EJB Dual Persistent 엔터티 빈 패턴을 이용한 효율적인 엔터티 빈 조립방법에 관한 연구

이창수[1] 이돈양 송영재
경 희 대 학 교 전 자 계 산 공 학 과
leechangs@hanmail.net dylee6211@hanmail.net yjsong@khu.ac.kr

A Study on Efficient Entity Bean Assembly Methodology Using EJB Dual Persistent Entity Bean Patterns

Chang-Su Lee[O] Don_Yang Lee Young-Jae Song
Dept. of Computer Engineering, KyungHee University

요 약

EJB는 컴포넌트 기반 개발(CBD)을 지원하는 컴포넌트 모델중의 하나이며 최근 들어 EJB를 이용한 활발한 어플리케이션이나 시스템 구축이 이루어지고 있다. 최근 증가하고 있는 EJB 컴포넌트들을 효율적으로 재사용 하기위한 필요성이 대두되고 있다. 본 논문에서는 EJB 컴포넌트 타입 중 엔터티 빈들 간의 조립을 목적으로 Dual Persistent 엔터티 빈 패턴을 이용한 효율적인 엔터티 빈 조립 방법을 제안하여 기존에 영속성에 대한 종속성이 강해 재사용이 쉽지 않던 BMP 엔터티 빈을 영속성이 독립적인 CMP 엔터티 빈으로 변환해 재사용 향상된 조립을 한다.

1. 서 론

컴포넌트 기반 개발은 이미 개발된 독립적인 컴포넌트를 재사용해 빠른 시간에 새로운 어플리케이션 및 시스템을 구축하는 방법이다. 컴포넌트 모델로는 EJB, CORBA, COM+ 등이 있으며 그 중에 EJB는 썬 사에서 제안한 컴포넌트 모델로서 J2EE 안에서 기업용 비즈니스 어플리케이션을 개발하고 운용하기 위한 서버측 컴포넌트 모델이다[1]. 자바로 작성된 EJB 컴포넌트들은 한번 개발되면 재 컴파일이나 소스 코드 수정 없이도 다른 EJB 어플리케이션 플랫폼에서도 그대로 재사용 가능하다[2,3,4] 향후 이런 EJB의 컴포넌트 모델에 의해 개발 배포된 컴포넌트(ejb.jar파일)들이 증가할 것이다. 결국 이런 기존에 개발 배포된 컴포넌트들을 새로운 어플리케이션 개발이나 시스템에 조립해 재사용할 필요가 있다. 컴포넌트 조립은 최근 까지 소개된 Catalysis, RUP, UML Component, KobrA등 많은 방법론들에 의해 강조되고 있으며 컴포넌트 기반 개발의 궁극적 목표로서 중요시 되고 있다[5,6].

컴포넌트(엔터티 빈)를 조립해 재사용함에 있어 기존에 개발된 컴포넌트와 새로운 컴포넌트와의 조립시 서로 다른 영속성과 호환성 같은 기능의 문제를 무시할 수 없다. 본 논문에서는 EJB로 개발된 기존의 컴포넌트를 새로운 어플리케이션이나 시스템으로의 조립을 위해 EJB Dual Persistent 엔터티 빈 패턴을 이용하여 서로 다른 영속성을 가지는 컴포넌트를 효율적으로 조립하는 방법을 제안하며 기존에 영속성에 대한 종속성이 강해 재사용이 쉽지 않던 컴포넌트들을 영속성이 독립적인 컴포넌트로 변환해 재사용이 향상된 조립을 한다.

2. 관련 연구

2.1 컴포넌트 조립

최근 컴퓨팅 환경은 클라이언트와 서버간의 N-tier 환경으로 분산된 환경에서 새로운 기능의 어플리케이션이나 시스템을 구축하는데 있어 기존의 컴포넌트를 재사용해 기능을 수행하고자 하는 필요성이 대두되고 있다.

