碩士學位 論文

오픈소스 프레임워크를 이용한 소프트웨어 개발 사례연구

A Case Study on Software Development using OpenSource Framework

2007年 12月

崇實大學校 情報科學大學院

소프트웨어공학과

전 혜 영

碩士學位 論文

오픈소스 프레임워크를 이용한 소프트웨어 개발 사례연구

A Case Study on Software Development using OpenSource Framework

2007年 12月

崇實大學校 情報科學大學院

소프트웨어공학과

전 혜 영

碩士學位 論文

오픈소스 프레임워크를 이용한 소프트웨어 개발 사례연구

A Case Study on Software Development using OpenSource Framework

指導敎授 류 성 열

이 論文을 碩士學位 論文으로 提出함

2007年 12月

崇實大學校 情報科學大學院

소프트웨어공학과

전 혜 영

전혜영의 碩士學位論文을 認准함

審査委員長 _____ 印

審 査 委 員 _____ 印

審 査 委 員 _____ 印

2007年 12月

崇實大學校 情報科學大學院

感謝의 글

오픈소스라는 주제를 가지고 1년반을 헤메고 있는데도 끝까지 놓지 않고 지도해 주신 류성열 교수님께 진심으로 감사드립니다. 또한 많이 부족한 논문이지만 심사해주신 이남용 교수님, 최용락 교수님 감사드립니다. 논문이 나올 수 있도록 개인시간 내서 도와주신 김상일 선배님 감사드립니다. 정보과학대학원의 임영환 교수님, 문영성 교수님, 정기원 교수님, 신용태 교수님, 정기철 교수님, 윤창섭 교수님, 이종호 교수님 감사드립니다.

여러 가지 어려움이 있었지만 같이 공부하고 힘이 되어준 원우들 덕분에 끝까지 마무리가 잘 되지 않았나 하는 생각이듭니다. 김경서 고문님, 고광섭 회장님, 송황희 부회장님, 오금석 총무님, 문용석님 등 35기 원우님들 모두 고생하고 수고 하셨습니다. 항상 격려와 힘을 보내주시는 김성진 교수님, 이순연 선생님, 성윤정 선생님 감사드립니다.

합격 통지서를 받고서도 첫 아이를 임신하고 5학기동안 잘할 수 있을까 하는 마음에 등록금 고지서를 가지고 고민했던 때가 엊그제 같은데 벌써 시간이 지나 졸업을 하게 되어 마음이 너무 벅차오르네요. 뱃속에서부터 신경 많이 못써주고 많은 시간 같이 못했지만 건강하게 자라고 있는 아들 민성아 사랑한다. 건강한 마음과 건강한 육체를 주시고 스스로 할 수 있는 힘을 보여주신 엄마 사랑합니다. 항상 옆에서 응원하고 힘이 되어준 남편 도형, 경식오빠, 정순이모, 호순이모, 인자이모, 가족들 너무너무 사랑하고 감사합니다.

목 차

그림 목차

표 목차

國文抄錄

오픈소스 프레임워크를 이용한 소프트웨어 개발 사례연구

소프트웨어공학과 전혜영
지도교수 류성열

상용 프레임워크를 이용하여 소프트웨어를 개발하면 초기엔 기술지원을 받을 수 있는 장점은 있지만 많은 구매 비용이 필요하고 특정 벤더 플랫폼에 종속되기 때문에 이후 유지보수 등의 문제가 발생할 수 있다.

본 연구에서는 도서 쇼핑몰 구축을 상용 프레임워크와 오픈소스 프레임워크를 이용하여 개발해 보았다. 도서 쇼핑몰을 구축하기 위해 개발 환경과 아키텍처를 설계하고 설계된 아키텍처에서 회원가입, 도서구매, 구매확인 컴포넌트를 상세 설계하여 구현하였다. 개발 결과 오픈소스 프레임워크를 이용한 개발이 상용 프레임워크를 이용한 개발 절차보다 훨씬 간단하고 효율적이라는 것을 확인할 수 있었다. 또 오픈소스 프레임워크를 이용한 개발 절차와 지침을 작성할 수 있었다.

ABSTRACT

A Case Study on Software Development using OpenSource Framework

JEON, HYE-YOUNG
Department of Software Engineering
Graduate School of Information Science
Soongsil University

By using commercial software development framework, you can initially receive many technical support. But a lot of cost will be needed to purchase software and because certain vendors are dependent on the platform, it could cause maintenance problems.

In this study, I developed the book store system to use commercial framework and OpenSource framework. I designed a developmental environment and architecture for implementing book store system, and implemented components of member subscription, book purchase, purchase confirmation in designed architecture. As a result, it could be knowned that the development is much simpler and more efficient when using OpenSource framework instead of commercial framework and I could write development processes and guidelines by using the OpenSource framework.

제 1 장 서 론

1.1 연구배경

빠르게 발전하는 IT기술 및 개발환경 변화로 이를 대응해서 기술 투자를 해야 하는 기업과 개발환경을 이해해야 하는 개발자들은 큰 부담일 수 밖에 없다. 그래서 각 기업이 모두 기술 투자를 하는 것이 아니라 이미 만들어진 상용 소프트웨어나 개발 툴을 구입하고 이를 기반으로 애플리케이션을 개발한다.

상용 소프트웨어를 이용하여 시스템을 개발하면 유용한 애플리케이션을 빨리 작성할 수 있다는 장점은 있지만 구매에 따른 비용이 증가할 뿐만 아니라 사용하지 않는 기능까지 제공하기 때문에 불필요한 비용이 추가되어 개발에 많은 부담이 된다. 또한 특정 벤더 플랫폼에 종속되기 때문에 이후 유지보수 등의 문제가 발생할 수 있다.

이에 대한 대안으로 등장한 것이 오픈소스 프레임워크이다. 그러나 오픈소스 프레임워크를 기반으로 소프트웨어를 개발하기 위해서는 관련된 수많은 클래스와 환경정보를 관리하는 구조를 이해해야 하는데 이러한 내용을 이해하기 위한 기술 문서가 상용 소프트웨어에 비해 턱없이 부족하다. 특히 오픈소스 소프트웨어의 범람으로 선택을 어렵게 하고 이를 기반으로 개발한 사례 연구 부족으로 실제 개발에 사용하기엔 매우 어려움이 많다.

