2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

엔터프라이즈 환경에서 효율적인 EJB 기반 컴포넌트 개발을 위한 디자인 패턴의 적용

°최성만*, 김정옥*, 이정열**, 유철중*, 장옥배*
*전북대학교 컴퓨터과학과
**정인대학 사무자동화과
sm3099@cs.chonbuk.ac.kr
{kjo3852, ljy8383}@hanmail.net
{cjyoo, okjang}@moak.chonbuk.ac.kr

Application of Design Pattern for Efficient EJB-Based Component Development in Enterprise Environment

°Seong-Man Choi*, Jeong-Ok Kim*, Jeong-Yeal Lee**,
Cheol-Jung Yoo*, Ok-Bae Chang*
*Dept. of Computer Science, Chonbuk National University
**Dept. of Office Automation, Chongin College

요약

엔터프라이즈 환경에서 효율적인 EJB 컴포넌트 개발을 위하여 전표등록 패키지를 선정하여 설계시에 Factory Method 패턴과 Facade 패턴을 적용해보았다. 이 디자인 패턴의 적용에 EJB 기반 컴포넌트의 무상태 세션 빈과 상태유지 세션 빈 및 엔티티 빈을 사용하였다. 그 결과 설계시간과 재사용성 및 시스템의 안정성 측면에서 향상가능성을 확인할 수 있었다.

1. 서론

컴포넌트 기반 소프트웨어 시스템 개발은 소프트웨어 업계에서 나타난 표준적인 컴포넌트 기반 구조 기술의 특성과 소프트웨어 생산성 모두를 주목할만하게 향상시키기 위한 솔루션으로 EJB, CORBA, ActiveX, JavaBeans와 같은 컴포넌트 기술의 공통적인 목적과 함께 오늘날 이미 유용하게 이용하고 있다[1]. 본 논문에서는 디자인 패턴의 적용에 의한 재사용성과 비종속성 EJB 컴포넌트의 개발을 위한 설계 및 구현에 대해서 논한다. 먼저 2장에서는 EJB 컴포넌트 개발을 위한 디자인 패턴에 대해서 알아보고, 3장에서는 디자인 패턴을 적용한 EJB 컴포넌트 설계에 대해서 알아본다. 또한 4장에서는 디자인 패턴의 적용 사례 및 적용 효과에 대해서 설명하고, 마지막 5장에서는 결론 및 향후 연구과제를 제시한다.

2. EJB 컴포넌트 개발을 위한 디자인 패턴

2.1 디자인 패턴의 특성

디자인 패턴이란 소프트웨어를 설계하는데 있어서 되풀이되는 경험적인 패턴들을 등록하여 특정한 문제에 대한 해결방안을 제시하고자 하는 설계의 한 방법이다[2]. 즉, 디자인 패턴은 미래의 비슷한 상황에서 다시 적용될 수 있는 과거에 잘 정의된 설계에 대한 정보를 기록하는 것이다[3]. 디자인 패턴을 EJB 컴포넌트 설계에 적용하면 설계 범위를 확장할 수 있고, 설계에 대한 확신을 가질 수 있으며, 설계의 재사용성을 높여주는 효과를 준다. 또한, 설계 시간의 단축 및 개발자간의 의사소통 시간을 줄일 수 있다는 장점과 효율적이고 유용성이 좋은 디자인 패턴을 이용함으로써 시스템의 안정성과 성능을 높일 수 있는 장점을 가진다. 이러한 디자인 패턴을 적용

시킬 때에는 세 가지 원칙이 적용된다. 첫째, 언제나 인터페이스를 먼저 생각해보며 둘째, 상속(inheritance)보다는 위임(delegation)을 주로 이용하고 셋째, 항상 결합도(coupling)를 최소화하는데 역점을 두어야 한다[4, 5]. Gamma는 디자인 패턴을 생성, 구조, 행위 패턴으로 분류하였으며, 이를 다시 패턴이 하는 일이 무엇인지를 반영하는 목적과 패턴이 객체 또는 클래스에 처음으로 적용될 때의 명확한 범위에 따라 구분을 해보면 [표 1]과 같이 표현할 수 있다[2].