컴포넌트 조립은 현재 컴포넌트 기반 개발(CBD)의 여러 분야에서 강조되고 있으며 몇몇 조립방법이 소개되었다. 대표적인 컴포넌트 조립 방법은 인터페이스간의 요구(require)와 공급(provide)에 의한 상호작용을 이용한 조립방법으로 인터페이스들간의 조립을 위해 포트를 이용한 조립, 래퍼방식을 이용한 조립이나 커넥터를 이용한 조립 등이 소개되었다. 또한 이런 조립방법을 EJB분야에 적용해 조립하고자 하는 연구의 필요성이 대두되고 있으나 아직 미약한 사항이며 대부분의 진행중인 EJB 조립이 세션 빈간의 조립으로 진행되고 있다[7].

2.2 엔터티 빈

EJB의 CMP(Container-Managed Persistence) 엔터티 빈은 EJB 컨테이너가 자동으로 영속성을 관리하며 개발이 용이하고 코드의 크기를 줄일 수 있다. 즉, 컨테이너는 빈의 모든 영속성 작업에 필요한 코드를 자동으로 생성하며 데이터베이스와 독립된다는 장점이 있다. BMP(Bean-Managed Persietence) 엔터티 빈은 CMP 엔터티 빈에서 컨테이너가 해주던 작업들을 개발자가 직접 명시적으로 해주어야 하며 CMP보다 유연하다는 장점을 가지지만 특정한 데이터베이스 타입이나 구조에 종속적이고 데이터베이스 또는 데이터 구조 내의 모든 변화가 빈 클래스의 구현에 영향을 미치는 단점이 있다. 따라서 다른 데이터베이스 환경에서 재사용할 때, 빈 내부의 메소드들에 포함되어 있는 질의들과 속성들의 수정을 요구한다. 이런 변경사항들에 대한 수정을 하는 것은 개발자들에게 많은 시간과 추가비용을 요구하게 되며 컴포넌트 기반 개발의 목적과 시장 적시성(time-to-market)의 성격에 많은 문제점을 드러낸다. CMP와 BMP의 비교분석은 표 1과 같다.

표 1. CMP와 BMP 비교표

	CMP	BMP
개발 시간	짧다	길다
엔터티 빈 성능	낮음	높음
재사용성	높음	낮음
영속성에 대한 독립성	높음	낮음

2.3 Dual Persistent 엔터티 빈

디자인 패턴의 개념을 EJB에 이용한 패턴 중 하나로 Dual Persistent 엔터티 빈 패턴은 영속성(persistence)이 다른 빈의 변경을 위해 비즈니스 로직을 CMP규약에 만족하는 슈퍼클래스와 BMP의 영속적인 로직을 담는 서브클래스의 두 개의 클래스로 분할함으로써 CMP와 BMP를 동시에 지원하도록 작성할 수 있는 패턴이다. 그림 1는 CMP와 BMP의 이중 영속성을 가진 클래스 다이어그램의 모습을 보여준다[8,9].

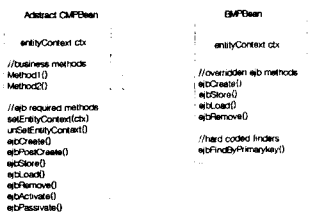

그림 1. Dual Persistent 엔터티 빈 패턴

2.4 디자인 패턴

디자인 패턴은 클래스가 어떻게 조립되어 있고, 각각의 클래스가 어떻게 연관되어 커다란 기능을 완수하는지를 표현하는 클래스 라이브러리보다 일반적인 개념이라 할 수 있으며 재사용성과 모듈성을 극대화 시킨 실제 구현과정에서의 해결책이다. 디자인 패턴 중 어뎁터(adapter) 패턴은 서로 다른 기능을 가진 즉, 이전 버전 혹은 레거시(legacy)시스템의 기능을 사용하기 위해 적용이 가능한 패턴이다.[10]

3. 엔터티 빈 조립방법

EJB 세션 빈과의 조립은 기존의 컴포넌트 조립 방법을 수정하거나 적용시켜 활발히 진행중이다. 하지만 서로 다른 영속성의 문제로 현재 엔터티 빈과의 조립이 미약하며 본 논문에서는 엔터티 빈과의 효율적인 조립방법을 제안하여 기존에 개발된 엔터티 빈과 새로운 엔터티 빈과의 기능의 결합을 동일 영속성을 가지도록 리팩토링된 이중 영속성 패턴을 적용하여 조립한다. 제안한 엔터티 빈 조립 방법은 다음과 같다.

