디자인패턴 기반 EJB Bean 클래스의 MIF와 CF의 측정에 관한 연구

이돈양, 신재준, 송영재
경희대학교 컴퓨터공학과
전화 : 031-201-2946 / 핸드폰 : 011-9017-3709

A Study of MIF & CF Evaluation for EJB Bean Class Based Design Pattern

Dong-Yang Lee, Jae-Joon Shin, Young-Jae Song
Dept. of Computer Engineering, KyungHee University
E-mail : dylee6211@hanmail.net

Abstract

We will take a multitude EJB Design Patterns that you can harness to enhance your EJB Project today. In this paper, we propose the EJB Based Entity Bean DBMS connecting system. Generally, EJB Based Entity Beans are respectively connected by DBMS. Therefore, for the this problems we suggest that Abstract Factory pattern uses DBMS connecting of Entity Beans.
As a result, we evaluate MIF and CF in every class relationship.

Ⅰ. 서론

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides의 Design Pattern(GoF Design Pattern) 기법 중에서 Abstract Factory Pattern을 이용해 부품을 조립하는 방법이 있다. Abstract Factory Pattern에 대해 개념적으로 의미를 정리해보면, 이는 추상적인 공장에서 추상적인 부품을 조립하여 제품을 생성할 수 있도록 하고 있다. 여기서 추상적인 구현은 단지 인터페이스(API)를 하는 상태라고 할 수 있다. 그리고 이 패턴 방법을 적용하여 본 논문에서는 EJB에서 각각의 클래스들을 생성하였다.
일반적으로 쇼핑몰환경에서는 여러개의 엔티티빈을 사용하고 있다. 각각의 엔티티빈들은 데이터베이스와 연동되어 운영되므로 비즈니스 객체의 코드 복잡도를 증가시킬 뿐만 아니라, 코드의 신뢰성과 개발 환경성을 저하 시킬 우려가 있다.
본 논문에서는 이런 문제점들을 개선하기 위해 빈 클래스에서 데이터베이스 연결시 환경설정 및 연결부분을 Abstract Factory Pattern 방법을 적용하여 각 클래스를 Abstract 부분과 Concrete 부분으로 분리하고 인터페이스를 이용하여 조립하였다.
또한, 클래스간의 MIF(Method Inheritance Factor)와 CF(Coupling Factor)를 분석하여 새로운 패턴을 적용한 클래스의 상속성과 결합도를 측정하고자 하였다.
여기서 결과로 얻어진 MIF와 CF 값은 소프트웨어의 상속성과 결합도를 수치로 나타내고 있으며, 이를 이용하여 소프트웨어의 복잡도, 이해도, 유지보수성 및 재사용의 기준으로 사용하고자 하였다.

Ⅱ. 연구배경

2.1 객체지향설계

객체지향설계(OOD:Object Oriented Design)는 객체지향 분석을 사용해서 생성한 분석모형을 소프트웨어 구축의 청사진이 되는 설계모형으로 변환시킨것이다[1]. 여기서 객체지향분석(OOA :Object Oriented Analysis)이란 의뢰인이 정의한 요구사항들의 집합을 만족시킬 수 있게 컴퓨터 소프트웨어를 기술해 주는 일련의 모형을 개발해 주는데 있다.
Fichman과 Kemerer[2]은 객체지향분석 접근법을 구조적분석(Structured Analysis)과 같은 프로세스 지향방법론(Process Oriented Methodology)들보다 급격한 변경을 표현하지만 정보공학(Information Engineering)과 같은 데이터지향(Data Oriented) 방법론들 보다 점진적으로 변경을 표현한다고 결론을 내렸다.
그리고 객체지향설계(OOD:Object Oriented Design)는 모듈성이 많은 다른 수준을 달성하는 설계를 만들어 낸다. 그래서 주요 시스템 컴포넌트는 서브 시스템이라고 부르는 시스템-수준의 "모듈"들로 조직된다. 객체지향설계의 고유 성질은 4개의 중요한 소프트웨어 설계개념인 추상화, 정보은폐, 기능독립, 그리고 모듈성을 구축하는 능력을 가지고 있다.

2.2 디자인 패턴(Design Pattern)

Gamma와 그의 동료들[3]은 패턴을 다음과 같이

설명하고 있다. 당신은 많은 객체 지향 시스템에서 클래스들과 의견교환하는 객체들이 순환되는 패턴들을 찾을 것이다. 이들 패턴들은 특정한 설계문제들을 해결하며 보다 유연하고, 우아하고, 그리고 궁극적으로 재사용 가능한 객체지향 설계를 만들어준다[4]. 이들은 설계자가 이전의 경험에 근거해서 성공적인 설계를 재사용하는데 도움을 준다고 했다.
일반적으로 모든 패턴은 다음의 네가지의 요소로 정의된다.

