EJB 엔터티 빈에 대한 동시성 제어 방법

정승욱[O] 김중배
한국전자통신연구원
{swjung[O], jjkim}@etri.re.kr

Concurrency Control Method for EJB Entity Bean

SeungWoog Jung[O] JungBea Kim
Electronics and Telecommunications Research Institute

요 약

EJB(Enterprise Java Beans)는 웹 응용 서버 스펙인 J2EE(Java2 Enterprise Edition)의 핵심으로서, 비즈니스 업무를 웹 환경에서 컴포넌트 형태로 작성하여 재 사용성을 높이기 위한 서버 측 컴포넌트 프로그래밍 모델이다. EJB는 컴포넌트를 특성에 따라, 일반적인 비즈니스 로직을 나타내는 세션 빈(Session Bean), 데이터베이스에 저장된 정보와 같은 여러 클라이언트에 의해 공유되며 영속 장치에 저장되는 엔터티 빈(Entity Bean) 그리고 JMS 메시지를 처리하는 메시지 드리븐 빈(Message-driven Bean)으로 구분한다. 본 논문에서는 ETRI에서 개발한 E504 EJB 서버에서 여러 클라이언트가 하나의 엔터티 빈에 동시에 접근할 경우 데이터 일관성 유지를 위한 동시성 제어 방법에 대해 논의한다.

1. 서 론

J2EE[1]는 썬(SUN)사의 주도로 개발되었으며 자바를 기업형 환경에 적용할 수 있도록 확장한 웹 응용 서버(Web Application Server)의 표준 프레임워크이다. J2EE는 웹 기반 비즈니스 컴포넌트를 개발하기 위한 기본 서비스를 제공 함으로서, 개발자에게 비즈니스 로직(logic) 만을 개발하도록 하고 기타 복잡한 과정은 자동적으로 처리해주는 환경을 제공한다.

엔터프라이즈 어플리케이션(Enterprise Application)은 JSP/Servlet으로 구성된 웹 컴포넌트와 EJB 빈으로 구성된 비즈니스 컴포넌트로 이루어지며, 웹 응용 서버로 배포되어 운영된다. JSP/Servlet은 웹 환경에서 사용자에게 동적 콘텐츠(Dynamic Contents)를 제공하며, EJB 빈은 비즈니스 로직을 담당하는 분산 객체 컴포넌트이다. J2EE 시스템은 JSP/Servlet과 같은 동적 컨텐츠를 담당하는 웹 컨테이너와 EJB 빈과 같은 분산 객체 컴포넌트를 담당하는 EJB 서버로 이루어져 있다.

EJB(Enterprise Java Beans)[2]는 웹 응용 서버 스펙인 J2EE(Java2 Enterprise Edition)의 핵심으로서, 비즈니스 로직을 컴포넌트 형태로 작성하여 재 사용성을 높이기 위한 서버 측 컴포넌트 프로그래밍 모델이다. EJB는 컴포넌트를 특성에 따라 일반적인 비즈니스 로직을 나타내는 세션 빈(Session Bean), 데이터베이스에 저장된 정보와 같은 여러 클라이언트에 의해 공유되며 영속 장치에 저장되는 엔터티 빈(Entity Bean) 그리고 JMS 메시지를 처리하는 메시지 드리븐 빈(Message-driven Bean)으로 구분한다.

세션 빈은 특정 클라이언트(Client) 마다 생성되는 컴포넌트로서 여러 클라이언트에 의해 공유되지 않는다. 반면, 엔터티 빈은 여러 클라이언트에 의해 공유되며 동시 접근이 가능하다. 하나의 엔터티 빈에 다수의 클라이언트가 동시에 접근할 경우 해당 엔터티 빈의 상태에 대한 일관성(consistency)이 파괴될 수 있다. 이를 위해 EJB 서버는 다수의 클라이언트 요청이 순차적으로(serially) 처리될 수 있도록 동시성 제어를 수행해야 한다.

본 논문에서는 다수의 클라이언트가 동시에 하나의 엔터티 빈에 접근할 경우 해당 엔터티 빈의 동시성 제어를 위한 락킹 메커니즘(Locking Mechanism)에 대해 논의한다.