첫째, 재사용 할 수 있는 엔터티 빈 저장소로부터 사용 하고자 하는 해당 엔터티 빈을 선택한다.

둘째, 선택된 엔터티 빈이 BMP인 경우 리팩토링된 이중 영속성 패턴을 적용하여 CMP로 작성한다. 단, CMP인 경우 과정생략.

셋째, 이중 영속성을 가지는 엔터티 빈을 배치시 CMP로 선택하고 BMP는 제거한다. 배치 디스크립터와 같이 변경 배치된 빈은 새로운 빈과 배치 디스크립터를 통해 조립한다.

넷째, 조립된 빈은 재사용이 가능한 엔터티 빈 저장소에 저장되어 재사용 가능한 형태로 존재한다.

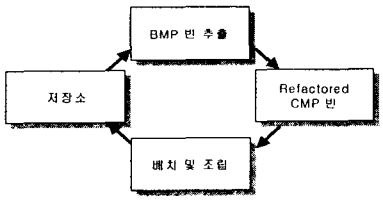

그림 2. 조립 작업 흐름도

3.1 엔터티 빈 추출

빈 저장소로부터 재사용 가능한 엔터티 빈을 선택하고 해당 빈의 영속성 타입을 검사한다. EJB 빈의 경우 배치 디스크립터를 포함하여 저장되어 있기 때문에 배치 디스크립터의 분석을 통해 원하는 엔터티 빈을 추출한다. 해당 엔터티 빈이 CMP로 작성되어져 있는 경우 이중 영속성 패턴 적용 없이 EJB 컨테이너에 배치시킨다. 하지만 기존에 개발되었거나 개발하고자하는 BMP의 경우 엔터티 빈의 재사용 향상과 동일 영속성을 가지는 빈 조립을 위해 이중 영속성 패턴 적용시킨다.

3.2 Refactored Dual Persistent 엔터티 빈

Dual Persistent 엔터티 빈 패턴은 CMP인 경우 BMP를 동시에 지원하기 위해 BMP부분을 추가하는 패턴으로 작성 시에는 CMP규약에 만족하도록 슈퍼클래스와 영속성 관련 메소드와 find 메소드등을 구현해야 하는 BMP로 분류하며 CMP와 BMP 엔터티 빈을 동시에 가지도록 작성한다.

엔터티 빈간의 조립과정은 재사용성과 데이터베이스에 대한 독립성이 높은 CMP와 CMP 엔터티 빈과 이루어 져야 하며 CMP와 BMP와 같이 영속성이 다른 빈들 간의 조립은 이중 영속성으로의 변경이 요구된다. 본 논문에서는 재사용성이 높은 CMP 엔터티 빈들 간의 조립을 위해 BMP인 경우 리팩토링된 CMP로 변환하기 위해 CMP 생성을 위한 슈퍼클래스를 만든 후 코드를 슈퍼클래스로 이동시키고 속성들과 속성접근자, 그리고 영속성 관련 메소드들 만을 서브클래스에 남겨둠으로써 기존에 개발된 BMP 엔터티 빈을 CMP로 전환한다. 그림 3은 CMP인 슈퍼클래스와 BMP인 서브 클래스의 두 가지 영속성을 가지는 하나의 엔터티 빈으로 배치 디스크립터인 ejb.jar.xml 파일에서 CMP인 슈퍼클래스와 BMP인 서브클래스로 작성된 엔터티 빈 사이에서 배치 디스크립터의 <persistence-type>요소 변경으로 영속성을 배치시 선택 변경할 수 있는 것을 보여준다. CMP 선택시에는 ejb.jar.xml 파일에 스키마와 속성들 그리고 find메소드등을 추가하기 위한 CMP에 특징적인 태그들로 구성되어야 한다.