1.2 연구목적

본 연구에서는 상용 프레임워크를 기반으로 개발하는 과정을 오픈소스 프레임워크를 이용하여 개발하는 사례를 보여줌으로 오픈소스 프레임워크를 사용한 개발 가능성을 확인하고 오픈소스 프레임워크를 이용한 구체적인 개발 절차 및 지침을 작성한다.

1.3 연구범위 및 방법

본 연구는 오픈소스 소프트웨어에 대한 이해를 하고 자바를 이용한 소프트웨어를 개발하기 위해 J2EE 아키텍처에 대한 연구를 한다. 또한 J2EE 아키텍처의 계층에 배치할 수 있는 오픈소스 프레임워크에 대한 연구를 기반으로 각 계층에 사용할 오픈소스 프레임워크를 선정한다. 선정된 프레임워크를 이용하여 소프트웨어 아키텍처를 설계하고 상용 프레임워크와 오픈소스 프레임워크를 기반으로 응용 소프트웨어를 작성한다.

1.4 기대효과

오픈소스 프레임워크를 기반으로 애플리케이션을 작성하면 다음과 같은 기대효과가 있다.

- 고객 측면에서는 유지보수성이 향상됨으로써 개발 비용의 몇 배에 달하는 유지보수 비용을 줄일 수 있다.
- 개발사 측면에서는 소프트웨어를 개발시 프레임워크의 사용은 개발 생산성 향상으로 프로젝트의 성공률을 높일 수 있다. 또 무료 소프트웨어를 사용함으로 개발 비용을 감소시켜 개발사에 더 많은 이익을 줄 수 있다.
- 개발자 측면에서는 개발생산성이 좋아짐으로써 개발자의 개발의욕 증대 및 기술력 향상을 가져다 줄 수 있다.

제 2 장 관련연구

2.1 오픈소스 프레임워크

오픈소스 소프트웨어(OpenSource Software) 프로젝트에 관한 정보는 아파치 그룹(Apache Group), 소스포지(SourceForge), KLDP(http://kldp.net) 등에서 확인할 수 있다. 오픈소스 소프트웨어 프로젝트는 서버, 운영체제, 컴포넌트, 프레임워크, IDE(Integrated Development Environment) 등 여러 분야가 있음에도 최근 활성화되는 프로젝트를 살펴보면 JBoss, 하이버네이트(Hibernate), 스프링(Spring), Ant, Beehive, iBATIS, Jackrabbit, 스트럿츠(Struts), Tapestry 등과 자바 개발에 유용한 소프트웨어임을 확인해 볼 수 있다. 대표적인 프로젝트 스트럿츠는 MVC Model2 디자인 패턴기반으로 개발할 수 있는 환경을 제공하는 프레임워크로 가장 활성화되어 있고 최근에는 ORM(Object Relational Mapping)에 관한 프레임워크도 많이 사용되고 있다.

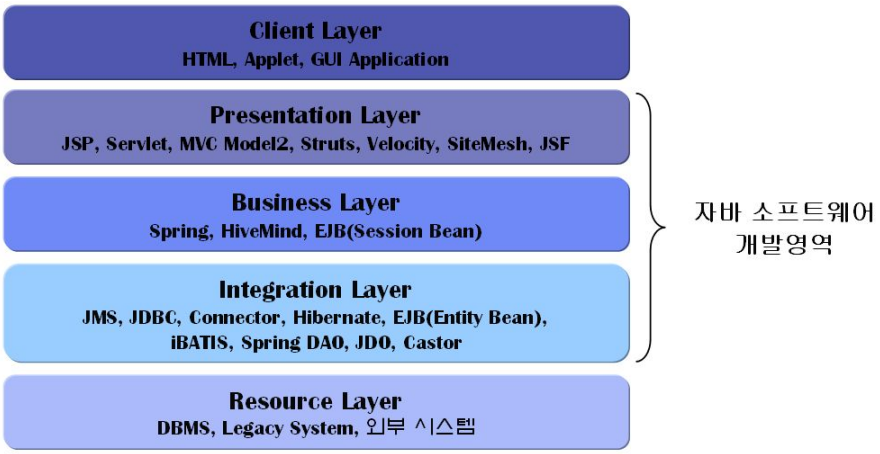
오픈소스 프레임워크(OpenSource Framework)를 활용하면 비용 측면 뿐 아니라 품질과 기술혁신 측면에서도 많은 장점을 가지고 있는데 [표 2-1]에서 오픈소스 프레임워크를 활용에 따른 장점을 나열하였다 [1][2][3].

[표 2-1] 오픈소스 프레임워크의 활용에 따른 장점

구분	내용
융통성	비용에 제약을 받지 않아 테스트후 선택의 폭이 다양하다.
기술지원	신속한 문제해결 및 성능개선 프로세스, 기술의 공동 습득이 가능
기술혁신	무료임으로 기술을 실험적으로 사용할 수 있다.
재활용	소스코드 접근이 가능하고 재활용이 증가한다.
품질	검증된 프레임워크를 사용함으로 개발이 빨라지고 유연해진다.
표준	표준에 충실하고 상호 운영이 뛰어나다.

2.2 J2EE 아키텍처와 오픈소스 프레임워크[4][5][6]

상용 프레임워크나 오픈소스 프레임워크는 J2EE 아키텍처(Architecture)를 기반으로 역할에 따라 계층별로 구분하여 관리된다. 이는 같은 프레임워크라 하더라도 어떤 계층을 기반으로 개발하느냐에 따라 프레임워크의 역할이나 특징이 달라질 수 있기 때문이다.



[그림 2-1] J2EE 아키텍처 Layer

[그림 2-1]을 보면 J2EE 아키텍처에서는 5가지 계층으로 나눠 관리한다. 클라이언트 계층(Client Layer)과 리소스 계층(Resource Layer)은 실제 자바 소스로 개발하기 보다는 이미 만들어진 사용 환경 및 시스템 환경이고 자바 소프트웨어를 이용하여 개발할 수 있는 영역은 프리젠테이션 계층(Presentation Layer), 비즈니스 계층(Business Layer), 인테그레이션 계층(Integration Layer)이다. 이때 인테그레이션 계층은 외부 시스템이나 장비를 접근하거나 처리하는 계층이다.

