제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003. 5)

BMP 기반 엔티티 빈의 재사용성과 유지보수성 향상 방안

김고운*, 이금석*
*동국대학교 컴퓨터공학과
e-mail:{gowoon, kslee} @dgu.ac.kr

Improving Reusability and Maintainability of Entity Bean in Bean-Managed Persistence.

Go-Woon Kim*, Keum-Suk Lee*
*Dept. of Computer Engineering, Dongguk University

요 약

EJB는 J2EE 기반의 분산 컴포넌트 모델로 재사용 가능한 소프트웨어이다. 그중 데이터베이스의 데이터를 객체화한 엔티티 빈 컴포넌트는 영속성에 따라 CMP와 BMP로 구분이 되는데, BMP는 영속성의 차이로 인해 CMP와 조립하기 어려워 재사용성이 떨어지고, 소스코드가 복잡해 유지보수가 어렵다. 본 논문에서는 EJB 컴포넌트 중에서 BMP를 기반으로 구현한 엔티티 빈에 Dual Persistent 엔티티 빈 패턴(Entity Bean Pattern)을 적용하여 재사용성을 향상시키고, 소스코드를 리팩토링(Refactoring) 하여 유지보수성 향상시키는 DPwR(Dual Persistence with Refactoring) 방법을 제안하였다.

1. 서론

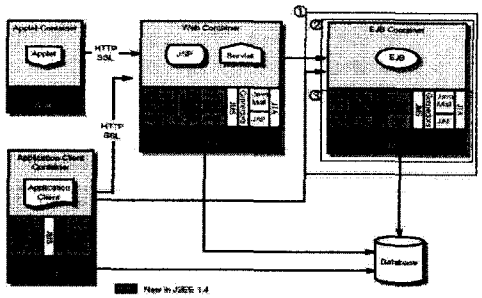
소프트웨어 컴포넌트의 주된 목적은 소프트웨어의 재사용[1]과 쉽게 대체 가능하여 소프트웨어의 관리를 용이하게 하는데 있다[2]. 개발되어 공표된 개별적인 소프트웨어 컴포넌트는 재 컴파일이나 소스 코드 수정 없이 재사용이 가능한데[3], 이러한 컴포넌트 모델로는 EJB, CORBA, COM+ 등이 있다. 그 중에 EJB(Enterprise JavaBeans™)는 Sun Microsystems사가 스펙을 제안한, J2EE(Java™ 2 Platform. Enterprise Edition) 아키텍처 기반의 분산 컴포넌트 모델로, 비즈니스 로직을 포함한 재사용 가능한 소프트웨어의 집단이다[4]. 컴포넌트 기반 소프트웨어 개발이 소프트웨어 위기의 해결 방안으로 주목을 받으면서 EJB의 컴포넌트 모델에 의해 개발되어 배포되는 컴포넌트들이 증가하고 있다.

엔티티 빈은 영속성을 관리하는 방법에 따라서 CMP(Container-Managed Persistence) 엔티티 빈과 BMP(Bean-Managed Persistence) 엔티티 빈으로 나눌 수 있다. BMP 엔티티 빈은 실행 환경에 종속적으로 구현되기 때문에 성능 면에서는 CMP 엔티티 빈보다 뛰어나지만 재사용과 유연성이 떨어진다[5]. 본 논문에서는 기존에 개발된 BMP 엔티티 빈에 Dual Persistent 엔티티 빈 패턴을 적용하여 환경에 따른 CMP, BMP 선택이 가능하게 함으로써 재사용성을 향상시키고, 소스코드에서 메소드를 분리하는 리팩토링을 수행하여 유지보수성을 향상시키는 방법을 제안하였다.

