2001 년 한국정보처리학회 추계 학술발표논문집 제 8 권 제 2 호

EJB 기반의 뱅킹컴포넌트 시스템

안태광*, 김병기**
*㈜광주은행 정보지원부
**전남대학교 전산학과
e-mail : tkahn@kjbank.com

EJB Based Banking Component System

Tae-Kwang Ahn*, Byung-Ki Kim**
*Dept. of System Planning & Operating, Kwangju bank co.
**Dept. of Computer Science, Chonnam National University

요 약

EJB(Enterprise JavaBeans)는 컴포넌트 트랜잭션 모니터를 위한 표준 서버측 컴포넌트 모델로서 트랜잭션을 보장하고 확장성, 이식성, 안정성 등이 우수하며 분산 트랜잭션을 지원하므로 뱅킹시스템과 같은 복잡한 시스템을 구현하는데 적합한 모델이다. 새로운 어플리케이션을 개발할 경우에도 기존의 컴포넌트를 재사용함으로써 보다 쉽게 개발할 수 있으며 이런 장점은 사회의 변화와 시장환경에 민감하게 대처할 수 있는 신속한 상품개발과 배포 기능, 용이한 유지보수성을 요구하는 뱅킹시스템의 요건을 충족시킨다고 볼 수 있다. 본 논문에서는 EJB 뱅킹컴포넌트의 유용성을 예상해보고 트랜잭션처리에 있어서 시스템공통 처리부분과 업무단위별 처리부분으로 구분하므로써 구현의 복잡도와 구현상의 오류를 줄일 수 있는 보다 효과적인 아키텍쳐를 제안하며 그 처리흐름과 각 부분별 기능들을 정의해 본다.

1. 서론

인터넷시대의 빠른 환경변화에서 살아남기 위해서는 비즈니스 분석의 결과를 소프트웨어의 설계에 반영함으로써 비즈니스와 이를 지원하는 소프트웨어를 적절히 통합(Integration)하고 또 급속히 변하는 기술들을 적시에 반영할 수 있도록 적응력(Adaptation)이 높은 소프트웨어 개발을 추구해야 하는데 이런 요구사항을 만족시킬수 있는 새로운 소프트웨어 개발방법이 컴포넌트 기반 개발 방법(CBD : Component Based Development) 이다[1].

EJB(Enterprise JavaBeans)는 컴포넌트 기반 개발을 위한 기반기술의 하나로서 컴포넌트 트랜잭션 모니터를 위한 표준 서버측 컴포넌트 모델이다. EJB 는 분산 트랜잭션을 보장하고 확장성, 이식성, 안정성 등이 우수하여 뱅킹시스템과 같은 복잡한 시스템을 구현하는데 적합한 모델이다. 또 재사용성이 우수하여 새로운 어플리케이션을 개발할 경우에도 기존의 컴포넌트를 재사용함으로써 보다 쉽게 개발할 수 있으며 웹환경의 클라이언트를 지원하여 클라이언트의 버전관리문제를 해결할 수 있다. 은행이 사회의 변화와 시장환경에 민감하게 대처할 수 있는 신속한 상품개발과 배포 기능, 용이한 유지보수성을 지닌 시스템을 요구하고 있음을 감안할 때 객체지향/컴포넌트 기반의 EJB 뱅킹컴포넌트는 유력한 해결 방안 중의 하나라고 볼 수 있다.

EJB 컴포넌트를 사용함으로써 얻는 잇점은 앞서 기술한 내용외에도 1) S/W 개발 및 유지보수 비용절감, 2)객체지향 트랜젝션 처리 기법으로 분산 트랜잭션처리 가능, 3) 가벼운 클라이언트의 구현 가능 등을 들 수 있다.

본 논문에서는 트랜잭션의 처리흐름을 시스템에서 공통으로 처리할 부분과 단위 업무별로 처리할 부분으로 나누고 업무단위별 처리부분에서는 각 업무의 고유흐름만을 구현함으로써 구현의 복잡도를 줄이고 오류의 가능성을 줄일 수 있는 보다 효율적인 아키텍쳐를 제안한다. 또 제안된 아키텍쳐 내에서의 트랜잭션의 처리흐름과 각 부분별 기능들을 정의해 본다.