클라이언트 계층은 사용자의 요청을 받아서 정보시스템에 전달하고 정보시스템에서 처리한 결과인 응답을 사용자에게 전달하는 역할을 한다. 사용자의

환경은 웹, 애플리케이션, 모바일 등으로 구분하는데 웹은 Explorer, Netscape, Mozilla, FireFox 등 웹 브라우저가 기본 환경을 제공하고 모바일은 다양한 환경에 따라 별도의 UI를 작성하여야 한다. 애플리케이션은 비주얼 베이직, , AWT, Swing 등 GUI 툴을 이용하여 사용자가 쉽게 사용활 수 있는 UI를 작성한다.

프리젠테이션 계층은 뷰와 제어 계층으로 구분하는데 주로 뷰는 사용자의 요청을 전송하거나 전송받는 역할을 하고 사용할 수 있는 프레임워크는 JSP, Taglib/JSTL, JSF, Velocity, Ajax 등이 있다. 제어 계층은 사용자 요청을 검증하고 알맞은 형태로 변환하여 비즈니스 계층으로 전달하는 역할을 하고 사용할 수 있는 프레임워크는 Servlet, 스트럿츠, WebWork, Turbine 등이 있다. 그리고 초창기 개발은 MVC Model2 아키텍처와 같은 디자인 패턴을 기반으로 직접 개발하였지만 최근에는 개발 환경을 제공하는 MiPlatform, Gauce, TrustForm, Flex 등과 같은 X-Internet 도구를 사용하기도 한다.

비즈니스 계층은 업무에 대한 동작을 정의하거나 관련 객체 또는 컴포넌트를 연결하는 계층이다. 즉 사용자 정보를 관리하는 세션, 트랜잭션 처리 등의 업무와 다른 계층들과 통신하기 위한 인터페이스 제공 및 해당 계층의 객체들 간의 관계를 관리하는 등이 비즈니스 계층의 역할로 정의할 수 있다. 그래서 시스템이 크지 않거나 아키텍처를 정의하지 않고 개발할 경우 비즈니스 계층에 있어야 하는 내용을 제어 계층이나 퍼시스텐스 계층에서 찾아 볼 수 있다. 비즈니스 계층에서 사용할 수 있는 프레임워크는 스프링, EJB 등이 있다.

인테그레이션 계층은 DBMS를 처리되는 경우가 많아 퍼시스턴스 계층(Persistence Layer)이라고 하기도 한다. 주로 데이터의 생성・검색・수정・삭제와 같은 CRUD연산을 수행하는데 ORM 툴을 사용하여 처리하기도 한다.

하지만 데이터가 처리뿐 아니라 다른 업무 시스템이나 웹 서비스, XML, 파일 시스템 등을 고려한 개발이라면 레거시(Legacy) 개념을 갖는 EIS(Executive Information System) 계층이라고 한다. 인테그레이션 계층에서 사용할 수 있는 프레임워크는 JDBC(JavaDataBaseConnectivity), EJB(EnterpriseJavaBean), JDO(JavaDataObjects), 하이버네이트, iBATIS, TopLink 등이 있다.

리소스 계층은 정보시스템에서 사용하는 서버, 시스템, 라이브러리, 모듈, 컨트롤 등과 같은 외부 자원을 말한다.

2.3 J2EE 기반의 응용소프트웨어 개발 방식[7][8]

자바로 응용 소프트웨어를 개발할 때 아키텍처를 NonEJB, EJB, Lightweight 컨테이너 등 세가지 방식으로 구분한다.

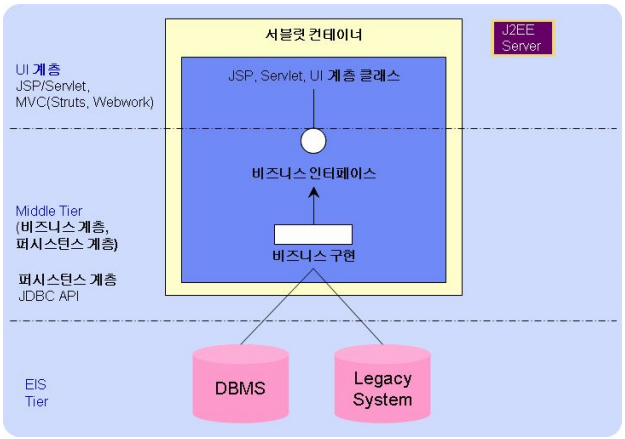
NonEJB 아키텍처에 의한 개발은 일반적으로 EJB를 사용하지 않고 개발하는 방식으로 자바 기본 문법과 JDBC, Servlet, JSP 등을 이용하여 특정 정형화된 방식에 구애받지 않는 개발 방식이다.

EJB 아키텍처에 의한 개발은 세션빈(SessionBean), 엔티티빈(EntityBean), 메시지드리븐빈(MessageDrivenBean) 등 EJB에서 지원하는 컴포넌트를 배치하여 개발하는 방식이다.

Lightweight 컨테이너 아키텍처에 의한 개발은 기존의 Non EJB EJB 방식에 한계를 보완하여 자체 프레임워크 및 오픈소스 프레임워크를 기반으로 개발하는 방식이다.

2.3.1 NonEJB 아키텍처에 의한 개발

NonEJB 아키텍처는 EJB가 등장하기 전에 많은 개발자들이 Servlet 컨테이너 기반으로 개발을 진행했던 개발 방식으로 개발자에 따라 다양한 형태로 개발하는 것이 가능하기 때문에 하나의 형태로 정형화하는 것은 의미가 없다. [그림2-2]는 NonEJB 아키텍처에 대한 구조를 정의하고 [표2-2]는 NonEJB 아키텍처의 장단점을 나열한다.