2. 관련연구

2.1 J2EE 구조

EJB 애플리케이션 컴포넌트(Application Components)를 관리하기 위한 J2EE 플랫폼의 구성 요소들 사이의 관계는 [그림 1]과 같다. ①은J2EE 런타임 환경(*Runtime Environment*)으로서의 EJB 컨테이너(Container)를 나타내는데, ②의 EJB 애플리케이션 컴포넌트에 필요한 서비스를 제공해준다. ③은 컨테이너가 EJB 애플리케이션 컴포넌트에 제공하는 서비스들을 나타내고, 각각의 화살표는 J2EE 플랫폼의 다른 부분으로의 접근을 나타낸다. ③에서 볼 수 있듯이 J2SE(Java™ 2 Platform, Standard Edition) 런타임 환경의 API가 각각의 애플리케이션 컴포넌트 타입을 지원한다[6].



[그림 1] J2EE 구조도

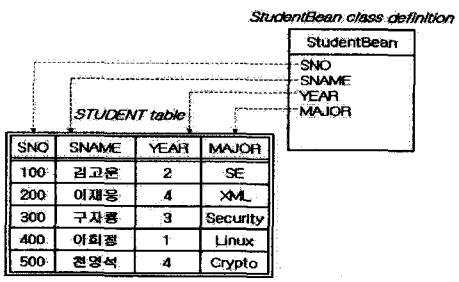
2.2 EJB 객체형

EJB의 객체에는 세션 빈(Session Bean)과 엔티티 빈(Entity Bean), 메시지 드리븐 빈(Message-driven Bean)으로 구분된다.

세션 빈은 단일 클라이언트만이 하나의 빈 인스턴스를 사용할 수 있는 형태로 영속성이 없는 일반적인 비즈니스 로직을 구현할 때 사용된다. 엔티티 빈은 데이터베이스와 같이 영속성이 있는 저장장치의 데이터를 객체화한 컴포넌트를 말한다[3]. 메시지 드리븐 빈은 EJB 스펙 버전 2.0이 발표되면서 새로 추가된 기능[9]으로, JMS 메시지를 처리하고 비동기적인 호출을 특징으로 갖는 빈이다.

2.2.1 엔티티 빈(Entity Bean)

데이터베이스의 데이터에 대한 객체 뷰(view)를 제공하는 빈으로, 객체에 여러 사용자가 동시에 접근하는 것을 허용한다. [그림 2]는 관계형 데이터베이스 테이블과 엔티티 빈 객체와의 관계를 보여준다[7].

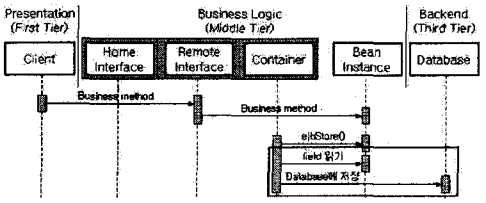


SNO	SNAME	YEAR	MAJOR
100	김고운	2	SE
200	이재웅	4	XML
300	구자룡	3	Security
400	이희정	1	Linux
500	천영석	4	Crypto

[그림 2] 엔티티 빈과 데이터베이스의 매핑

엔티티 빈이 데이터베이스에 업데이트를 수행하던 중에 EJB 컨테이너가 crash되면 클라이언트에게 예외를 발생시키고, 자동적으로 트랜잭션 이전의 상태로 되돌린다[3]. 영속성(Persistence)은 객체나 객체의 정보가 애플리케이션의 생명주기나 클라이언트, 서버 프로세스와 관계없이 지속되는 것을 뜻하는데[5], 엔티티 빈은 데이터의 영속성을 어떻게 관리하는지에 따라 CMP와 BMP로 나눌 수 있다.

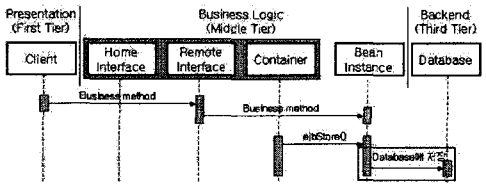
① CMP(Container-Managed Persistence) : 빈의 영속성을 EJB 컨테이너가 자동으로 관리하는 엔티티 빈을 말한다. 빈 인스턴스의 생명주기를 관리하는 콜백 메소드는 대부분 컨테이너에 의해 생성되기 때문에 콜백 메소드는 빈 메소드로 남겨둔다. 따라서 소스코드가 간단해지며, 특정 데이터베이스에 종속적이지 않기 때문에 재사용성과 유연성이 좋다. 그러나 데이터베이스 환경에 특성화된 코딩이 불가능하므로 비효율적인 빈이 생성된다. [그림 3]은 CMP 엔티티 빈의 상태를 데이터베이스에 저장하는 순차도이다.



