EJB 2.1 타이머 서비스 설계 및 구현

정승욱[0] 이경호 김중배
한국전자통신연구원
(swjung[0], khleesun, jjkim)@etri.re.kr

Design and Implementation of EJB 2.1 Timer Service

Seung-Woog Jung[0] Kyeong-Ho Lee Joong-Bae 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. EJB 2.1에서는 기존 EJB 2.0에 기술된 기능 이외에 웹 서비스, 타이머 서비스, EJB QL 업그레이드 등의 기능을 추가하였다. 타이머 서비스는 지정된 시간마다 EJB 빈의 특정 함수를 호출하는 기능이다. 또한, 타이머 서비스는 트랜잭션과 연관된 경우 해당 트랜잭션 컨텍스트(context) 내에서 타이머의 롤백(rollback)을 지원해야 하며, 시스템의 고장 후 재시작 시에 기존 타이머의 복구 기능을 지원해야 한다. 본 논문에서는 EJB 스펙 2.1에서 제시한 타이머 서비스의 요구 사항에 대해 알아보고, ETRI 에서 개발한 E504 EJB 서버에서 타이머 서비스를 구현한 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.

EJB 스펙 2.1은 현재 Proposed Final Draft 2까지 발표되었으며, EJB 스펙 2.0에 웹 서비스, 타이머 서비스, EJB QL 업그레이드 등의 기능을 추가하였다.

본 논문에서는 EJB 스펙 2.1에서 제시한 타이머 서비스의 요구 사항에 대해 알아보고, ETRI 에서 개발한 E504 EJB 서버에서 타이머 서비스를 구현한 방법에 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 이용하여 EJB 빈에 요청을 보내게 된다. 이 요청은 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, 예외처리, 타이머 서비스 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, JCA기반 레거시 정보 시스템 연동 기능, 데이터베이스 연결의 트랜잭션 연동 관리 기능을 수행한다.

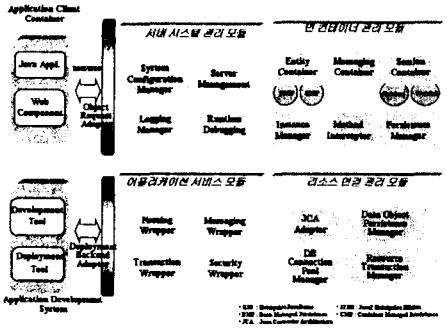

그림 1. E504 EJB 컨테이너 구조

3. 타이머 서비스 요구 사항

타이머 서비스는 특정 시점 혹은 특정 시간 간격마다 등록된 EJB 빈에게 통보를 해주는 서비스이다. EJB 타이머 서비스는 실시간 이벤트를 처리하기 위한 서비스가 아니라, 응용 레벨의 프로세스가 사용하는 정교하지 않은 시간 통보 서비스를 위해 사용되도록 개발되었다. 즉, 밀리 초까지의 정확한 통보를 요하는 실시간 서비스보다는 약간의 시간 통보에 지연이 있어도 크게 문제가 되지 않은 환경을 위해 EJB 스펙에 추가된 서비스이다.

3.1 빈 제공자의 관점에서 본 타이머 서비스

EJB 빈은 EJBContext 인터페이스를 통해 타이머 서비스를 접근할 수 있다. 타이머 서비스는 새로운 타이머의 생성, 기존 타이머의 취소, 특정 빈과 연관된 타이머 검색등과 같은 기능을 제공한다. 타이머 서비스를 사용하고자 하는 EJB 빈은 javax.ejb.TimedObject 인터페이스를 구현해야 한다. javax.ejb.TimedObject 인터페이스에는 ejbTimeout()라는 메소드가 존재한다. 타이머 생성 시 명시된 시간이 지나면 컨테이너는 등록된 빈의 ejbTimeout() 메소드를 호출한다. 타이머를 취소할 경우에는 타이머의 cancle() 메소드를 호출한다.
타이머의 생성, 취소, ejbTimeout() 메소드는 트랜잭션내에서 호출될 수 있다. 타이머는 영속성 객체이며, 시스템이 다운되거나, 빈이 활성화/비활성화 되어도 없어지지 않는다.

3.2 타이머 서비스 관련 클래스

타이머 서비스와 관련된 클래스는 다음과 같다.

