EJB Design Pattern 기반의 Bean Class 조립에 대한 모델 설계 및 Testing 에 관한 연구

신재준, 이돈양*, 송영재**
경희대학교 컴퓨터 공학과
전화 : 031-201-2946 / 핸드폰 : 011-9171-6139

A Study on Composition Model Design and Testing for EJB Design Pattern Based Bean Class

Jae-Joon Shin ,Don-Yang Lee* ,Young-Jae Song**
School of Electronics and Information, Kyung Hee Univ.
E-mail : pinkeye@cvs2.khu.ac.kr

Abstract

Today, E-Business develop because, Internet diffuse rapidly. Variety function and excellence performance needs for changeable user's requirement. So, EJB comes Component's addition and modify.For Component adds Function, Existing System modifies all of the source code. In EJB Component System, but, modifies simple source code. But, Today's System don't have cleary way that assembly component and design system. This Paper used design pattern for maintain of the component.
this paper describes EJB Design Pattern using Abstract Factory Pattern. And EJB Component that used Pattern, decribes advantage and disavantage.

I. 서론

인터넷의 급속한 확산으로 인해 전자상거래와 같은 E-Business가 활발해지고 그에 따라 사용자의 요구사항도 더욱 증가하고 있다. 기존의 개발 방법으로는 급변하는 인터넷 환경과 사용자의 요구를 수용하기 어렵다. 따라서 최근 몇 년 사이 주목을 받고 있는 것이 EJB 컴포넌트 기반의 개발이다. EJB 컴포넌트는 기존의 방법보다 변화하는 사용자의 요구를 언어와 플랫폼의 독립성으로 보다 유연성 있게 대처할 수 있다. 기존의 개발 방법은 새로운 기능이나 메소드를 추가하고자 할 때 코드 전체를 수정해야하는 문제가 있는 반면에 EJB 컴포넌트 기반 개발 방법은 코드의 간단한 수정만으로 메소드나 기능을 추가할 수 있다. 다만, 새로운 컴포넌트나 메소드를 추가 할 때의 컴포넌트 조립 방법에 따라 클라이언트의 응답시간이나 트랜잭션 처리시간 , 메소드와 빈들의 처리시간등의 시스템의 전체적인 성능이 변화할 수 있으므로 컴포넌트 조립 방법에 대한 연구가 필요하다. [1]

본 논문에서는 Gof의 디자인 패턴 중 Abstract Factory 패턴을 적용하여 EJB 컴포넌트 조립방법을 제안하였다. 기존의 OOD 방법을 사용하여 조립한 EJB 컴포넌트와 Abstract Factory 패턴을 사용하여 조립한 EJB 컴포넌트간의 성능을 비교 분석한다. 비교 분석 방법으로는 Jakarta 프로젝트에서 제공하는 JAVA 기반 시스템 테스트 툴인 Jmeter를 사용하여 클라이언트의 응답시간, 트랜잭션 처리시간, 메소드와 빈들의 처리시간 등의 시스템 성능을 평가한다.

본 논문의 2절에서는 EJB, 디자인패턴, Jmeter등의 관련연구를 설명하고, 3절에서는 3.1절의 Abstract Factory 패턴을 이용한 컴포넌트 조립, 3.2절의 디자인 패턴을 적용한 EJB 컴포넌트의 조립을 제안한다. 4절

에서는 Jmeter를 이용한 EJB 컴포넌트 성능평가 방법을 기술한다. 마지막으로 5절에서는 본 논문의 결론과 앞으로 연구해야할 과제들을 설명한다.

II. 관련연구

2.1 EJB(Enterprise Java Beans)

EJB란 컴포넌트에 기반한 분산 비즈니스 어플리케이션 개발과 실제 운용을 위한 컴포넌트 아키텍처이다. 즉, 컴포넌트를 개발하기 위한 스팩과 개발된 컴포넌트를 운용, 배치하는 방법을 열거한 것이 EJB라고 볼 수 있다.