[표 1] 디자인 패턴의 유형

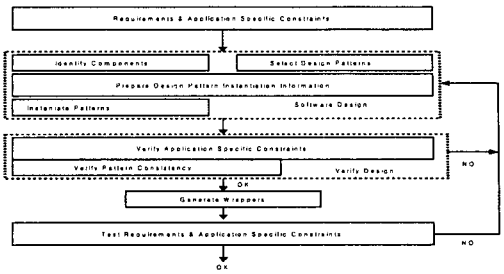
목적 / 범위	생성 패턴	구조 패턴	행위 패턴
클래스	Factory Method	Adapter(class)	Interpreter Template Method
객체	Abstract Factory Builder Prototype Singleton	Adapter(object) Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Observer State Strategy Visitor

2.2 디자인 패턴을 이용한 EJB 컴포넌트화 단계

EJB는 엔터프라이즈 애플리케이션을 위해 Java 진영에서 내놓은 새로운 컴포넌트 모델로서 차세대 컴포넌트 기술 표준으로 부상하고 있으며, 빠르게 변화하는 인터넷 환경에서 애플리케이션을 안정적으로 구축할 수 있도록 해주는 서버측 컴포넌트 표준 모델이다[6]. EJB를 기반으로 애플리케이션을 구축한다면 기존에 개발자들이 부담해야 했던 보안, 자원 풀링, 지속성, 동시성, 트랜잭션 등의 복잡한 내용을 처리하는 서비스를 애플리케이션 서버가 제공해주기 때문에 개발자들은 비즈니스 로직에만 전념할 수 있도록 해주는 장점이 있다[7]. 이와 같은 이유로 인하여 기존의 시스템이나 EJB 기반이 아닌 시스템들도 EJB로 구현하기 위한 시스템으로의 전환을 시도하고 있는 실정이다. 이러한 EJB를 개발하기 위한 언어로 Java가 선호되는 이유로는 플랫폼에 독립적이고 분산 객체지향 언어이기 때문이다. 다음은 디자인 패턴을 이용하여 컴포넌트화 단계를 보여주는 그림이다[8].

3. 디자인 패턴을 적용한 EJB 컴포넌트 설계

본 장에서는 디자인 패턴을 적용하기 위하여 전표

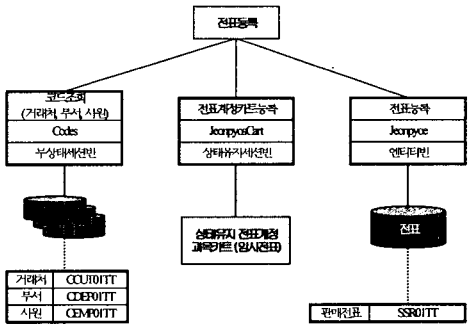


[그림 1] 디자인 패턴을 이용한 컴포넌트화 단계

등록 패키지를 선정하여 설계를 위한 시나리오를 작성하고 전표등록 패키지의 구성도를 작성한다. 또한, 엔터프라이즈 환경에서 EJB 기반 컴포넌트를 개발하기 위해서 디자인 패턴을 적용한 MVC 클래스 모듈간의 참조 관계를 설계한다.

3.1 전표등록 패키지의 설계

(1) 사용자가 전표등록을 위하여 로그인 한다.
(2) 전표생성을 위한 신규등록 버튼을 클릭 한다.
(3) 거래처코드, 부서코드, 사원코드를 세션 빈을 이용하여 정보의 서비스를 제공한다.
(4) 계정과목별 전표의 세부 항목자료를 입력하고 등록버튼을 클릭 한다.
(5) 계정과목 임시전표 카트의 상태유지 세션 빈에 등록한다.
(6) (5)의 계정과목 수만큼 반복처리에 의한 전표를 작성한다.
(7) 전표 작성이 완료되면 전표실적 엔티티 빈을 호출하여 임시 전표를 전표테이블에 등록한다.
(8) 전표등록 업무를 종료한다.



[그림 2] 전표등록 패키지의 구성

[그림 2]에서 보여주듯이 전표등록 패키지 시스템의 처리과정은 크게 세 가지의 기능으로 구분할 수 있

다. 이 세 가지 빈의 유형을 표현하면 첫째, 코드조회 무상태 세션 빈은 사용자가 요구하는 코드 정보를 DB에서 읽어들여 사용자의 입력항목을 제공함으로써 사용자 정보서비스를 제공한다. 둘째, 전표계정카트등록의 상태유지 세션 빈은 계정 과목별 전표를 등록하기 위하여 임시로 등록하여 보관하고 수정, 삭제를 제공하는 임시전표 정보를 상태유지 세션 빈 형태로 관리하고 정보를 제공한다. 셋째, 전표등록 엔티티 빈은 임시전표 카트에 의한 전표 작성이 완료되면 그 결과를 전표테이블에 저장하는 기능을 제공함으로써 영속성 엔티티 빈으로 설계하였다.