① 패턴의 이름
② 패턴이 일방적으로 적용되는 문제
③ 디자인 패턴의 특성
④ 적용한 디자인 패턴의 결론

2.3 EJB 디자인 패턴

EJB(Enterprise JavaBeans) 기반 시스템을 설계할 때 중요한 것 중 하나는 퍼포먼스나 유지보수성, 또는 이식성과 같은 사항들을 충족시키기 위한 정확한 아키텍처를 선택하거나 로직을 분할하는 것이다[4]. 잘 만들어진 EJB 프로젝트들은 대부분을 최적의 디자인 패턴을 적용하고 있다. 일반적으로 EJB 레이어 아키텍처 패턴으로는 Session Façade와 Message façade 패턴을 사용한다. 그리고 퍼포먼스의 향상을 위한 방법에서 네트워크를 통한 적은 단위의 메소드 호출을 피하기 위해서 Value Object 패턴을 사용하고 있으며, 한번의 클라이언트 요청에 대한 다수의 네트워크 호출을 피하기 위해서는 Value Object Factory/Value Object Assembler 패턴을 사용하고 있다.
또한 일반적인 J2EE 어플리케이션에서는 영속적인 데이터를 사용하는 경우가 대부분이다. 여기서 영속적인 데이터를 나타내기 위한 엔티티 빈과 같이 영속적이고 공유되고 분산된 컴포넌트를 사용할 수 있다[6]. 엔티티 빈이 영속적인 기억장치를 액세스 할 때 어플리케이션의 엔티티 빈에서 빈 관리 영속성을 쓰도록 고려해야 한다. 이를 관리하기 위한 기법으로는 DAO Factory 패턴이 사용되기도 한다.

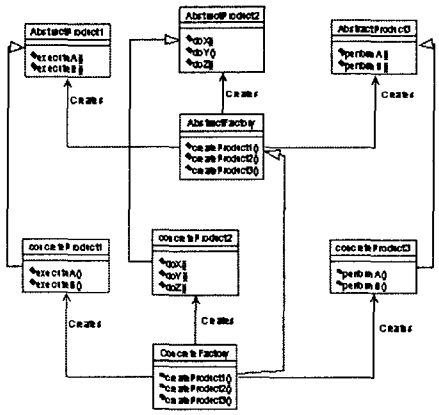
Ⅲ. EJB 디자인 패턴 선정 및 적용

본 논문에서는 EJB기반으로 구현된 쇼핑몰환경에서 엔티티 빈 클래스의 데이터베이스 연결시 환경설정 및 연결부분을 Abstract Factory 패턴 방법을 적용하여 각각의 클래스로 분리하고 조립하였다.
하나의 쇼핑몰 어플리케이션에서 일반적으로 여러개의 엔티티 빈들이 사용되고 있다. 이들에게 공통적으로 적용되고 있는 데이터베이스 엑세스에 대한 코드부분을 관련된 몇 개의 클래스로 분리하므로써 이를 다른 엔티티 빈에서 공용으로 사용할 수 있도록 인터페이스를 생성하여 코드의 신뢰성과 개발의 생산성을 향상시킬 수 있도록 하였다.[6]

3.1 Abstract Factory 패턴

Abstract Factory 패턴은 구체적인 클래스를 지정하지 않고 관련성을 갖는 객체들의 집합을 생성하거나 서로 독립적인 객체들의 집합을 생성할 수 있는 인터페이스를 제공한다.[7]

<참여객체>
① AbstractFactory(WidgetFactory)
- 개념적 제품에 대한 객체를 생성하는 오퍼레이션으로 인터페이스를 정의함
② ConcreteFactory(MotifWidgetFactory)
- 구체적인 제품에 대한 객체를 생성하는 오퍼레이션을 구현한다
③ AbstractProduct(Window, ScrollBar)
- 개념적 제품객체에 대한 인터페이스를 정의함
④ ConcreteProduct(MotifWindow, MotifScrollBar)
- 구체적으로 팩토리가 생성할 객체를 정의하고, AbstractProduct가 정의하고 있는 인터페이스를 구현한다.
⑤ Client
- AbstractFactroy와 AbstractProduct 클래스에 선어된 인터페이스를 사용한다.