EJB는 클라이언트, EJB 서버, 데이터베이스로 구성된 3-Tier 구조로 되어있다. 본 논문에서 설계하고자하는 EJB 컴포넌트는 Middle-Tier인 EJB 컨테이너 안의 EJB 빈 형태로 존재한다. EJB 빈은 홈 인터페이스와 원격 인터페이스를 가지며 클래스가 지원하는 상태 자료의 특성에 따라 세션 빈과 엔티티빈으로 나뉜다[2].

2.2 디자인패턴(Design Pattern)

2.2.1 디자인 패턴

시스템의 설계를 하는 동안 어떤 경우에는 어떤 방법을 적용하고 어떤 문제는 어떤 솔루션으로 해결하는지를 다음 번에 적용하고 재사용 할 수 있도록 정제하고 문서화 해놓은 것이 디자인 패턴(Design Pattern)이다. 디자인 패턴을 이용하면 좋은 설계나 아키텍처를 재사용하기 쉬워진다.

보통 패턴은 4가지 요소를 정의한다[3].

1. **패턴이름** : 패턴에 이름을 부여함으로서 높은 추상화 수준을 제공한다. 설계에 대한 생각을 더욱 쉽게 하도록 도와준다.

2. **문제** : 언제 패턴을 사용하는가를 서술하고 그 배경을 설명한다.

3. **해법** : 설계를 구성하는 요소들과 그 요소들 간의 관계, 책임 그리고 협력관계를 서술한다. 구체적인 설계나 구현을 설명하지는 않는다.

4. **결과** : 패턴을 적용해서 얻는 결과의 장, 단점을 서술한다. 소프트웨어 설계시 비용과 효과를 측정하는데 중요한 요소로 쓰인다.

2.2.2 Abstract Factory 패턴

Abstract Factory 패턴은 상세화 된 서브클래스를 정의하지 않고 관련성이 있거나 독립적인 객체를 이용하여 시스템을 개발하기 위한 인터페이스를 제공하는 패턴이다. 시스템을 독립적으로 만들고자 할 때, 여러 컴포넌트들중 하나를 선택하여 시스템을 설정하고 나중에 다른 컴포넌트로 대체할 수 있을 때, 관련된 컴포넌트를 사용하여 시스템을 구성하고 다른 컴포넌트들이 구성된 시스템의 제약사항을 따라야 할 때 Abstract Factory 패턴을 쓴다. 본 논문에서는 기존의 EJB 시스템에서 컴포넌트를 추가할 때, 패턴을 적용한 시스템과 적용하지 않은 시스템의 성능 차이를 알아보고자 하므로, 위에서 열거한 Abstract Factory 패턴의 마지막 용도인 관련 컴포넌트를 사용하여 시스템을 구성하고 추가되는 다른 컴포넌트들이 구성된 시스템의 제약사항을 따라야하는 경우가 적용된다[3].

2.3 Jmeter

JMeter는 Jakarata 프로젝트에서 제공하는 Java 언어 기반의 테스트 및 성능 평가를 하는 어플리케이션으로서 자바객체, HTTP 서버, Database와 Query 등 동적이나 정적인 자원 모두의 성능을 평가 할 수 있다. 또한 서버의 과부하처리 문제도 테스트 할 수 있는 어플리케이션이다[4].

III. 컴포넌트 조립

3.1 Abstract Factory 패턴에서의 조립

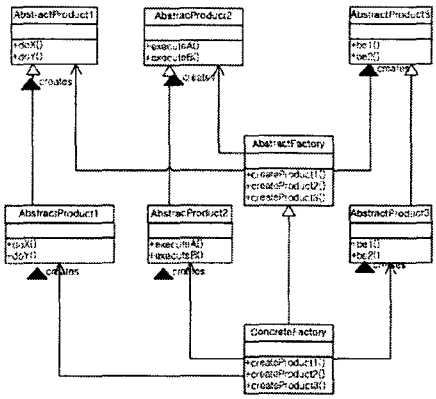
Abstract Factory 패턴을 이용하여 기존의 클래스에 메소드를 추가하고자 한다면 그림.1 의 AbstractFactory, AbstractProduct1, AbstractProduct2, AbstractProduct3 의 하위클래스를 만들어 각각의 추상 메소드만을 만들어 구현한다.