2. EJB 서버 기본 구조

EJB 응용을 작성하기 위해서는 홈 인터페이스(Home Interface), 리모트 인터페이스(Remote Interface), 비즈니스 로직을 구현한 EJB 빈 등 세 개의 클래스를 작성해야 한다. EJB 클라이언트는 JNDI 네이밍 서버[3]에서 홈 인터페이스에 대한 레퍼런스를 얻은 후 홈 인터페이스를 통해 리모트 인터페이스에 대한 레퍼런스를 얻고, 이를 이용해 EJB 빈에 메시지를 보내게 된다. 이 메시지는 바로 빈 개발자가 작성한 EJB 빈에 전달되는 게 아니라, EJB 서버가 중간에 가로채 트랜잭션, 보안, 동시성 제어 등의 기본 기능을 수행 한 후, 해당 빈의 메소드를 호출하고 결과를 반환하게 된다.

그림 1은 ETRI에서 개발한 E504 EJB 서버 시스템의 전체 구조를 나타낸 것이다.

서버 시스템 관리 모듈은 어플리케이션 개발자로부터 배포된 어플리케이션의 탑재, 컨테이너의 생성, 서버에서 구동되는 각 관리자의 상태 정보의 모니터링 및 시스템 로그 정보를 수집하는 기능을 수행한다. 빈 컨테이너(Container) 관리 모듈은 배포된 빈의 속성 정보 관리, 인스턴스 생명주기(Instance Lifecycle) 관리, 빈 메소드 실행, 예외 처리 등의 기능을 수행한다.

어플리케이션 서비스 모듈은 개발자가 설정한 각종 미들웨어 서비스를 적용하기 위해서, 인증 및 접근제어 관리, 분산 트랜잭션 관리, 메시지 서버 연동 관리 기능을 수행한다.

리소스 연결 관리 모듈은 빈과 영속성 관리자에서 사용하는 데이터베이스 연결 관리, JCA 기반 레거시 정보 시스템 연동

기능, 데이터베이스 연결의 트랜잭션 연동 관리 기능을 수행한다.

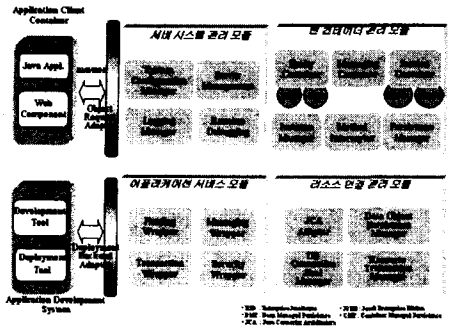

그림 1. E504 EJB 컨테이너 구조

클라이언트의 요청이 EJB 서버 내의 컨테이너에게 전달되면 컨테이너는 트랜잭션, 보안 등의 기본 서비스를 제공하게 된다. 컨테이너가 빈의 메소드를 호출하기 전, 인스턴스 관리자에게 해당 요청을 처리할 빈을 요구한다. 컨테이너는 해당 빈을 인스턴스 관리자로부터 제공 받은 후 빈의 메소드를 호출하고 결과를 클라이언트에게 반환한다.

상태 세션 빈은 특정 클라이언트에 종속된 빈으로서 클라이언트으로 부터 일정 기간 동안 요청이 없는 경우 활성화된 빈을 메모리에서 제거해 주도록 EJB 스펙은 규정하고 있다.

엔터티 빈의 경우에는 일정 시간 동안 클라이언트로부터 요청이 오지 않는 빈의 경우 무한정 활성 상태로 메모리 상에 상주하도록 하는 것은 메모리 부족 현상을 초래할 가능성이 높기 때문에 비활성화 시켜 주어야 한다. 또한, 기본 키가 같은 엔터티 빈에 대해 다수의 클라이언트가 동시에 접근이 가능하며, 다수의 클라이언트 요청이 동시에 동일한 빈에게 전달되면 빈 상태의 일관성이 파괴될 수 있다. 이를 방지하기 위해 컨테이너는 각 클라이언트 요청을 순차적으로 수행하게 해야 한다.

인스턴스 관리자(Instance Manager)는 빈의 생성, 활성화, 비활성화, 제거 등의 생명 주기(Life cycle)를 관리하고 엔터티 빈의 경우 동시성 제어 기능도 함께 수행 한다.

빈은 빈 인스턴스 뿐만 아니라 현재 해당 빈에 적용되어 있는 트랜잭션 및 기타 다양한 부가 정보를 가지고 있어야 하며 이를 빈 컨텍스트(Context)에 저장한다. 인스턴스 관리자는 실제로는 빈 인스턴스 및 부가 정보를 가지고 있는 빈 컨텍스트를 관리한다.

3. 엔터티 빈의 동시성 제어

엔터티 빈은 다수의 클라이언트에 의해 공유될 수 있다. 다수의 클라이언트 요청이 동시에 동일한 빈에게 전달되면 빈이 원하지 않는 상태가 될 수 있기 때문에 컨테이너는 엔터티 빈에 대한 동시성 제어를 수행해야 한다.