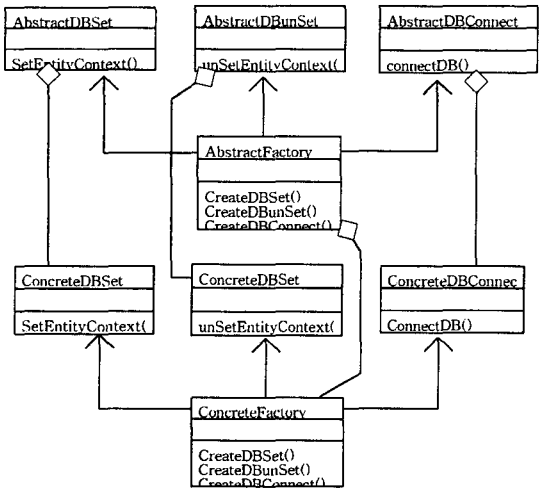
<그림1> Abstract Factroy Pattern

3.2 엔티티 빈 클래스에서 패턴 적용

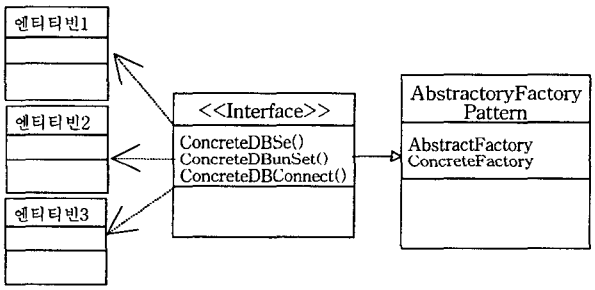
엔터프라이즈 빈 생성시 서버측과 클라이언측에 해당하는 프로그램의 작성이 요구되어 진다. 서버측 프로그램에는 리모트 인터페이스, 홈 인터페이스, 엔티티 빈 클래스, 프라이머리 키 클래스가 요구되어 진다. 여기서 데이터베이스와 관계되는 것은 엔티티 빈 클래스가 해당된다.
아래의 <표1>은 쇼핑몰에서 엔티티빈으로 사용되는 ProductsBean EntityBean Class를 Abstract Factory Pattern을 적용하여 여러개의 해당 클래스로 분리한 것을 나타내고 있다.

<표1> AbstractFactoryPattern의 Class 목록

적용클래스	클래스명
Abstract Product	AbstractFactory
	AbstractDBSet
	AbstractDBunSet
	AbstractDBConnect
Concrete Product	ConcreteFactory
	ConcreteDBSet
	ConcreteDBunSet
	ConcreteDBConnect



<그림2> 데이터베이스 연결 Abstract Factory 패턴

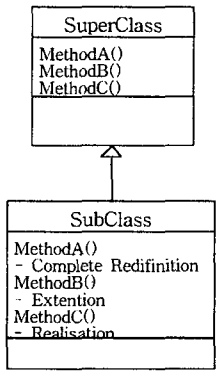


<그림3> 엔티티빈 클래스와 Abstract Factory 패턴의 관계

3.3 MIF(Method Inheritance Factor)와 CF(Coupling Factor) 적용

EJB Bean 생성에서의 클래스 설계 시 Abstract Factory Pattern을 적용한 방법과 기존의 OOD 설계방법을 이용할 수 가 있다. 여기서 Design Pattern이 적용된 방법과 적용되지 않는 방법의 차이점이 존재하는데 본 논문에서는 이를 정량적인 지표로 나타내고자 하였다.

MIF(Method Inheritance Factor)와 CF(Coupling Factor)을 측정하는데는 <그림1>과 Harrison 및 Counsell의 아래의 <식1>,<식2>[8]를 적용하였다. 그리고 그 결과를 이용하여 소프트웨어의 복잡도, 이해도, 유지보수성 및 재사용의 기준으로 사용하고자 하였다.



<그림4>SuperClass와 SubClass와 상속관계

<표2> 분리된 SuperClass와 SubClass

클래스종류	클래스명	메소드수
SuperClass	AbstractFactory	4
	AbstractDBSet	1
	AbstractDBunSet	1
	AbstractDBConnect	1
SubClass	ConcreteFactory	4
	ConcreteDBSet	1
	ConcreteDBunSet	1
	ConcreteDBConnect	1

<식1>

$$MIF = \sum Mi(Ci) / \sum Ma(Ci)$$

$Ma(Ci) = Md(Ci) + Mi(Ci)$
$Ma(Ci)$ = 클래스 Ci에 연관되어 호출될 수 있는 메소드들의 수
$Md(Ci)$ = 클래스 Ci에 선언된 메소드들의 수
$Mi(Ci)$ = 클래스 Ci에 상속될 수 있는 메소드들의 수

<표1>과 같이 본 논문에 적용된 Class의 개수는 8개이다. 이를 위의 <식1>에 적용하면 다음의 결과값이 산출된다.