만약, 컴포넌트를 추가하고자 한다면 추가하고자 하는 컴포넌트를 concreteFactory 클래스에 추가하고 추가하고자하는 컴포넌트에 대응하는concreteProduct 클래스를 새로 생성하여 조립한다[5].

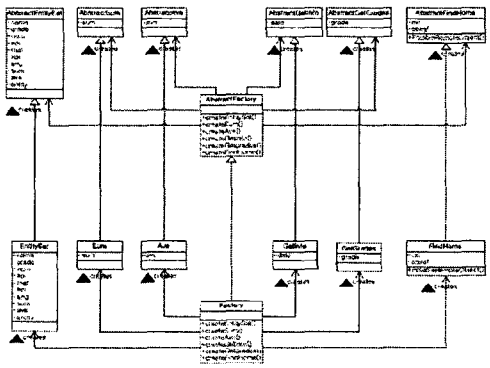
3.2 Abstract Factory 패턴을 적용한 EJB 컴포넌트의 조립

패턴을 적용한 빈 클래스의 조립도 Abstract Factory 패턴의 조립과 마찬가지이다. 예를 들면, 그림 2에서 정보를 받아들이는 Getinfo라는 컴포넌트를 추가하고자 한다면 Factory 클래스와 AbstractFactory 클래스에 createGetinfo 클래스를 새로 생성하여 조립한다. 하나의 클래스가 여러 개의 클래스로 분리되어서 파일의 로딩이나 유지보수가 어려울 것으로 생각되지만 시스템이 대형화되고 복잡해지면 **(그림 2)**와 같은 패턴으로 분리시켜 놓은 것이 더 유지보수가 쉬워진다. EJB 컴포넌트를 추가하거나 제거할 때도 추상적인 인터페이스 선언 클래스만 수정하거나 제거하면 되기 때문이다. 또한, 최근에는 상용컴포넌트(COTS)들이 많이

개발되었기 때문에 사용자들이 직접적으로 구현하는 부분은 많지가 않다.



(그림 1) Abstract Factory 패턴



(그림 2) Abstract Factory 패턴을 적용한 EJB Session Bean

Ⅳ. EJB 컴포넌트 성능평가

4.1 EJB 컴포넌트 성능평가 항목

EJB 컴포넌트의 성능 평가를 위해서는 크게 4가지 항목을 측정해야 한다[6].

첫째로 효율성이다. 접속한 클라이언트의 수에 따른 EJB 컴포넌트의 처리 시간을 측정한다.

둘째로 메소드의 효율성이다. EJB 컴포넌트 안에는 여러 개의 메소드가 존재하는데 메소드 하나당 그 메소드의 기능을 수행하는데 소요되는 시간을 측정한다.

셋째로 처리응답시간이다. 이 방법은 빈을 생성하고 그 빈을 수행하는데 걸리는 총 시간이다.

넷째로 트랜잭션 응답시간이다. 어떤 EJB 컴포넌트는 데이터베이스에 데이터를 저장하거나 읽어 들인다. 이런 경우에 데이터베이스의 설계가 잘되어있다면 트랜잭션 응답시간이 짧을 것이다.

위의 4가지 항목들이 전체적인 시스템의 성능을 결정짓는다. 즉, 각 항목마다 측정하는 요소들이 기준에 부합하는지가 시스템의 성능을 결정짓는다. 위의 4가지 항목들이 측정하는 요소는 응답시간이고 응답시간이 짧다면 시스템의 전체적인 성능이 우수하다고 할 수 있다.