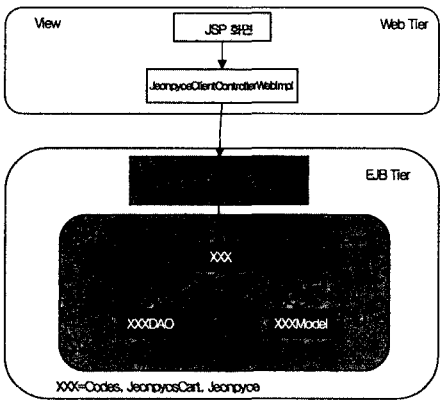
3.2 MVC 클래스 모듈간의 참조 관계

(1) Codes 모듈 : 전표 등록을 위한 기준코드(거래처코드, 부서코드, 사원코드, …)의 정보를 제공하는 무상태 세션 빈

(2) JeonpyosCart 모듈 : 전표의 계정과목별 등록 내용을 임시 저장하는 상태유지 세션 빈

(3) Jeonpyoe 모듈 : 전표생성 결과를 전표실적 테이블에 등록하기 위한 엔티티 빈

(4) Control 모듈 : 전표등록, 코드정보, 계정과목 등록 카트, 전표등록 빈을 종합적으로 제어하는 엔터프라이즈 세션 빈



[그림 3] MVC 클래스 모듈간의 참조 관계

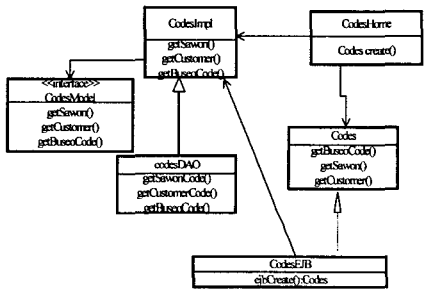
4. 디자인 패턴의 적용 사례 및 적용 효과

4.1 적용 사례

전표등록은 EJB 기반 컴포넌트 시스템으로 설계시에 Factory Method 패턴을 적용해보며, 전표등록 처리에서는 필요한 모든 빈을 통합적으로 관리하기 위해서 Facade 패턴을 적용한다.

(1) Factory Method 패턴을 적용한 코드조회 세션 빈

전표등록 EJB 컴포넌트 시스템의 설계에 있어서 엔터프라이즈 빈 설계시 Factory Method 패턴을 적용하면 오브젝트를 생성할 때 하위 클래스들이 어떤 클래스를 개체화 할 건지를 결정할 수 있도록 인터페이스를 정의한다. 그런 후에 Factory Method 패턴은 클래스의 개체화를 서브클래스에게 위임하도록 함으로써 불필요한 코딩작업을 줄여주는 효과가 있다. [그림 4]는 전표처리에서 코드조회 무상태 세션 빈을 Factory Method 패턴에 적용하고 구현한 결과를 보여주고 있다.

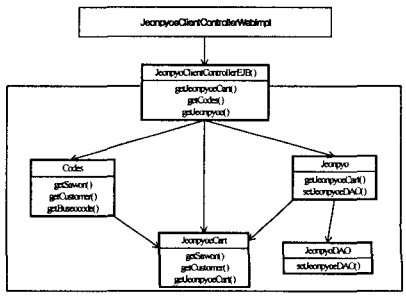


[그림 4] Factory Method 패턴의 적용

(2) Facade 패턴을 적용한 컨트롤 세션 빈

본 시스템에서는 전표등록 처리에서 필요한 모든 빈을 통합적으로 관리하도록 하기 위하여 JeonpyoClientController 엔터프라이즈 세션 빈을 설계함으로써 모든 빈의 호출이 JeonpyoeClientController 엔터프라이즈 빈을 통하여 이루어지도록 함으로써 유지보수 및 관리가 용이하도록 하였다.

전표등록 처리에서 적용된 Facade 패턴은 하부구조의 인터페이스 집합에 대한 통합된 인터페이스를 제공해야 할 때 사용하므로 클라이언트와 내부 구현



[그림 5] Facade 패턴의 적용

클래스간의 결합도를 최소화시켜주는 장점이 있다.