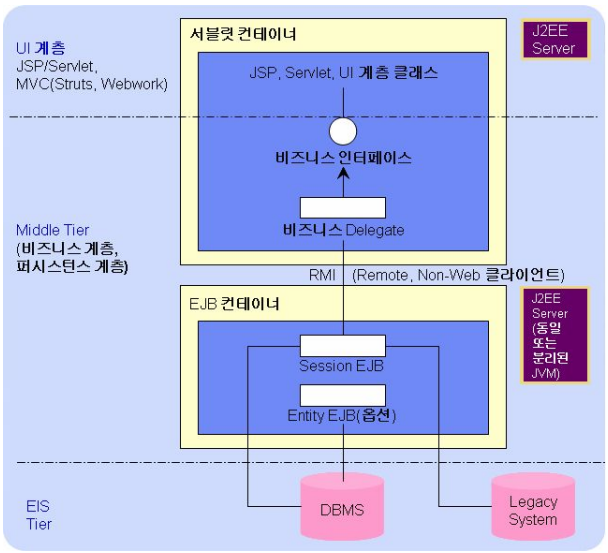
[그림 2-2] NonEJB 아키텍처

[표 2-2] NonEJB 아키텍처의 장단점

구분	내용
장점	• 이해가 쉽고 개발이 빠르다. • 저렴한 Servlet 컨테이너만 운영 가능하고 이식성이 좋다. • 개발주기가 EJB보다 짧기 때문에 개발생산성의 향상된다.
단점	• 초기엔 개발이 빠르나 점점 느려진다. • 개발 계층이 명확하지 않아 중복 개발된다. • 분산 환경을 지원할 수 없다. • 생명주기 및 트랜잭션 관리를 직접 구현해야 한다.

2.3.2 EJB 아키텍처에 의한 개발

RMI의 기반으로 작성된 EJB는 분산 환경에 대한 지원이 과거보다 쉬워지고 UI 계층과 비즈니스 계층을 논리적 분리뿐 아니라 물리적으로도 분리가 가능함으로 애플리케이션 개발에 있어서 계층화가 가능하게 되었다. [그림 2-3]는 EJB 아키텍처에 대한 구조를 정의하고 [표2-3]는 EJB 아키텍처의 장단점을 나열한다.



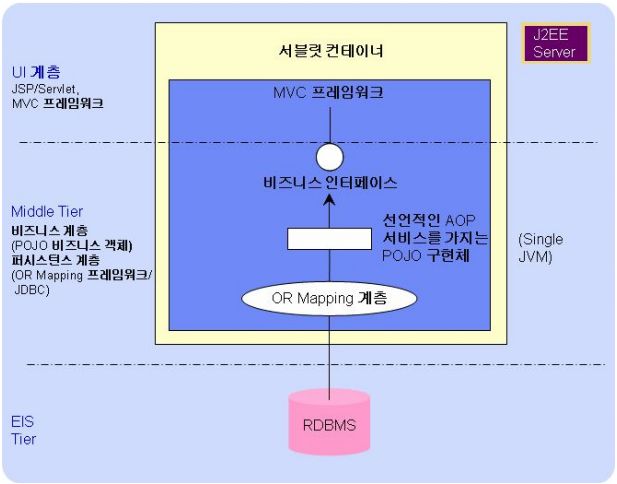
[그림 2-3] EJB 아키텍처

[표 2-3] EJB 아키텍처의 장단점

구분	내용
장점	• 정형화된 비즈니스 계층을 제공한다. • EJB는 계층 분리 뿐 아니라 물리적인 분리(분산)까지 가능하다. • 생명주기 및 트랜잭션 관리를 자동으로 처리가 가능하다. • EJB는 다양한 클라이언트에 대한 지원이 가능하다.
단점	• 분산처리 시 필요한 객체를 직렬화로 실행속도가 저하된다. • 개발 사이클이 소스 수정, 빌드, 배포, 테스트와 같이 복잡한 과정을 거치기 때문에 개발생산성이 저하된다. • EJB는 EJB 컨테이너에 종속되어 이식성이 떨어진다.

2.3.3 Lightweight 컨테이너 아키텍처에 의한 개발

Lightweight 컨테이너 아키텍처는 가벼우면서도 컨테이너 기능을 가지고 있는 아키텍처라는 의미를 가진다. 즉 EJB 컨테이너와 비교하여 복잡하고 세밀한 서버의 역할은 축소하고 트랜잭션 처리, Security, 빈의 생명주기관리와 같은 기능은 컨테이너에서 지원 가능하도록 한다. [그림2-4]는 Lightweight 컨테이너에 대한 구조를 정의하고 [표2-4]는 Lightweight 컨테이너의 장단점을 나열한다.



[그림 2-4] Lightweight 컨테이너 아키텍처

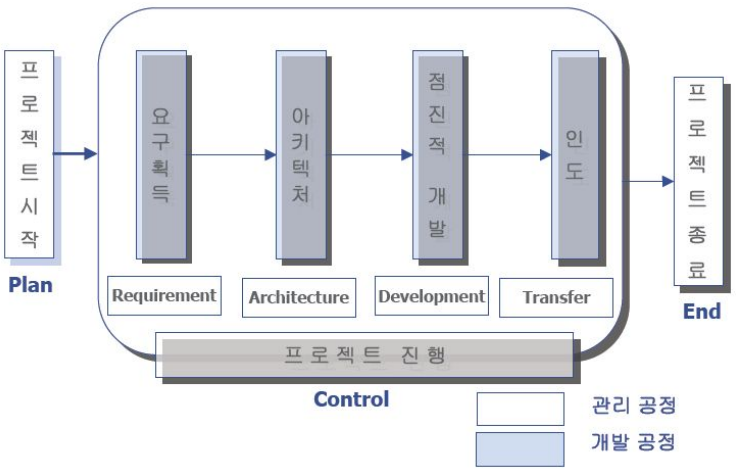
[표 2-4] Lightweight 컨테이너 아키텍처의 장단점

구분	내용
장점	• EJB에 설정이 쉽게 서버에 종속되지 않아 이식성이 좋다. • 특정 인터페이스에 종속되지 않은 POJO(Plain Old Java Object)를 기반으로 하기 때문에 테스트가 용이하다. • AOP(Aspect Oriented Programming)를 지원한다.
단점	• 분산 환경을 지원하지 못한다. • 아직까지 Lightweight 컨테이너에 대한 표준이 없다. • 기존 방식에 비해 보편화 되지 않았다.