4.2 Jmeter를 이용한 EJB 컴포넌트 성능평가

4.1절에서 언급한 4가지 항목의 성능평가 요소를 평가하기 위해서 본 논문에서는 Jmeter를 사용하였다.

Jmeter에는 5가지 테스트 요소들이 있다[4].

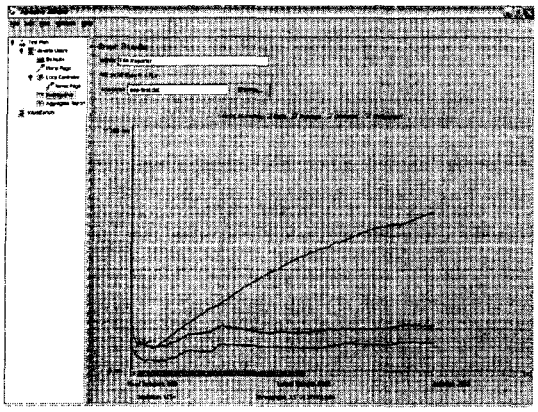
1. **쓰레드 그룹 (Thread Group)** : Jmeter에서 테스트 계획을 하는데 있어서 시작점적인 요소가 된다. 각각의 쓰레드는 다른 쓰레드에 대해 독립적으로 실행되므로 서버 어플리케이션에 대해 동시접속을 시뮬레이션 하는데 사용된다.
2. **컨트롤러 (Controller)** : Jmeter는 2가지 타입의 컨트롤러를 가지고 있는데 테스트할 서버에 요청을 전달하도록 하는 샘플러(Sampler)와 언제 요청을 전달할지를 결정하는 로직 컨트롤러(Logic Controller)가 있다. 이 두 가지 컨트롤러를 사용하여 4.1절에서 언급한 4가지 항목의 성능평가 요소들을 평가 할 수 있다.
3. **리스너 (Listener)** : Jmeter에서 테스트된 결과들을 사용자에게 제공할 수 있는 수단을 제공한다. 본 논문에서는 그림 3, 4와 같이 Graph 리스너와 Aggregate 리스너를 사용하여 테스트 결과들을 그래프와 수치로 보여준다.
4. **타이머 (Timer)** : 타이머는 계속되는 서버로의 요청들을 지연(delay)을 주어서 클라이언트들의 독립성을 보장해준다. 타이머의 지연(delay)를 줄이고 많은 양의 요청을 서버에 보낸다면 서버의 과부하 테스트를 할 수 있다.
5. **어서션 (Assertions)** : 어서션은 특정 요청에 대한 결과를 사용자에게 제공한다. 어서션을 사용하여 특정 빈에 대한 테스트를 할 수 있다.

4.3 결과 비교 및 분석

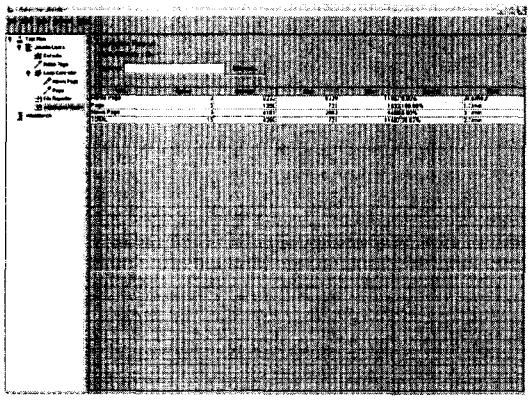
테스트 결과는 Graph 리스너와 Aggregate 리스너를 사용하여 수치적으로 비교, 분석한다.

그림 3과 그림 4는 Jmeter를 이용하여 Jakarta 사이트의 특정 페이지를 테스트 한 것이다.