[그림 3] CMP 엔티티 빈 순차도

컨테이너에 의해 ejbStore() 메소드가 호출되면, 컨테이너는 빈 인스턴스를 거치지 않고 컨테이너가 생성한 절차에 따라 인스턴스의 상태를 데이터베이스에 저장을 한다.

② BMP(Bean-Managed Persistence) : CMP 엔티티 빈과 달리 개발자가 직접 영속성과 관련된 로직을 코딩하기 때문에 환경에 따른 최적화된 빈을 생성할 수 있다. 그러나 개발자는 데이터베이스 구조에 대한 이해가 있어야하고, SQL문을 처리하는 모듈이 엔티티 빈에 함께 구현되므로 코드가 복잡하다. [그림 4]는 BMP 엔티티 빈의 동기화 과정이다.

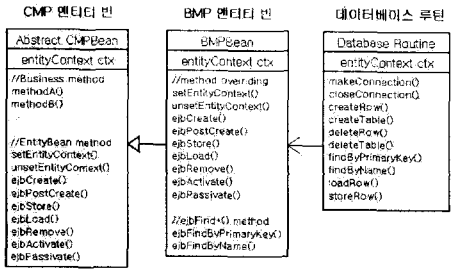


[그림 4] BMP 엔티티 빈 순차도

컨테이너에 의해서 호출되어지는 ejbStore() 메소드는 빈 인스턴스에 구현이 되어 있기 때문에, 구현된 메소드를 통해서 데이터베이스에 빈 인스턴스의 상태를 저장한다.

3. DPwR 방안

본 논문에서 제안하고 있는 DPwR 방안은 Dual Persistence with Refactoring의 약자로 BMP 엔티티 빈의 재사용성과 유지보수성을 향상을 목표로 하고 있다. [그림 5]는 DPwR의 구조를 나타내고 있다.

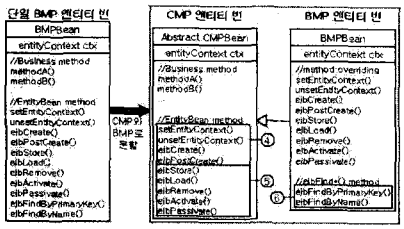


[그림 5] DPwR 구조

3.1 DPwR의 재사용성 향상방안

3.1.1 Dual Persistent 엔티티 빈 패턴

DPwR 은 BMP 엔티티 빈의 재사용성을 향상시키기 위해 Dual Persistent 엔티티 빈 패턴을 적용하였다. [그림 6]은 CMP와 BMP를 모두 지원하는 엔티티 빈 디자인 패턴인 Dual Persistent엔티티 빈 패턴[8]을 나타낸다. Dual Persistent 엔티티 빈 패턴은 단일 BMP 엔티티 빈의 비즈니스 로직을 CMP 규약에 만족하는 슈퍼클래스와 BMP의 영속적인 로직을 담는 서브클래스로 클래스를 분할함으로써 CMP와 BMP를 동시에 지원하도록 작성하는 패턴이다. 슈퍼클래스인 CMP 엔티티 빈에는 비즈니스 메소드와 필수 EJB 메소드인 set/unsetEntityContext(), ejbCreate()([그림 6]의 ④) 등을 구현하고, 컨테이너에서 자동으로 구현하는 ejbStore(), ejbLoad(), ejbRemove()([그림 6]의 ⑤) 등의 메소드는 구현하지 않는다. BMP 엔티티 빈에서 반드시 필요한 findByPrimaryKey() 등의 Finder 메소드는 컨테이너가 자동으로 구현하므로 ejb-jar.xml 파일에서 선언만 해주면 된다.