제 3 장 오픈소스 프레임워크를 이용한 개발

3.1 오픈소스 프레임워크를 이용한 아키텍처 설계[9][10]

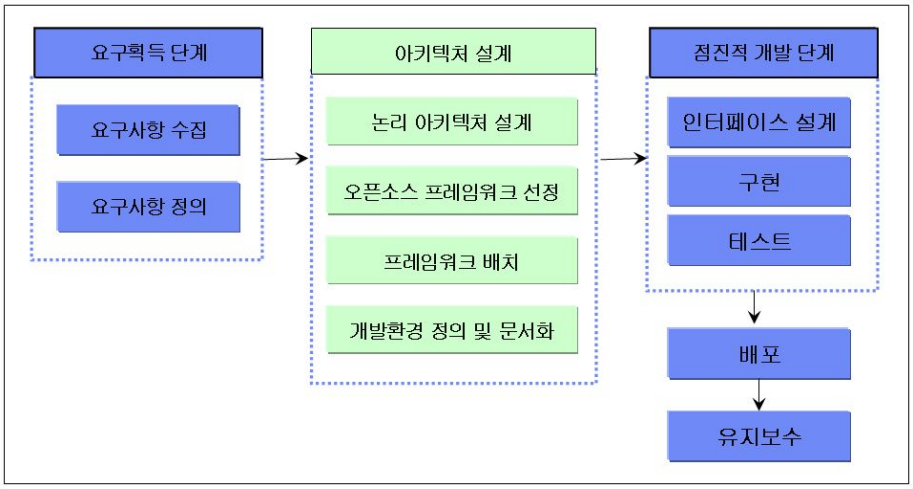
전자통신연구원에서 발표한 마르미III의 개발 방법론은 삼성SDS, 제임스마틴코리아, 컴포넌트비젼, 투이컨설팅, 한더정보시스템, 화이트정보통신 등에 의해 공통 연구되었다. 마르미III는 EJB 기반설계 및 구현 절차 및 지침과 프로젝트 관리 활동을 강화하였다. 그리고 개발 공정과 프로젝트 관리 공정을 분리하고 프로젝트 초기 계약과정을 포함하였다.



[그림 3-1] 마르미III v4.0 공정

[그림3-1]의 개발 공정에서 아키텍처 단계는 시스템 품질을 고려한 재사용성의 증대, 개발 초기에 견고한 아키텍처 정의, 소프트웨어 컴포넌트 아키텍처를 정의한다. 점진적 개발 단계에서는 미니프로젝트(Time Box), 소규모 팀으로 분할, 엄격한 개발일정 관리, 1개 이상 컴포넌트 개발, 순차적/병행적 진행, 프로젝트 위험 최소화, 고객 중심의 변경관리, 사용자 참여 검증, 개발자의 지속적 동기 부여 위험관리 등의 역할을 수행한다.

오픈소스 프레임워크를 이용하여 소프트웨어를 개발하기 위해서 마르미III를 응용하여 개발 프로세스를 [그림 3-2]와 같이 작성한다.



[그림 3-2] 마르미III를 응용한 개발 프로세스

마르미III의 개발 공정을 응용하여 프로세스는 요구획득 단계, 아키텍처 설계, 점진적 개발 단계 등으로 개발이 진행된다.

요구획득 단계에서는 주로 프로젝트에 대한 요구사항 파악 및 개발 환경 정의하는 단계로서 요구사항 수집, 요구사항 정의 단계로 세분화된다.

아키텍처 설계단계는 요구사항을 기반으로 프레임워크를 선정하고 선정된 프레임워크를 기반으로 아키텍처를 설계하는 단계로서 논리 아키텍처 설계, 오픈소스 프레임워크 선정, 프레임워크 배치, 개발환경 정의 및 문서화 단계로 세분화 된다.

점진적 개발 단계는 설계된 아키텍처를 기반으로 업무에 필요한 인터페이스 설계 및 실제 개발하는 단계로 인터페이스 설계, 구현, 테스트 단계로 세분화 된다. 점진적 개발 단계는 업무 크기에 따라 한번 이상 반복될 수 있다. 이후 개발이 끝나면 배포하고 유지보수 한다.

3.1.1 도서 쇼핑몰 구축을 위한 개발 환경

도서 쇼핑몰을 구축하기 위한 개발 환경은 [표 3-1]과 같이 정의하였다 [6][11].

[표 3-1] 응용 소프트웨어 개발 환경

구분	상용 소프트웨어 기반	오픈소스 소프트웨어 기반
OS	Windows/Linux	
JVM	J2SDK 4.X 이상	
WAS	BEA WebLogic 9.X	Tomcat 5.X, JBoss 4.2
프리젠테이션 계층	HTML, Servlet, JSP	
비즈니스 계층	EJB의 세션빈	스프링
퍼시스턴스 계층	EJB의 엔티티빈	하이버네이트
DBMS	Oracle 9i	MySQL 4

개발 환경은 응용 소프트웨어의 OS, JVM, WAS의 환경 뿐 아니라 상용 프레임워크와 오픈소스 프레임워크를 이용하여 각각 사례연구를 하기 위해서 상용 소프트웨어를 기반으로 개발한 환경과 오픈소스 소프트웨어를 기반으로 한 개발한 환경을 정의하였다. 단 프레임워크의 구분 및 확인을 위해 프리젠테이션 계층은 공통으로 HTML, Servlet, JSP로 구현하고 특별한 프레임워크를 사용하지 않는 대신 MVC Model2 디자인 패턴을 응용하여 작성한다.

3.1.2 도서 쇼핑몰을 구축하기 위한 요구사항 정의

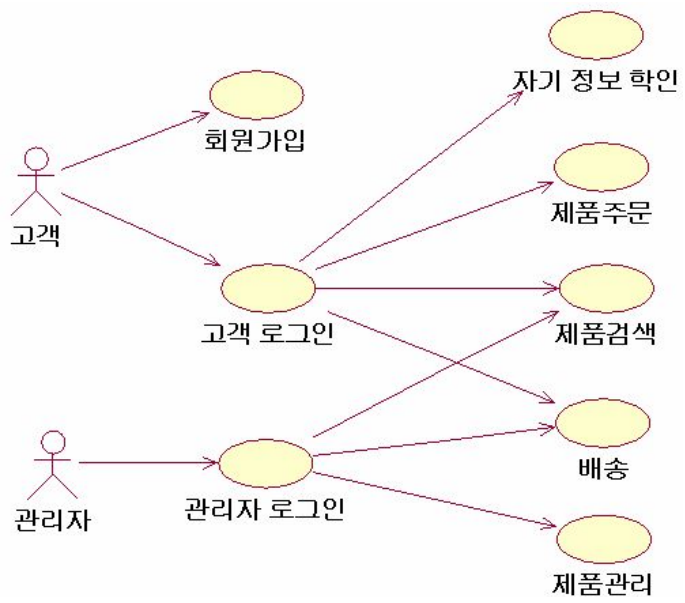
앞에서 작성한 개발 프로세스의 공정 중 요구획득 단계로서 요구사항 수집 및 요구사항 정의 단계이다.