- TimerService 인터페이스
  타이머 서비스를 통해 타임아웃(timeout)이 한번만 발생하는 타이머 혹은 일정 간격으로 발생하는 타이머를 생성할 수 있다. 타이머의 타임아웃은 타이머 생성 시점으로부터의 간격(duration) 혹은 특정 시점을 명시할 수 있다. 타이머 생성 시 특정 정보를 타이머 서비스에 전달할 수 있다. 이 정보는 타이머 서비스에 의해 관리되며 타이머 객체를 통해 접근할 수 있다.
- TimedObject 인터페이스
  타이머 서비스를 사용하고자 하는 EJB 빈은 javax.ejb.TimedObject 인터페이스를구현해야 한다. ejbTimeout() 메소드 내에서 타임 아웃 시 수행해야 할 비즈니스 로직을 처리할 수 있다. ejbTimeout()의 인자인 Timer 객체의 getInfo() 메소드를 통해 타이머 생성시 전달되었던 정보를 접근할 수 있다.
  컨테이너는 EJB 빈의 비즈니스 메소드나 ejbLoad(), ejbStore() 같은 라이프 싸이클 메소드 호출 사이에 ejbTimeout() 메소드를 호출할 수 있다. 이러한 경우에 EJB 빈의 락킹(locking) 방법에 따라 지정된 시간 보다 지연되어 ejbTimeout()이 호출될 수 있다. 여러 개의 타이머가 거의 동시에 타임 아웃이 되었을 경우, 타임 아웃 순서와 다르게 ejbTimeout() 메소드가 호출될 수 있다. 타이머를 취소 중에 타임 아웃이 발생하여 추가적인 ejbTimeout() 메소드 호출도 가능하며 이러한 경우 EJB 빈 개발자가 이를 해결해야 한다.
- Timer & TimerHandle 인터페이스
  빈 개발자는 Timer 인터페이스를 통해서 타이머를 취소하거나 타이머에 대한 정보를 얻을 수 있다. 또한 Timer 인터페이스를 통해 TimerHandle 인터페이스를 얻을 수 있다. TimerHandle 인터페이스는 직렬화 가능 객체이며, 이를 통해 Timer 객체를 얻을 수 있다.
- Transactions
  EJB 빈은 트랜잭션 범위 내에서 타이머를 생성할 수 있으며, 트랜잭션이 롤백(rollback) 되었을 경우 타이머 생성도 롤백되어야 한다. 또한, 타이머 취소도 트랜잭션 범위 내에서 호출될 수 있으며, 트랜잭션이 롤백되었을 경우 타이머 취소도 롤백되어야 한다. ejbTimeout() 시 트랜잭션이 롤백되면 컨테이너는 ejbTimeout() 연산을 다시 수행해야 한다.

4. 타이머 서비스

4.1 EJB 컨테이너의 역할

타이머 서비스를 제공할 EJB 컨테이너의 역할은 다음과 같다.

- TimerService, Timer, TimerHandle 인터페이스 구현
  컨테이너는 TimerService, Timer, TimerHandle 인터페이스에 대한 구현 클래스를 제공해야 한다. Timer 인스턴스는 직렬화가 가능하지 않아야 한다. 컨테이너는 Timer 객체가 존재하는 동안 TimerHandle을 사용할 수 있도록 해야 한다. 컨테이너는 Timer 객체의 equals(Object obj) 메소드와 hashCode() 메소드를 구현해야 한다.
- 타이머 타임 아웃 검사 및 ejbTimeout() 호출
  컨테이너는 타이머의 타임아웃을 검사하여 타임아웃이 되었을 경우, EJB 빈의 ejbTimeout() 메소드를 호출해야 한다.
  만약, ejbTimeout()메소드가 RequiresNew 트랜잭션 속성으로 호출된 후 해당 트랜잭션이 롤백 되었을 경우, 컨테이너는 적어도 다시 한번 ejbTimeout()을 호출해야 한다. 타이머의 타임아웃이 되었고 해당 빈이 비 활성 상태에 있을 경우, 해당 빈에 ejbActivate()를 호출하여 활성화 시키고 ejbLoad()를 호출하여 상태를 읽은 후 ejbTimeout() 메소드를 호출해야 한다. 타이머는 영속성 객체이므로, 컨테이너가 다운되더라도 타이머에 대한 정보는 컨테이너 복구 시 유지되어야 한다.
- 타이머 취소
  타이머가 취소되면 컨테이너는 해당 타이머를 제거해야 한다. 취소된 타이머에 메소드를 호출하면 javax.ejb.NoSuchObjectLocalException을 발생해야 한다.
  트랜잭션 상에 타이머가 취소되고 해당 트랜잭션이 롤백되었을 경우, 타이머 취소도 롤백되어야 한다.

- 엔터티 빈 삭제
  엔터티 빈이 삭제되면 해당 빈의 타이머도 삭제되어야 한다.