본 논문에서는 재사용성이 높은 엔터티 빈 조립을 위해 BMP로 작성된 Account 엔터티 빈을 리팩토링된 CMP로 작성하였다. Account 엔터티 빈은 accountid와 balance의 두개의 속성을 가지고 있고 deposit, withdraw, balance의 3가지 비즈니스 메소드와 find 메소드를 포함한다. CMP 슈퍼클래스는 비즈니스 메소드와 추상 get/set 메소드들과 setEntityContext()/unSetEntityContext()와 같은 EJB 기본 메소드의 소스를 BMP로부터 가져와 간단한 구현만을 포함시키고 ejbCreate(), ejbLoad(), ejbStore(), ejbRemove()과 같은 영속성 관련 메소드의 구현사항은 BMP부분에 남겨둔다. BMP 서브클래스는 ejbActivate()/Passivate(), set/unSetEntityContext()와 같은 비즈니스 로직을 슈퍼클래스로부터 상속받는 형태로 작성한다.

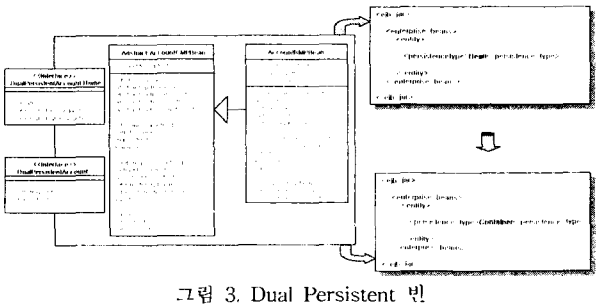

그림 3. Dual Persistent 빈

3.2 엔터티 빈 조립

리팩토링된 Dual Persistent 엔터티 빈은 조립시 BMP부분을 제거하여 ejb.jar.xml파일과 같이 CMP로 컨테이너에 배치시킨

다. 컨테이너에 배치할 리팩토링된 CMP 배치 디스크립터를 작성한 모습은 그림 4와 같다.

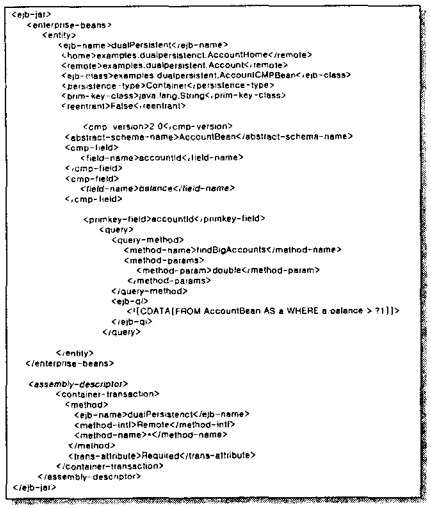

그림 4. Account 배치 디스크립터

영속성이 CMP로 변경된 엔터티 빈의 조립은 비즈니스 로직의 결합과 멤버필드들의 결합으로 이루어지며 클라이언트에 의한 기능 요구시 컨테이너가 엔터티 빈의 빈 인스턴스를 자동 생성한 후에 리모트 혹은 로컬 인터페이스들 간의 통신으로 비즈니스 로직을 결합한다. 실질적인 조립은 배치 디스크립터의 비즈니스 로직, 스키마, 속성들의 결합으로 엔터티 빈을 조립한다.

본 논문에서는 CMP와 CMP를 연결시켜주는 커넥터역할로 디자인 패턴의 한 부분인 래퍼 역할의 어댑터(adapter)패턴을 이용해 조립시 발생할수 있는 버전문제나 호환성과 같은 서로 다른 기능의 재사용성 문제를 해결하고자 하였으며 기존에 개발되어진 Account 빈의 기능을 새로운 시스템이나 어플리케이션 구축시 재사용 가능한 기능을 새로운 기능에 추가하여 사용할 수 있도록 하였다. 그림 5는 기존에 개발된 BMP가 CMP로 변경된 Account의 deposit, withdraw, balance 3가지 비즈니스 메소드의 재사용을 위해 새로운 BankAccount CMP 엔터티 빈 작성시 bankdeposit, bankwithdraw, bankbalance의 기능에 적합되어 조립되어 지는 모습을 나타낸다.