본 연구에서 사례연구로 개발하기 위한 프로젝트는 회원 전용 도서 쇼핑몰 구축이다. 이에 대한 요구사항은 다음과 같이 정의한다.

- 본 사이트는 쇼핑몰은 등록된 회원만 물건을 주문할 수 있다.
- 사이트에 접속하면 로그인 또는 회원가입 페이지를 보여준다.

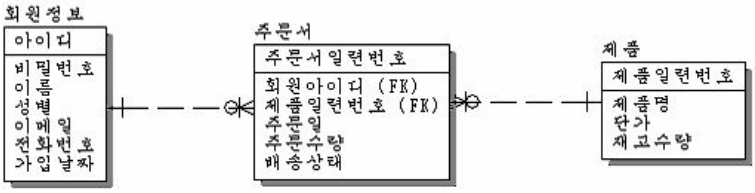
- 로그인하면 제품 목록을 확인 할 수 있다.
- 고객은 장바구니에 특정 제품을 임시로 등록 할 수 있다.
- 장바구니에 등록된 제품의 구입 수량을 변경하여 주문할 수 있다.
- 장바구니에 등록된 제품의 특정 제품만 구매할 수 있다.
- 구매한 제품은 구매목록에 추가되고 장바구니에는 삭제된다.
- 쇼핑몰 재 접속시 장바구니에 등록된 제품 정보는 삭제된다.
- 회원은 주문한 제품의 배송정보 등 주문 정보를 확인할 수 있다.
- 관리자는 애플리케이션 프로그램으로 별도로 개발한다.
- 관리자는 로그인후 제품 정보를 등록할 수 있다.
- 등록된 제품 정보를 확인할 수 있다.
- 고객이 주문한 주문 목록을 확인할 수 있다.
- 고객이 주문한 정보를 바탕으로 배송 상태를 처리 할 수 있다.

회원 전용 쇼핑몰의 요구사항을 기반으로 작성된 유스케이스 다이어그램(UseCase Diagram)이다.



[그림 3-3] 회원전용 쇼핑몰에 대한 유스케이스 다이어그램

회원전용 쇼핑몰은 회원가입한 회원만이 제품을 구매할 수 있고 주문은 등록된 제품을 여러 번 구매할 수 있다. 이러한 요구사항을 기반으로 작성된 데이터베이스 테이블은 회원정보, 제품, 주문서 정보를 관리하도록 정의하였다. [그림 3-4]는 요구사항을 기반으로 작성된 데이터베이스 논리 모델링이다.



[그림 3-4] 회원전용 쇼핑몰에 대한 데이터베이스 논리 모델링

3.1.3 오픈소스 프레임워크 선정[5][6]

[그림 3-2]의 개발 프로세스 공정에서 아키텍처 설계를 위해 J2EE 아키텍처의 각 계층에 맞는 프레임워크를 선정한다. 상용 프레임워크와 오픈소스 프레임워크의 올바른 선정을 위해 인테그레이션 계층에 사용할 수 있는 프레임워크와 비즈니스 계층에서 사용할 수 있는 프레임워크의 특징을 [표3-2], [표3-3]을 참고하여 조사한다.

[표 3-2] 인테그레이션 계층에 사용할 수 있는 프레임워크

구분	내용
JDBC	• JDBC는 Sun에서 인터페이스 제공 • Sun의 표준 인터페이스를 기반으로 각 벤더에서 구현한 DBMS드라이버 제공한다. • java.sql.*를 구현한 다형성으로 DBMS에도 독립 코딩 가능 • http://java.sun.com/products/jdbc/overview.html
EJB	• EJB의 엔티티빈을 이용하여 컨테이너에 의해 DBMS 처리 • CMP(Container Managed Persistence), BMP(Bean Manage Persistence)으로 구분된다. • EJB2.0에서는 CMR(Container Managed Relationship)과 EJB-QL를 제공하고 EJB3.0에서는 POJO를 사용 가능하다. • http://java.sun.com/products/ejb/
JDO	• JDO API는 Java 기술에서의 아카이브 객체 지속성에 대한 표준적이고 명확한 방법을 제공한다. • XML 메타데이터와 바이트코드를 확장, 조합하여 사용한다 • Sun의 표준에 따르고 API가 간단하다. • JDO는 도메인 객체 모델과 관리 클래스를 구분한다. • http://java.sun.com/jdo/
하이버네이트	• ORM 프레임워크로 JMX, HQL 등을 지원한다. • XML과 매핑해서 CRUD은 POJO를 이용하여 처리된다. • 문서화와 커뮤니티 지원이 활발하다. • http://www.hibernate.org/
iBatis	• 완벽한 ORM은 아니다. • SQL 쿼리를 이용하여 매핑을 한다. • 개발자가 익숙한 SQL을 사용함으로 오버헤드가 적다. • http://ibatis.apache.org/
TopLink	• 오라클 애플리케이션 서버에서 제공하는 프레임워크이다. • 현재는 오픈소스 소프트웨어이다. • Java 객체와 EJB를 관계형 데이타베이스 테이블에 저장하는 방식으로 관리하기 쉽고 융통성 있는 메커니즘을 제공한다. • http://www.oracle.com/tools/toplink_adf.html