4.2 적용 효과

전표 등록 패키지를 선정하여 엔터프라이즈 환경의 EJB 기반 컴포넌트 개발을 위한 디자인 패턴으로 적용해보았다. 적용 결과 효율적인 EJB 컴포넌트 개발을 위한 효과는 다음과 같다.

(1) EJB 컴포넌트와 같은 분산 객체 컴포넌트 아키텍처는 방화벽(firewall) 안쪽에 중요한 비즈니스 로직을 유지할 수 있도록 정보의 분리를 지원한다.

(2) 공유 데이터를 액세스하는 여러 클라이언트를 위해 EJB 컴포넌트는 데이터에 대한 액세스를 제어해 주뿐만 아니라 개발자으로 트랜잭션과 데이터베이스를 관리해준다.

(3) EJB 컴포넌트는 자동적으로 공유 데이터를 읽고 수정하기 위한 데이터베이스 로킹, 동시성, 데이터 무결성 등을 처리함으로써 데이터베이스 제어 로직을 작성하는데 드는 노력을 감소시켜줌으로써 전체 프로그래밍 노력을 줄여준다.

(4) 트랜잭션 능력을 가진 다른 데이터 자원을 애플리케이션 서버(Websphere, WebLogic 등)에서 담당한다.

(5) EJB 컴포넌트는 HTML 문서, Servlet, JSP 파일, 클라이언트 로그인에 대한 보안이 완전하게 통합된 메소드 레벨의 보안을 허용함으로써 메소드 실행을 허용 및 거절할 수 있는 사용자 및 사용자 그룹을 생성한다.

(6) EJB 컴포넌트 아키텍처는 플랫폼, 벤더, 애플리케이션 서버 구현 등의 독립성을 실현함으로써 표준, 이식성, 컴포넌트 기반 아키텍처로 변환을 지원한다.

(7) 같은 비즈니스 로직을 공유하는 여러 클라이언트 유형을 가진다.

5. 결론 및 향후 연구과제

본 논문에서는 엔터프라이즈 환경에서 효율적인 EJB 기반 컴포넌트 개발을 위해서 전표등록 패키지의 설계에 Factory Method 패턴과 Facade 패턴을 적용해보았다. EJB 기반 컴포넌트 개발을 위해서 디자인 패턴을 적용해 본 결과 설계 시간과 재사용성 및 시스템의 안정성 측면에서 향상가능성을 확인할 수 있었다. 향후 연구과제로는 EJB 기반 컴포넌트 시스템의 설계 및 구현을 개발자들이 쉽게 이해하고 적용할 수 있는 지침을 만들고, EJB 컴포넌트 설계에 적합한 디자인 패턴의 개발이 필요하며, 디자인 패턴에 적용할 수 없는 경우에 발생하는 문제에 대해서 지속적인 연구가 필요할 것이다. 또한 EJB 기반 컴포넌트 개발에 다른 디자인 패턴을 적용시켜보고, 설계 시간과 재사용성 및 시스템의 안정성 측면에서의 향상가능성에 대한 구체적인 측정 연구가 필요하다.

참고문헌

[1] Gilda Pour, "Enterprise JavaBeans, JavaBeans & XML Expanding the Possibilities for Web-Based Enterprise Application Development", in Proceedings of Technology of Object-Oriented Languages and Systems, pp. 282-291, 1999.

[2] Erich Gamma, Richard Helm, Raplh Johnson, John Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison Wesley Longman, Inc., 1995.

[3] 김운용, 최영근, "패턴적용을 통한 기능분할 GUI 컴포넌트 구조 모델", 한국정보처리학회 추계학술발표논문집, 제7권 2호, pp. 547-550, 2000.

[4] Schmidt, Douglas, "Pattern-Oriented Software Architecture Vol.2 H/C", John Wiley & Sons, Inc., 2000.

[5] Cooper, James William, "Java Design Patterns : A Tutorial", Addison Wesley Longman, Inc., 2000.

[6] Ed Roman, "Mastering Enterprise JavaBeans™ and the Java™ 2 Platform, Enterprise Edition", John Wiley & Sons, Inc., 1999.

[7] http://www-106.ibm.com/

[8] Stephen S. Yau, Ning Dong, "Integration in Component-Based Software Development Using Design Patterns", in Proceedings of COMPSAC, pp. 369-374, 2000.