2. EJB 뱅킹컴포넌트에서의 트랜잭션 처리방법

2.1 트랜잭션 처리유형

EJB 컴포넌트에서 사용할 수 있는 트랜잭션 처리 방법은 컨테이너 관리 트랜잭션과 빈 관리 트랜잭션으로 나눌 수 있다. 컨테이너 관리 트랜잭션은 트랜잭션에 대한 관리를 EJB 컨테이너가 담당한다. 메소드 단위로 트랜잭션에 대한 요구 사항을 Deployment Descriptor 에 기술하게 되면, 트랜잭션 컨텍스트(Transaction Context)를 컨테이너가 자동적으로 생성, 전달하게 된다. 따라서 프로그램 내에서 User Transaction 을 생성, begin, Commit 할 필요가 없다. 이에 반해 빈 관리 트랜잭션은 개발자가 직접 트랜잭션 구현을 함으로써 컨테이너의 지원을 받지 않는 방식이다.

제안한 시스템에서는 컨테이너 관리 트랜잭션을 사용하여 감사용 로그(Audit Log)를 기록하는 부분만 트랜잭션에서 분리시키고, 모든 해당 업무는 반드시 하나의 트랜잭션으로 묶어 이에 대한 관리를 컨테이너가 담당하도록 한다. 하나의 트랜잭션에서는 해당 트랜잭션의 성공과 실패에 관계없이 반드시 하나 이상의 감사용 로그를 남기는 것을 원칙으로 한다.

2.2 트랜잭션의 시작과 종료시점

트랜잭션의 시작시점과 종료시점을 어디로 볼것인가의 문제는 분산 트랜잭션을 설계하는데 있어 매우 중요한 문제이다. 2 tier 클라이언트/서버환경에서는 클라이언트가 트랜잭션의 제어를 직접 담당하고 TP 모니터를 사용하는 3 tier 이상의 클라이언트/서버환경에서는 서버측에서 트랜잭션을 제어하는 것이 일반적이다.

EJB 컴포넌트 환경에서는 트랜잭션 처리모델을 어떻게 설계하느냐에 따라 달라질 수 있는데 JSP 와 같은 클라이언트를 그 시점으로 하는 방법과 EJB 빈 메소드(Bean method)를 시점으로 하는 두 가지 방법으로 나누어 볼 수 있다.

트랜잭션의 시작과 종료시점을 기존의 클라이언트/서버환경에서와 같이 클라이언트로 할 때와 EJB 빈 메소드로 할 때와의 장단점을 비교해 보면 다음과 같다.

1) 클라이언트 단위에서 트랜잭션을 처리하는 경우

장점	여러 분산된 자원에 대한 접근을 하나의 트랜잭션으로 관리할 수 있다. 트랜잭션을 발생시킨 쪽에서 트랜잭션 프로세스로 구성되어야 할 작업 단위를 명료하게 구분할 수 있으므로, 트랜잭션 시작과 종료에 대한 요구를 명확히 할 수 있다.
단점	어플리케이션 담당자가 전체 시스템 차원에서 제어되는 트랜잭션 관리에 대한 부분까지 알고 있어야 하므로 자칫 트랜잭션 구성을 잘못 할 수 있는 위험성을 안고 있다. 클라이언트 어플리케이션의 유지, 보수에 드는 비용이 증가한다. 클라이언트가 트랜잭션 객체에 원격으로 바인딩하여, 이에 대한 원격 메소드를 호출해야 하므로 하나의 트랜잭션이 완료되기까지 시간이 더 많이 걸린다.

2) EJB 빈 메소드단위로 트랜잭션을 처리하는 경우

장점	트랜잭션 제어를 서버 쪽에서 담당하므로 트랜잭션 구성과 관리를 일관성 있게 할 수 있다. 클라이언트 측에서는 원격 메소드를 한번만 호출하고, 이 메소드 내에서 트랜잭션에 대한 제반 사항을 담당하므로, 클라이언트에서 이를 제어하는 것보다 시간이 더 적게 소요된다.
단점	중첩 트랜잭션을 지원하지 않으므로, 여러 개의 작업이 하나의 트랜잭션으로 묶일 경우 이에 대한 구성 단위를 정확하게 파악하여 원격 메소드를 정의해야 한다.

제안한 시스템에서는 컨테이너 관리 트랜잭션을 사용하여 트랜잭션을 제어하고 EJB 빈 메소드 단위의 트랜잭션 처리 방법을 사용한다.