동시성 제어를 위해 주로 사용되는 방법은 아래와 같다.

첫번째 방법은 특정 기본 키를 갖는 엔터티 빈을 하나만 생성하고, 여러 트랜잭션이 해당 빈을 공유하게 하는 것이다. 여러 트랜잭션이 해당 빈을 접근하려고 할 때는 순차적으로 수행될 수 있도록 컨테이너가 보장해 주어야 한다. E504 컨테이너 시스템에서는 락킹 메커니즘을 통해 해당 엔터티 빈에 대한 락을 소유한 트랜잭션 만이 해당 빈을 접근하도록 하며, 트랜잭션이 종료되면 락을 놓아 다른 트랜잭션이 해당 빈을 접근하도록 하고 있다.

두번째 방법은 기본 키가 같더라도 수행되는 트랜잭션이 다르면 각 트랜잭션 마다 하나씩 엔터티 빈을 생성하게 하는 방법이다. 이 경우는 트랜잭션마다 하나의 엔터티 빈이 생성되므로 동시성 제어를 위한 부가적은 작업을 필요로 하지 않는다. 이 방법은 첫번째 방법보다는 속도가 빠르나 메모리는 더 많이 차지하게 된다.

엔터티 빈 호출 중 해당 빈의 다른 메소드를 호출할 수 있으며 이를 재진입(reentrancy)라고 한다.

재진입은 해당 엔터티 빈 상태의 일관성을 파괴할 수 있으므로 EJB 스펙에서는 재진입을 허용할 것인지 혹은 허용하지 않을 것인지를 빈마다 빈 배포자에 의해 설정 가능하도록 하고 있다. 같은 트랜잭션 상에서 엔터티 빈의 다른 메소드를 호출하더라도 재진입이 허용되지 않았을 경우는 예외(Exception)가 발생한다.

4. 엔터티 빈의 동시성 제어

클라이언트의 요청을 처리한 엔터티 빈은 인스턴스 관리자 내부의 캐쉬 관리자에 의해 메모리 내에 캐쉬된다. 해당 엔터티 빈이 트랜잭션 상에서 수행되었다면 해당 트랜잭션 정보는 해당 빈의 컨텍스트에 저장되어 관리된다.

클라이언트에 의해 해당 엔터티 빈에 대한 호출 요청이 컨테이너에 도달하게 되면, 컨테이너는 새로운 요청의 트랜잭션에 따라 다음과 같은 처리를 수행한다. 현재 빈에서 수행중인 트랜잭션을 기존 트랜잭션이라 하고, 새로운 요청 시 사용된 트랜잭션을 호출 트랜잭션이라 하자.

- 호출 트랜잭션이 기존 트랜잭션과 같을 경우
  : 해당 빈에 재진입이 가능하면 호출을 처리하고 가능하지 않으면 예외 발생
- 호출 트랜잭션이 기존 트랜잭션과 다를 경우
  : 기존 트랜잭션이 종료될 시까지 기다림

컨테이너는 EJB 락 관리자(Lock Manager) 통해 위와 같은 처리를 수행한다. E504 EJB 서버에서는 다양한 락 관리 정책을 제공하고, 또한 동시성 관리 정책을 용이하게 변경할 수 있도록 하기 위해 LockPolicy 객체를 제공한다. 락 관리자는 LockPolicy 객체에 의해 생성되고 관리되며, LockPolicy 객체는 인스턴스 관리자에 의해 생성되고 관리된다. LockPolicy 객체는 시스템의 환경 정보를 통해 빈 마다 다르게 지정할 수 있으며 변경 가능하다. 인스턴스 관리자는 LockPolicy 객체에 동시성 제어를 처리하도록 하며, 락에 대한 관리는 락 관리자가 동시성 처리는 LockPolicy 객체가 수행한다.

그림 2 는 EJB 락 관리자의 구조를 나타낸 것이다.

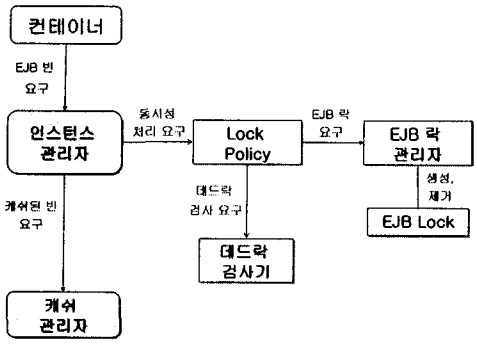

그림 2. EJB 락 관리자 구조

엔터티 빈에 대한 클라이언트 요청이 컨테이너에 도달하게 되면, 해당 컨테이너는 인스턴스 관리자에게 해당 요청을 처리할 EJB 빈을 요구한다. 인스턴스 관리자는 해당 빈이 인스턴스 관리자의 캐쉬 내에 존재하면 이를 반환하고 그렇지 않은 경우 데이터베이스에서 해당 빈에 대한 정보를 읽어서 반환하게 된다. 이때, 인스턴스 관리자는 LockPolicy 객체를 통해 동시성 제어를 수행하게 된다.