[표 3-3] 비즈니스 계층에 사용할 수 있는 프레임워크

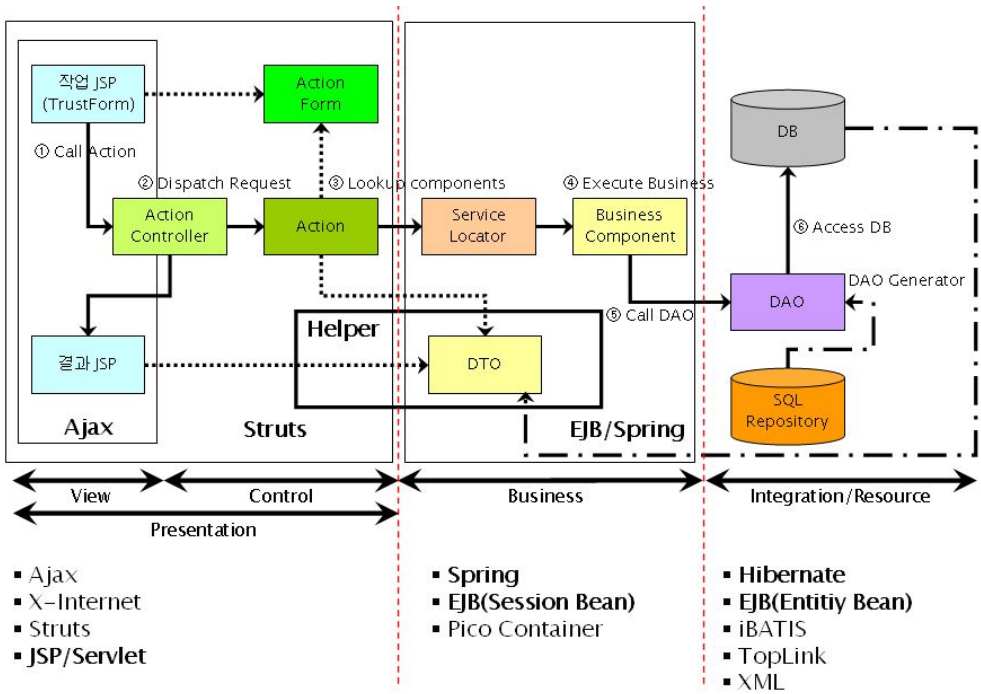
구분	내용
EJB	• 정형화된 비즈니스 계층 제공 • EJB의 세션빈(Stateless, Stateful)이 비즈니스 업무 수행 • EJB3.0에서는 POJO 지원 • 분산처리, JNDI, 트랜잭션 등 EJB컨테이너에서 모두 지원 • http://java.sun.com/products/ejb/
스프링	• 로드 존슨에 의해 2003년 오픈소스 프로젝트로 시작 • EJB의 단점을 보완하고 EJB의 역할 수행 • J2EE 서비스에 종속되지 않으면서 계층화, 모듈화가 잘 되는 매우 유용한 프레임워크이다. • http://www.springframework.org/

상용 프레임워크를 이용한 개발은 EJB를 이용하여 소프트웨어를 개발하는데 비즈니스 계층은 EJB의 세션빈 중에서 무상태(Stateless)를 기반으로 하고 인테그레이션 계층은 EJB의 엔티티빈 중에서 BMP, CMP 기반으로 선정하여 소프트웨어를 작성한다[4].

오픈소스 프레임워크는 로드 존스(Rod Johnson)에 의해 시작된 프로젝트로 기존의 EJB의 단점을 보완하고 경량의 비즈니스 계층을 구현할 수 있는 환경을 제공하는 스프링 프레임워크을 선정하고 인테그레이션 계층은 ORM의 탁월한 기능을 가진 하이버네이트 프레임워크로 선정한다[7][8][12].

3.1.4 오픈소스 프레임워크를 이용한 아키텍처 설계[4][13][14]

[그림 3-5]는 J2EE 아키텍처의 각 계층에 앞에서 선정된 프레임워크를 배치하여 아키텍처를 설계하였다.



[그림 3-5] J2EE 아키텍처를 기반으로 프레임워크 배치

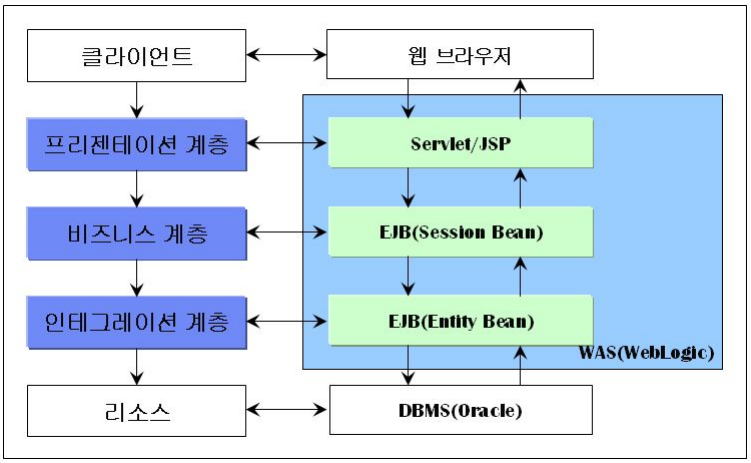
프리젠테이션 계층은 뷰와 컨트롤 계층으로 분리되어 Ajax, X-Internet 도구, Structs, JSP/Servlet 등이 사용가능하지만 본 논문에서는 개발결과를 확인하기 위해서 JSP/Servlet를 이용하여 통일해서 개발한다. Servlet은 메시지를 고객에게서 요청을 받으면 비즈니스 계층에 전송해 주는 역할을 하고 JSP는업무 처리결과를 비즈니스 계층에서 전송받거나 요청하여 클라이언트에게 전송해 주는 역할을 한다.

비즈니스 계층에서 사용할 프레임워크는 EJB의 세션빈, 스프링으로 한다.

비즈니스의 계층은 프리젠테이션에서 요청이 들어오면 처리 객체의 위치를 확인하여 연결해 주는 역할을 하고 결과는 DTO(Data Transfer Object)로 객체화해서 프리젠테이션 계층에 전송한다.

인테그레이션 계층에서 사용할 프레임워크는 EJB의 엔티티빈, 하이버네이트로 한다. 비즈니스 계층에 요청 들어오는 업무 중에 데이터베이스를 처리해야 하는 업무가 들어오면 해당 인테그레이션 계층의 위치를 찾아서 메시지를 전송해 준다. CRUD 연산결과 비즈니스 계층에 전송하면 결과를 DTO객체에 넘겨주면 DTO에서 프리젠테이션 계층에서 처리할 수 있는 형태로 객체화한다. 단 인테그레이션 계층에서 DBMS를 처리 시 어떤 프레임워크를 쓰더라도 JDBC기반으로 된 프레임워크라면 DBMS에 독립적이다. 단 인테그레이션 계층의 JNDI(Java Naming and Directory Interface)의 데이터 소스 설정을 어떻게 하느냐에 따라 인테그레이션 계층을 서비스 하는 서버에 종속될 수는 있다.