3. EJB 뱅킹컴포넌트의 시스템 아키텍쳐 및 처리흐름

제안한 EJB 뱅킹컴포넌트 시스템은 크게 클라이언트, 서비스제어 세션빈, 각 업무컴포넌트로 나누고 각 구성단위간 메시지 전달은 일반적인 자바빈즈를 사용한다. 클라이언트는 가벼운 클라이언트를 지향하여 웹 환경의 JSP 와 서블릿을 사용한다. 제안한 EJB 뱅킹컴포넌트 시스템에 적용한 기준을 정리하면 다음과 같다.

구분	기준
트랜잭션 관리유형	컨테이너 관리 트랜잭션
메소드별 트랜잭션속성	Required
트랜잭션의 시점	EJB 빈 메소드 단위
클라이언트	JSP, 서블릿
메시지 전달방법	자바빈즈
EJB 버전	EJB 1.1 or Higher

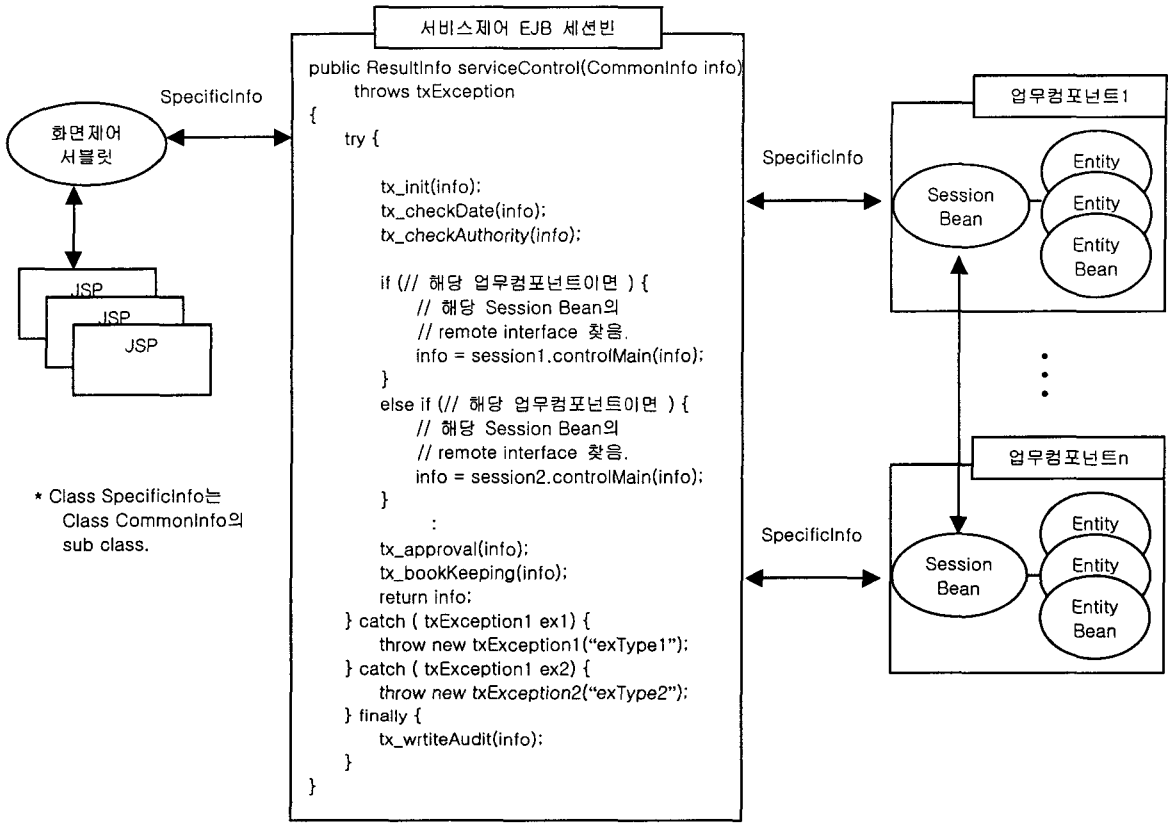
제안한 EJB 뱅킹컴포넌트 시스템의 시스템 아키텍처와 처리흐름을 도시하면 <그림 1>과 같다.