LockPolicy 객체는 호출 대상이 되는 빈에 대한 락을 락 관리자를 통해 얻게 된다. 만약, 해당 빈에 대한 락이 설정되어 있지 않으면 해당 호출은 허가되어 지고, 기존에 설정되어진 락이 있는 경우는 새로운 호출의 트랜잭션 및 해당 빈의 재진입 여부에 따라 달라진다. 해당 빈에 대한 락을 얻은 후 LockPolicy 객체는 데드락(Deadlock)이 발생했는 지를 검사하기 위해 데드락 검사기를 호출한다.

트랜잭션이 종료 되면 컨테이너는 해당 트랜잭션의 종료를 인스턴스 관리자에게 통보한다. 인스턴스 관리자는 LockPolicy에게 이를 통보하고, LockPolicy는 다른 트랜잭션이 사용할 수 있도록 해당 트랜잭션과 연관된 락을 해제한다.

LockPolicy 에서 수행되는 동시성 제어를 상세히 기술하면 다음과 같다. 현재 빈에서 수행중인 트랜잭션을 기존 트랜잭션이라 하고, 새로운 요청 시 사용된 트랜잭션을 호출 트랜잭션이라 하자.

■ 기존 트랜잭션이 존재하는 경우
- 호출 트랜잭션과 기존 트랜잭션이 다를 경우
  : 락 해제 시까지 기다림
- 호출 트랜잭션과 기존 트랜잭션이 같을 경우
  . 재진입이 허용된 경우 호출 허용
  . 재진입이 허용되지 않은 경우 예외 발생

■ 기존 트랜잭션이 존재하지 않는 경우
- 호출 트랜잭션이 존재하는 경우
  . 트랜잭션 없이 해당 빈이 호출되었을 경우 호출이 종료될 시까지 기다림
  . 트랜잭션 없이 해당 빈을 호출하고 있는 요청이 없는 경우이며 호출 허용
- 호출 트랜잭션이 존재하지 않는 경우
  . 트랜잭션 없이 해당 빈이 호출되었을 때 재진입이 허용된 경우는 호출이 허용되며 그렇지 않은 경우는 호출이 종료될 시까지 기다림

위의 LockPolicy에서 구현된 동시성 제어 방법은 일반적인 엔터티 빈에 적용되는 방법이며, 엔터티 빈의 특성에 따라 동시성 제어 방법은 달라질 수 있다. 예를 들어, 읽기 만이 허용된(Read only) 엔터티 빈의 경우 위의 동시성 제어가 필요 없게 되며, 갱신의 거의 발생하지 않는(Read most) 엔터티 빈의 경우도 동시성 제어 방법이 경우에 따라 달라질 수 있다. 또한, 엔터티 빈의 일관성이 중요하지 않을 경우에도 달라질 수 있다. 엔터티 빈의 동시성 제어 방법을 변경하기 위해서는 XML(eXtensible Markup Language)로 구성된 시스템의 환경설정 파일에 LockPolicy 객체를 변경하면 가능하다.

5. 결론

본 논문에서는 다수의 클라이언트가 동시에 하나의 엔터티 빈에 접근하였을 경우 해당 요청을 순차적으로 처리하는 동시성 제어 방법에 대해 논의 하였다. 동시성 제어 방법은 엔터티 빈의 특성에 따라 달라질 수 있으며, 이를 위해 엔터티 빈의 동시성 제어 정책을 용이하게 변경하도록 해야 한다.

향후, 시스템의 성능 향상을 위해 엔터티 빈의 특성에 따른 다양한 락킹 메커니즘의 개발이 필요하다.

참고문헌

[1] Java 2 Platform Enterprise Edition Specification, v1.4 Public Draft, SUN microsystems, July 15, 2002.

[2] Enterprise Java Beans Specification Version 2.0 Final Release, SUN microsystems, Auguest 22, 2001

[3] Java Naming and Directory Interface (JNDI), Version 1.2.1, Sun Microsystems, 1999

[4] Java Transaction API (JTA), Version 1.0.1, Sun Microsystems, 1999