4.2 타이머 서비스 설계 및 구현

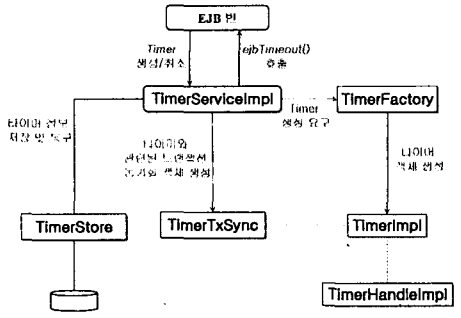

그림 2. E504 EJB 2.1 타이머 서비스 구조도

□ TimerServiceImpl 클래스
TimerServiceImpl은 TimerService 인터페이스를 구현한 클래스로서 EJB 빈의 타이머에 대한 요청을 받아 적절히 처리하는 역할을 수행한다. TimerServiceImpl 클래스의 역할은 다음과 같다.
- 타이머 생성 요청 처리
  EJB 빈의 타이머 생성 요청을 TimerFactory에 전달하고 생성된 타이머를 클라이언트에게 반환.
- 타이머 취소 요청
  EJB 빈의 타이머 취소 요청을 처리함.
- 타이머와 관련된 트랜잭션 처리
  EJB 빈의 타이머 생성, 취소 및 타임아웃 통보 처리 시 트랜잭션과 연관된 경우 트랜잭션 동기화 함수인 TimerTxSync를 생성하고 관리한다. 만약, 트랜잭션이 롤백 되었을 경우 타이머 생성/취소를 롤백한다. 타임아웃 통보 도중 롤백 되었을 경우에는 타임아웃 통보를 정해진 횟수 만큼 재 반복한다.

□ TimerFacoty 클래스
TimerFactory 클래스는 새로운 Timer 객체를 생성하는 역할과 TimerHandle을 통해 Timer를 생성하는 역할을 수행한다. TimerFactory는 상황에 따라 동일 기능을 수행하는 다른 클래스로 교체 가능한 플러그인(plugin) 구조로며, 이를 통해 다양한 타이머의 생성이 가능하다.

□ TimerImpl 클래스
TimerImpl 클래스는 타임 아웃 정보뿐만 아니라 해당 타이머와 연관된 트랜잭션 정보, 타이머 복구 시 필요한 부가 정보를 포함한다.

□ TimerHandleImpl 클래스
타이머 객체 자체는 직열화가 불가능하다. 타이머 객체를 파일이나 기타 저장 매체에 저장하고 향후 해당 타이머에 대한 정보를 얻고자 할 때, 타이머에 대한 TimerHandle은 직렬화가 가능하므로 이를 구하여 저장 매체에 저장하면 된다. EJB 컨테이너는 TimerHandle을 이용하여 적절한 타이머 객체를 반환하는 기능을 제공해야 한다.

□ TimerTxSync 클래스
타이머 서비스에 대한 요청을 수행할 때 해당 요청은 특정 트랜잭션 컨텍스트 하에서 수행될 수 있다. 만약, 해당 트랜잭션이 롤백 되었을 경우 타이머에 대한 요청도 롤백 되어야 한다. 이를 위해 타이머 서비스는 타이머 객체와 연관된 트랜잭션 정보를 저장해야 하며, 해당 트랜잭션이 롤백 되었는지 정상적으로 완료되었는지에 대한 콜백(callback) 메시지를 수신해야 한다. TimerTxSync 객체는 이러한 기능을 수행한다.
다음은 트랜잭션 롤백 시 수행해야 할 기능을 나타낸 것이다.
- 타이머 생성 요청 시 트랜잭션이 롤백 된 경우
  : 타이머 생성을 취소해야 한다.
- 타이머 취소 요청 시 트랜잭션이 롤백 된 경우
  : 타이머 취소 연산을 취소해야 한다.
- 타임아웃 통보 시 트랜잭션이 롤백 된 경우
  : 타임아웃 통보 연산을 최소한 한 번 이상 재 수행 해야 한다.

□ TimerStore 클래스
EJB 서버의 고장으로 인해 시스템이 다운(down)된 후 다시 복구되어 재시작 되었을 때 기존에 등록된 타이머도 복구되어야 한다. 이를 위해 생성된 타이머 및 타임아웃 정보를 저장해야 하며, 이러한 기능을 TimerStore가 수행한다. 타이머 서비스는 타이머가 생성되면 타이머 저장소에 저장하고, 타이머가 취소되면 해당 저장소에서 삭제한다. 또한, 타임아웃 관련 정보가 변경되면 타이머 저장소의 해당 타이머 정보를 변경한다. 시스템이 고장 후 재시작 되면 저장된 타이머를 복구하게 된다.

이상으로, E504 EJB 서버에서 구현된 타이머 서비스에 대해 설명하였다.

5. 결론 및 향후 연구

EJB를 사용하는 웹 응용에서 타이머 서비스에 대한 요구가 많아지고 있으며, 이를 위해 EJB 스펙 2.1에서는 타이머 기능을 추가하였다.
그러나, 현재 스펙에 기술된 타이머의 기능은 정해진 시간 간격에 따른 통보 기능 만 제공하고 있다. 향후, 특정 날짜 및 매주/매월 중 지정된 날에 통보하는 기능 등 다양한 통보 기능의 추가가 필요하다. 또한, 사용자가 통보 시간을 API를 통해 자유로이 변경할 수 있는 기능의 추가가 필요하다.

6. 참고 문헌

[1] Java 2 Platform Enterprise Edition Specification, v1.4 Public Draft, SUN microsystems, July 15, 2002.
[2] Enterprise Java Beans Specification Version 2.1 Proposed Final Draft 2 , SUN microsystems, June 2, 2003
[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