3.1 클라이언트

클라이언트를 표현하기 위한 JSP 들은 텔러로그인 시 거래영업일자,거래점번, 텔러번호, 단말유형 등의 기본정보를 세션에 저장하고 있다가 사용자의 입력사항을 추가하여 화면제어 서블릿에 제어를 넘긴다. 이때 입력된 내용에 대한 유효성체크는 JSP 가 담당한다. 세션에 저장되어 있는 정보는 서비스제어 세션빈에 의해 검증되어 유효한 사용자인지 확인받는다.

화면제어 서블릿은 JSP 로부터 받은 기본정보와 사용자의 입력사항을 조합하여 각 업무고유의 자바빈을 생성한 후 서비스제어 세션빈에 생성된 자바빈을 넘겨준다. 모든 처리가 끝난 후 서비스제어 세션빈은 그 처리결과 출력을 위한 자바빈을 생성하여 다시 화면제어 서블릿에 넘긴다.

화면제어 서블릿에서는 처리결과를 출력하기 위해 처리결과가 들어있는 자바빈을 Request 객체에 담아 JSP 로 넘기고 JSP 에서는 이 자바빈을 화면형식에



(그림 1) 시스템 아키텍처 및 처리흐름

맞게 출력한다. 혹시 서비스제어 세션빈으로부터 예외가 던져 진다면 해당 예외에 대한 적절한 오류메세지를 출력한다.

화면제어 서블릿이 서비스제어 세션빈에게 넘겨주는 기본정보(CommonInfo)는 <표 1>과 같다. (그림 1)에서 각 구성단위간 메시지 전달방법으로 사용된 SpecificInfo 는 CommonInfo 를 상속받아 업무고유의 입출력속성을 추가한 자바빈이다.

<표 1> CommonInfo 의 속성 예

Attribute	Comments
String txId	거래 ID
String txDate	거래일자
String txBusinessDate	거래영업일자
String txRequestTime	거래요청시간
String txBranchId	거래점번
String txTellerId	텔러번호
String txIpAddress	IP 주소
String txTerminalType	단말유형
String txCode	거래코드
String txClassName	서비스 클래스명
String txMethodName	서비스 메소드명
String resultCode	거래결과코드
Collection txApproval	승인내역
Collection txAccountBook	일계처리내역

3.2 서비스제어 세션빈

서비스제어 세션빈은 클라이언트 서블릿으로부터 SpecificInfo 을 넘겨 받은 후, 트랜잭션 처리를 위해 트랜잭션 ID 를 생성하고, 텔러의 로그인 시간을 확인하기 위해 SpecificInfo 의 거래영업일자와 실제 서버에 설정된 거래영업일자를 비교한다. 그리고 요청된 서비스의 실행가능 여부를 체크하기 위해 텔러의 담당업무에 따라 현재 요청된 서비스가 요청가능한 것인지의 여부, 서비스를 요청한 단말유형이나 업무가 현재 거래금지로 세트되어 있는지의 여부, 요청된 서비스가 가능한 시간인지 여부 등을 검사한다.

모든 검사가 정상이면 SpecificInfo 의 서비스 클래스명을 참고하여 서비스를 요청해야 할 컴포넌트와 세션빈을 알아낸 후 해당 세션빈에게 서비스를 요청한다. 서비스가 정상정으로 처리되면 세션빈으로부터 처리결과를 SpecificInfo 로 다시 반환받고 승인사항 처리와 일계처리를 각각의 해당 세션빈에게 요청한다. 승인처리는 일부 동기승인을 필요로 하는 트랜잭션에서 발생할 수 있는데 승인이 거부되거나 실패하면 예외를 발생시키고 해당 트랜잭션은 롤백된다. 일계처리 역시 그 처리가 정상적으로 이루어지지 못한다면 예외를 발생시키고 해당 트랜잭션은 롤백된다.

지금까지의 모든 과정은 하나의 트랜잭션으로 동작하게 되며 모두 정상이라면 그 결과를 SpecificInfo

에 포함하여 서비스를 요청했던 서블릿으로 반환한다. 업무컴포넌트의 세션빈으로부터 예외가 던져지면 서비스제어 세션빈은 감사용 로그기록을 제외한 이후의 모든 처리를 취소하고 해당 예외를 서비스요청 서블릿에 던져 예외발생을 알린다.