먼저 그림 3의 Graph Listener를 살펴보면 녹색 선으로 나타나는 그래프가 테스트 샘플 개수당 처리 시간이다. 즉, 사용자의 요구 증가에 대한 서버의 처리속도이다. 그래프가 급격하게 증가하면 처리 속도가 급격



(그림 3) Graph Listener



(그림 4) Aggregate Listener

하게 증가하는 것이고 그래프가 서서히 증가하면 처리 속도가 서서히 증가하는 것이다.

청색 선으로 나타나는 그래프는 사용자의 요구에 대한 평균적인 응답시간이다. 적색 선으로 나타나는 그래프는 사용자의 요구 편차에 대한 응답시간이다. 따라서 이 그래프를 통하여 시스템의 전체적인 성능을 파악할 수 있다.

그림 4의 Aggregate Listener를 살펴보면 7개의 항목이 있는데 처음 항목인 'URL'은 특성 페이지를 뜻한다. 'count'는 테스트 셋의 개수 즉, 클라이언트 수를 뜻한다. 'min'은 테스트 중 가장 적은 요구의 개수 'max'는 가장 많은 요구의 개수를 뜻하며 'average'는 테스트 중 발생한 요구들의 평균을 뜻한다. 마지막으로 에러 비율과 분(minute)당 처리 속도도 'Error%'와 'Rate'에서 표시하고 있다. 따라서 이 항목들을 이용하여 클라이언트 수에 따른 서버의 요구처리 속도와 에러 비율을 알 수 가 있다.

V. 결론 및 향후 연구방향

본 논문 3.1절과 3.2절에서는 EJB 컴포넌트의 조립방법을 Abstract Factory 패턴을 사용하여 설계하여 효율적인 조립방법을 제시하였다. 그리고 4절에서는 기존의 방법과 패턴을 적용한 방법을 Jmeter로 비교하여 시스템의 효율성이나 처리응답시간, 트랜잭션 응답시간 등의 차이를 분석하는 방법을 제시하였다. 본 논문에서 제시한 조립방법과 성능평가방법을 증명하기 위하여 EJB 컴포넌트의 조립과 성능평가에 대한 실제 구현이 필요하다.

본 논문에서는 Abstract Factory 패턴 하나만을 가지고 설계, 비교, 분석하였으므로 EJB 컴포넌트에 디자인 패턴이 미치는 영향을 파악하기 쉽지 않다. 따라서, Bridge 패턴, Session Facade 패턴, Message Facade 패턴 등도 EJB 컴포넌트에 적용하여 시스템의 성능에 미치는 영향을 파악해야 한다. 또한, EJB 컴포넌트의 재사용성이나 확장성을 위해 MIF(Method Inheritance Factor)와 CF(Coupling Factor)를 이용하여 클래스의 상속성(Inheritance)과 결합도(Coupling)를 측정하여 디자인 패턴이 EJB 컴포넌트의 재사용성과 확장성에 미치는 영향에 대해서도 연구가 필요할 것이다.

참고문헌(또는 Reference)

[1]이돈양, 송영재 "EJB 디자인 패턴기반 Session Facade와 Message Facade의 합성 모델", 한국 정보처리학회 소프트웨어공학연구회지, 제5권 1호, 2002.3

[2]Enterprise JavaBeansTutorial, Sun Microsystems Inc.,"http://developer.java.sun.com/developer/onlineTraining/Beans/EJBTutorial"

[3]Eric Gamma, Richard Helm, Ralph Johnson, John Vlissides, "Design Pattern : Element of Reusable Object-Oriented Software", Addison-Wesley, 1995

[4]Jmeter , Apache Software Foundation, "http://jakarta.apache.org/jmeter"

[5]Hiroshi Yuki, "Java 언어로 배우는 디자인 패턴 입문", 영진닷컴, 2002

[6]Xiaoying Bai, "Distributed End-to-End Testing Management" , 5th IEEE International Enterprise Distributed Object Computing Conference, 2001.9