[그림 6] Dual Persistent 엔티티 빈 패턴

BMP 서브클래스는 CMP 슈퍼클래스를 확장한 클래스로, 슈퍼클래스에서는 구현하지 않은 영속성에 관련된 메소드인 ejbCreate(), ejbLoad(), ejbStore(), ejbRemove() 등을 구현하고, Finder 메소드([그림 6]의 ⑥)를 구현한다.

3.1.2 재사용성 향상을 위한 구현 방안

Dual Persistent 엔티티 빈을 적용하면 쉽게 BMP 엔티티 빈과 CMP 엔티티 빈 두 가지로 사용할 수 있다. 배치 디스크립터는 BMP와 CMP에 따라 따로 작성하고, 환경에 따라 배치 디스크립터인 ejb-jar.xml 파일을 변경하기만 하면 재 컴파일이나 재 프로그래밍 없이 컴포넌트를 변경할 수 있다. [그림 7]은 BMP 엔티티 빈의 배치 디스크립터의 내용을 보여준다.

```
<ejb-jar>
 <enterprise-beans>
  <entity>
   <ejb-name>StudentEJB</ejb-name>
   <home>example.StudentHome</home>
   <remote>example.Student</remote>
   <ejb-class>example.StudentBean</ejb-class>
   <persistence-type>Bean</persistence-type>
   <prim-key-class>java.lang.String</prim-key-class>
   <reentrant>False</reentrant>
   <resource-ref>
    <res-ref-name>jdbc/StudentDS</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
   </resource-ref>
  </entity>
 </enterprise-beans>
 ...
</ejb-jar>
```

[그림 7] BMP 배치 디스크립터

```
<ejb-jar>
 <enterprise-beans>
  <entity>
   <persistence-type>Container</persistence-type>
   ...
   <cmp-version>2.x</cmp-version>
   <abstract-schema-name>StudentBean</ abstract-schema-name >
   <cmp-field><field-name>SNO</field-name></cmp-field>
   <cmp-field><field-name>SNAME</field-name></cmp-field>
   ...
   <primkey-field>name</primkey-field>
   <query>
    <query-method>
     <method-name>findBySNAME</method-name>
     <method-params>
      <method-param>String</method-param>
     </method-params>
    </query-method>
    <ejb-ql><![CDATA[FROM StudentTable s Where s.SNAME = ?1]]></ejb-ql>
   </query>
  </entity>
 </enterprise-beans>
 ...
</ejb-jar>
```

[그림 8] CMP 배치 디스크립터

BMP 엔티티 빈 배치 디스크립터에서 <persistence-type> 태그의 내용을 'Bean'에서 'Container'로 변경하고, <resource-ref> 태그에서 지정해주었던 SQL DataSource 대신에 CMP 엔티티 빈을 지원하는 스펙의 버전과 스키마, Finder 메소드 등을 선언 하면 [그림 8]과 같이 CMP 엔티티 빈의 배치 디스크립터가 된다. Dual Persistent 엔티티 빈 패턴을 적용하고, 배치 디스크립터만 변경하면 쉽게 CMP와 BMP 모두로 사용할 수 있기 때문에, 기존의 BMP 엔티티 빈의 재사용성을 높인다.

3.2. DPwR의 유지보수성 향상 방안

엔티티 빈의 재사용성 보다는 성능에 중점을 둔다면 CMP 엔티티 빈보다는 BMP 엔티티 빈으로 구현하는 것이 좋다. 그러나 BMP 엔티티 빈은 구현이 복잡하고, 소스코드가 길어서 코드를 관리하기가 어렵다.