<적용1>

$$\sum Mi(Ci) = 7$$

$$\sum Md(Ci) = 7 \sum Ma(Ci) = 7$$

$* MIF = 7/14 => 0.5$

여기서 Super Class와 Sub Class에 해당되는 각각의 클래스들은 <그림2>와 같이 대칭적으로 상속되고 있다.
그리고 본 논문에 적용된 Class의 개수는 <표1>에서 8개이다. 이를 위의 <식2>에 적용하면 다음의 결과값이 산출된다.

<식2>

$$CF = \sum i \sum j \; isClient(Ci, Cj)/(TC**2 - TC)$$

$$.TC = Total \;\; Class$$

<적용2>

$$CF = 1/(7^2 - 7)$$
$$= 0.024$$

여기서 CF가 1의 값에 가까우면 결합도가 낮고 반대이면 결합도는 높아진다.[9]
본 논문에서 적용한 디자인 패턴기법에 대한 CF는 비록 높지만 이기법을 적용하지 않았을 경우에 각각의 엔티티 빈에서 데이터베이스를 사용하기 위해서는 각각의 엔티티 빈에서 호출해야 하기 때문에 클래스의 재사용에 대한 효율적인 프로그램작성 및 시스템에 대한 성능향상에 이점을 가질 수가 없다

Ⅳ. 결론 및 향후연구

EJB기반 Application 개발에서는 엔티티빈에 의한 영속적인 데이터의 사용이 많이 이용되고 있다. 일반적으로는 동일한 DBMS를 사용한 데이터베이스 연동이어서 각각의 엔티티빈에서 DB에 Access한다. 이는 시스템에 대한 퍼포먼스 및 유지보수 그리고 이식성등에서 시스템의 저하로 이루어지고 있다.
본 논문에서는 이를 개선하기 위한 방법으로 Abstract Factory Pattern을 이용하여 엔티티빈의 데이터베이스 환경설정 및 연결부분을 Abstract와 Concrete로 분리하고 있다. 그리고 이를 인터페이스를 통하여 각각의 엔티티빈에서 이용가능하도록 하고 있다. 비록 여러개의 클래스로 분리되어 각각이 상속되고 있어 프로그램작성의 어려움과 결합도의 증가가 있지만 비즈니스 객체의 운영과 재사용 측면에서는 많은 이점을 가지고 있다.
마지막으로 향후 추가적인 연구는 서로다른 영속장치에서의 데이터베이스 접근에 대한 연구 및 새로운 디자인 패턴의 적용하고 비교하여 기존에 데이터베이스 연동 및 접근 Tier를 추가하여 EJB 컴포넌트의 재사용성과 확장성을 통한 시스템의 퍼포먼스 향상에 연구가 요구된다.

<참고문헌>

[1] Berard, E.V., Essays on Object-Oriented Software Engineering, Addison-Wesley, 1993
[2] Fichman, R.G. and C.F. Kemerer, "Object-Oriented and Conventional Analysis and Design methodologies", Computer, vol. 25, no. 10, Oct 1992. pp. 22-39.
[3] Gamma, E. et al., Design Patterns, Addison-wesley, 1995
[4] Floyd Marinescu, EJB Design Pattern, Wiley, 2002
[5] Krishnan Subramanian, EJB Design Patterns : Designing EJBs for maximum re-useablity, compactness and flexiblity
[6] 이돈양,이창수,송영재,"EJB Persistence Pattern을 이용한 효과적인 엔티티빈 설계", 한국정보과학회,2002 가을 학술발표논문집(II)
[7] Java 언어로배우는 디자인 패턴입문, 김윤정 역, 2002 영진닷컴
[8] Harrison. R., S.J. Counsell, and R.V.Nithi, "An Evaluation of the MOOD Set of Object-Oriented Software Metrics," IEEE Trans. Software Engineering, vol. SE-24, no. 6, June 1998, pp. 491-496
[9] Dhama, H., "Quantitative Models of Cohesion and Coupling in Software," Journal of Systems and Software, vol. 29, no. 4, April 1995.