필요로 하는 트랜잭션을 지원할 수 있도록 외부 JTS 를 연동할 수 있는 트랜잭션 래퍼와 호출되는 Bean 인스턴스의 트랜잭션을 결정하고 관리하는 트랜잭션 인터셉터로 설계되어 있다.

EJB 2.0 규약에서는 아직은 optional 로 명시되어 있지만, 서로 다른 업체에서 만들어진 컨테이너들 사이에서의 트랜잭션 상호연동(interoperability)을 지원하기 위한 설계내용의 추가가 향후과제로 남아있다.

참고문헌

[1] Linda G. Demichiel 외 2 인, "Enterprise Javabeans Specification, Version 2.0", Final Release, Sun Microsystems, 2001

[2] Susan Cheung 외 1 인, "Java Transaction API (JTA)", Version 1.0.1, Sun Microsystems, 1999

[3] Susan Cheung, "Java Transaction Service (JTS)", Version 1.0, Sun Microsystems, 1999

[4] 박재현, "전자상거래용 분산컴포넌트 개발에 관한 연구 최종연구보고서", 에이젠텍, 한국전자통신연구원, 1999

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

[6] "The Bluestone Arjuna Java Transaction Service 2.0 Programmer's Guide", Arjuna Solutions, 2000

[7] "JBoss - World class J2EE technologies in open source", http://www.jboss.org