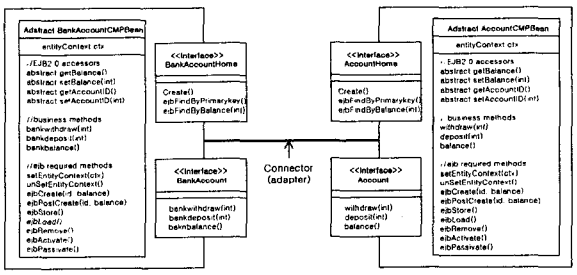

그림 5. CMP와 CMP 엔터티 빈 조립

래퍼 역할의 어댑터 패턴을 배치 디스크립터에 적용시키기 위해 본 논문에서는 PIML[11]구문을 적용시켜 배치 디스크립터에 어댑터 패턴을 매핑시켰다.

클라이언트에서 BankAccount의 비즈니스 메소드 호출시 실질적인 기능은 Account 비즈니스 메소드에서 수행하게 되며 배치 디스크립터에 래퍼역할로 어댑터를 적용시켜 조립한 모습을 그림 6은 보여준다. 배치 디스크립터에는 <structure>요소는 반드시 존재하여야 하며 요소 내에는 빈들 간의 구조와 역할 등이 위치하게 된다. 또한 <intent>와 <motivation>요소는 생략 가능하다.

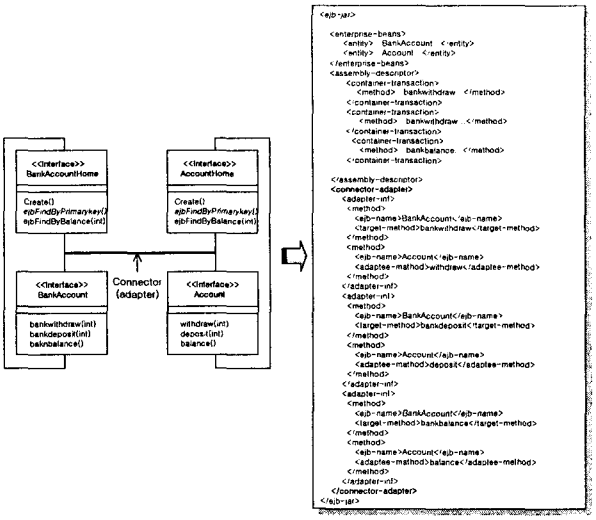

그림 6. 배치 디스크립터

5 결론 및 향후 연구

EJB로 구축된 서로 다른 기능의 엔터티 빈들 간의 조립이 가능해 짐으로써 기존의 세션 빈 조립에 국한되었던 EJB 컴포넌트 조립의 범위를 확장시킬 수 있고 EJB로 개발된 기존의 엔터티 빈의 기능을 재사용해 효율적으로 조립함으로써 재사용성 향상을 높일 수 있었다.

향후 연구로는 조립시 발생되는 의존성과 어느 정도의 재사용성 향상을 이룰 수 있는지 분석을 통한 엔터티 빈 테스트와 성능 평가가 요구된다.

참고 문헌

[1] Heineman. Councill, "COMPONENT-Based software engineering", Addison wesley, 2001
[2] JStorm. 박지훈, 이용원, 김종윤, 남궁윤, 신중식, 전부현. "EJB 엔터프라이즈 자바빈즈", 대청, 2001
[3] 프리렉. 유재우, 최재영, 최종명, 박춘서. "프로그래머를 위한 EJB", 2001
[4] Enterprise JavaBeans™ Specification, Version 2.0, Sun Microsystems, 2001
[5] John Cheesman & John Daniels, "UML Component", Cool Software Korea, 2001
[6] Colin Atkinson "Component-based Product Line Engineering with UML", Addison Wesley, 2002
[7] 최유희, 권호천, 신규상, "C2 스타일을 이용한 EJB 컴포넌트 합성 방법", 정보처리학회논문지D, 제8-D권 제6호, 2001
[8] Floyd Marinescu, "EJB™ Design Patterns", Wiley, 2002
[9] http://www.theserverside.com
[10] Eric Gamma, R. Helm, Richard Johnson and J. Vlissides, "Design Patterns : Elements of Reusable Object-Oriented Software", Addison-Wesley, 1995
[11] M. Ohtsuki, N. Yoshida, "A Source Code Generation Support System Using Design Pattern Document Based on SGML", Proc. of the APSEC 98, 1998