감사용 로그는 전술한 바와 같이 해당 트랜잭션이 정상완료이거나 실패이거나에 관계없이 모두 기록하며 사후감사용으로 사용할 수 있도록 <표 1>에서 기술한 내용을 위주로 트랜잭션처리에 관계된 대부분의 정보를 입력하게 된다.

지금까지 기술한 서비스제어 세션빈의 기능을 요약하면 <표 2>와 같다.

<표 2> 서비스제어 세션빈의 기능

메소드명	기능
tx_init()	거래 ID 생성
tx_checkDate()	거래시간, 영업일자 검사
tx_checkAuthority()	업무에 따른 사용자권한 검사 단말유형별 거래금지여부검사 업무별 거래금지여부 검사 서비스시간별 거래가능여부 검사
tx_approval()	책임자 동기승인 처리
tx_bookKeeping()	일계처리
tx_writeAudit()	감사용 로그 기록

3.3 업무컴포넌트

업무컴포넌트는 각 단위업무별로 필요한 엔티티빈과 업무로직을 통해 엔티티빈을 제어하는 세션빈으로 구성된다. 세션빈은 서비스제어 세션빈으로부터 서비스를 요청받으면 넘겨받은 SpecificInfo 의 서비스 메소드명을 참고하여 해당 메소드를 실행한다.

만약 업무처리 과정상 다른 업무컴포넌트를 이용할 필요가 있는 경우에는 (그림 1)에서 보인 바와 같이 해당 컴포넌트의 세션빈에게 직접 서비스를 요청할 수 있다. 이런 경우 트랜잭션속성이 Required 로 설정되어 있으므로 새로운 세션빈이 트랜잭션에 참가하더라도 이는 하나의 트랜잭션으로 관리된다.

트랜잭션처리에 필요한 세션빈과 엔티티빈을 이용하여 처리결과가 정상적이면 결과출력에 필요한 내용을 SpecificInfo 에 담아 서비스제어 세션빈에 반환하고 예외가 발생하면 해당 예외를 반환한다.

반환되는 SpecificInfo 에는 출력에 필요한 내용뿐 아니라 동기적인 승인처리를 위한 정보와 일계계정처리를 위한 정보도 포함되며 이들은 서비스제어 세션빈에서 승인처리와 일계처리를 위한 정보로 사용된다.

4. 결론

EJB 뱅킹컴포넌트는 EJB 컴포넌트 모델에 기반하여 확장성, 이식성, 재사용성, 안정성이 우수하며 완벽한 분산 트랜잭션을 지원하므로 새로운 어플리케이션을 개발할 경우에도 기존의 컴포넌트를 재사용하여 보다 쉽게 개발할 수 있다. 또 클라이언트를 웹으로 구성하는 경우 가벼운 클라이언트를 구현할 수 있으며 버전관리에 신경쓰지 않아도 된다는 장점이 있다. 이런 장점으로 인해 시장환경에 민감하게 대응할 수 있는 신속한 상품개발 및 용이한 유지보수가 가능하게 된다.

본 논문에서는 지금까지 EJB 뱅킹컴포넌트의 유용성과 그 효과를 예상해보고 트랜잭션처리에 있어서 시스템공통 처리부분과 업무단위별 처리부분으로 구분함으로써 구현의 복잡도와 구현상의 오류를 줄일 수 있는 보다 효과적인 아키텍쳐를 제안하였으며 그 처리흐름과 각 기능들을 기술하였다.

참고문헌

[1] 배두환. "e-Business 를 위한 컴포넌트 소프트웨어의 개발", 정보처리학회지, Vol.7, No.4, pp.27-32, .2000.7.

[2] 김재훈. "자바기반의 엔터프라이즈 이-뱅킹 컴포넌트 모델", 정보처리학회지, Vol.7, No.5, pp.51-57, .2000.9.

[3] Duane K. Fields & Mark A. Kolb. "Web Development with Java Server Pages", Manning, 1999

[4] Richard Monson-haefel. "Enterprise Javabeans, 2E", O'Reilly, 2000

[5] http://www.javasoft.com. "EJB2.0 - Proposed Final Draft 2", 2001

[6] http://e-docs.bea.com/wls/docs60/ejb/index.html . "Programming WebLogic Enterprise JavaBeans", 2001