3.2.1 BMP 엔티티 빈 리팩토링

데이터베이스 접근을 위한 구현 부분이 추가되면서 BMP 엔티티 빈의 소스코드는 복잡해지고, 데이터베이스 관련 루틴들이 분산되어 있어 빈의 유지보

수가 어렵다. 소스코드에서 데이터베이스와 관련된 루틴들을 따로 분리하는 리팩토링[10]을 하면 데이터베이스 환경의 변화에 따른 질의문의 변경이 요구될 경우, 데이터베이스 루틴만 수정하면 전체 엔티티 빈에 영향 없이 변경이 가능하므로 유지보수가 쉬워진다. [그림 8]은 데이터베이스 루틴을 호출하는 BMP 엔티티 빈과 데이터베이스 루틴의 소스코드이다.

```
public class StudentBean implements EntityBean{
  public SNO_ejbCreate(int num, String name, int year, String major) throws CreateException{
    try{
      createRow(num, name, year, major);
    }catch(Exception ex){
      throw new EJBException("ejbCreate: " + ex.getMessage());
    }
    return primaryKey;
  }
  ...
}
```

```
public class DBRoutine{
  private Connection con;
  private String dbName = "java:comp/env/jdbc/StudentDB";

  public void createRow(int num, String name, int year, String major) throws SQLException{
    String queryStatement = "INSERT INTO StudentTable (SNO, SNAME, YEAR, MAJOR) values(?,?,?,?)";
    PreparedStatement ps = con.prepareStatement(queryStatement);
    ps.setInt(1, num);
    ps.setString(2, name);
    ps.setInt(3, year);
    ps.setString(4, major);

    ps.executeUpdate();
    ps.close();
  }
}
```

[그림 8] 데이터베이스 루틴 분리

데이터베이스 루틴은 SQL문을 처리하고 있으므로 데이터베이스 환경이 변경될 경우, 데이터베이스 루틴의 질의문만 수정해주면 된다.

4. 실험결과

변경하기 전의 단일 BMP 엔티티 빈으로 구현된 컴포넌트와 DPwR을 적용한 CMP/BMP 엔티티 빈의 성능 각각 측정하여 비교하였다. 실험 방법은 각각의 컴포넌트를 서버에 배치하고, 클라이언트 호출에 대한 작업 처리 시간을 측정하였다. 또한 클라이언트의 수가 증가함에 따른 처리 시간도 측정하였다. DPwR을 적용시킨 BMP 엔티티 빈의 성능이 저하되는 것으로 나타났으나 유지보수에 걸리는 시간은 많이 감소되었다.

5. 결론

구현된 EJB 컴포넌트가 실행될 환경에 최적화되어 구현된 BMP 엔티티 빈은 그 성능과 효율 면에서 CMP 엔티티 빈보다 훨씬 뛰어나지만, 데이터베이스에 대한 구현에 따라 엔티티 빈이 서로 다르기 때문에 빈의 재사용성이나 유연성이 많이 떨어진다. 또한 데이터베이스에 관련된 로직이 복잡해 소스코드의 크기가 길어지고 복잡해진다. 본 논문에서는 재사용성과 유지보수성이 떨어지는 BMP 엔티티 빈의 재사용성을 높이고 유지보수성을 향상시키기 위한 DPwR 방안을 제안하였다. 향후 과제로는, DPwR 자동화 툴을 개발하여 기존의 BMP 엔티티 빈들의 재사용성과 유지보수성을 향상시키도록 하고, 또 다른 향상 방안에 대한 연구가 필요하겠다.

참 고 문 헌

[1] George T. Heineman ,"Component-Based Software Engineering", Addison-Wesley, 2001.

[2] John Cheesman and John Daniels, "UML Components", Addison-Wesley, 2000,

[3] Linda G. DeMichiel, "Enterprise JavaBeans™ Spec. v 2.1", Sun microsystems, 2002.

[4] Rahim Adatia, "Professional EJB", Wrox, 2002.

[5] 박천수 문창수, "EJB & WebLogic", 가메출판사, 2003.

[6] Bill Shannon, "Java™ 2 Platform Enterprise Edition Spec. v1.4", Sun microsystems, 2002.

[7] Richard Monson-Haefel, "Enterprise JavaBeans™", 3rd Edition, O'Reilly, 2001.

[8] Floyd Marinescu, "EJB™ Design Patterns", Wiley, 2002.

[9] Linda G. DeMichiel, "Enterprise JavaBeans™ Spec. v 2.0", Sun microsystems, 2001.

[10]Martin Fowler, "Refactoring", Addison Wesley, 1999.