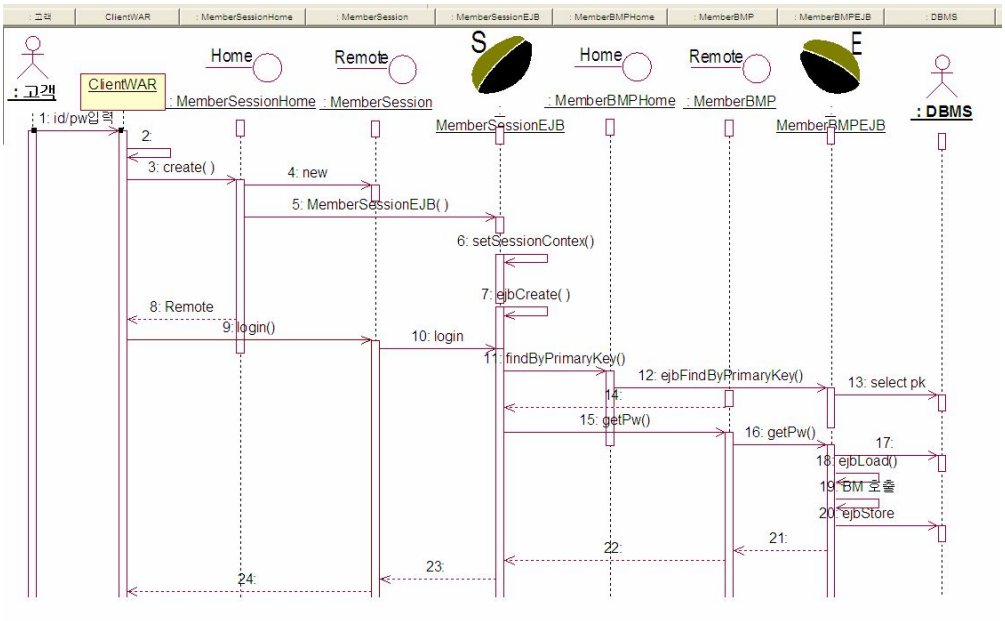
3.2 상용 프레임워크를 이용한 개발[4][7][13][15]



[그림 3-6] J2EE 아키텍처의 각 계층에 배치된 상용 프레임워크

[그림 3-6]은 J2EE 아키텍처의 각 계층에 배치할 수 있는 상용 프레임워크와 관계를 설명하고 있다. 클라이언트 계층은 웹 브라우저가 담당하고 리소스 계층은 DBMS가 담당하기 때문에 개발 영역에서는 제외된다. 웹로직(WebLogic) 어플리케이션 서버에 프리젠테이션 계층은 Servlet/JSP를 이용하여 개발, 비즈니스 계층은 EJB의 세션빈을 기반으로 개발, 인테그레이션 계층은 EJB의 엔티티빈을 기반으로 개발한다.

각 계층을 연결하는 인터페이스는 [그림 3-7]의 상용 프레임워크를 기반으로 개발시 처리되는 시퀀스 다이어그램(Sequence Diagram)을 보면 확인할 수 있다.



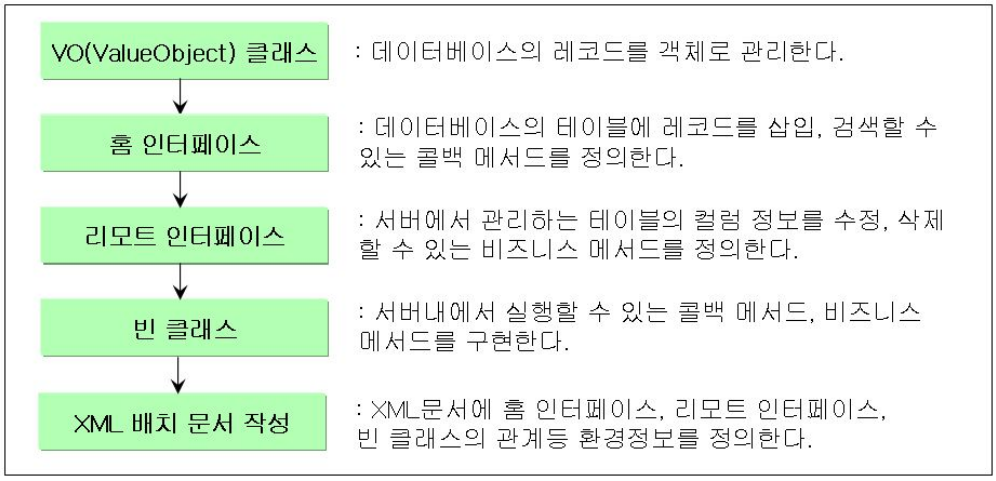
[그림 3-7] 상용 프레임워크를 이용한 시퀀스 다이어그램

프리젠테이션 계층은 Servlet/JSP로 공통으로 사용하기 때문에 ClientWAR로 표현하고 비즈니스 계층은 javax.ejb.EJBHome 인터페이스를 상속받은 홈 인터페이스, javax.ejb.EJBObject 인터페이스를 상속받은 리모트 인터페이스,

javax.ejb.SessionBean 인터페이스를 상속받아 구현한 빈클래스로 구성된다. 빈 클래스에 의해 호출되는 인테그레이션 계층은 홈인터페이스, 리모트인터페이스, javax.ejb.EntityBean을 상속받아 구현한 빈 클래스로 구성되어 있다. 호출 순서는 프리젠테이션 계층, 비즈니스 계층 인테그레이션 계층이지만 개발시에는 인테그레이션 계층, 비즈니스 계층, 프리젠테이션 계층 순으로 개발한다.

3.2.1 엔티티빈을 이용한 인테그레이션 계층 개발

다음 [그림 3-8]은 엔티티빈을 이용하여 인테그레이션 계층을 개발할 때 개발하는 순서이다.



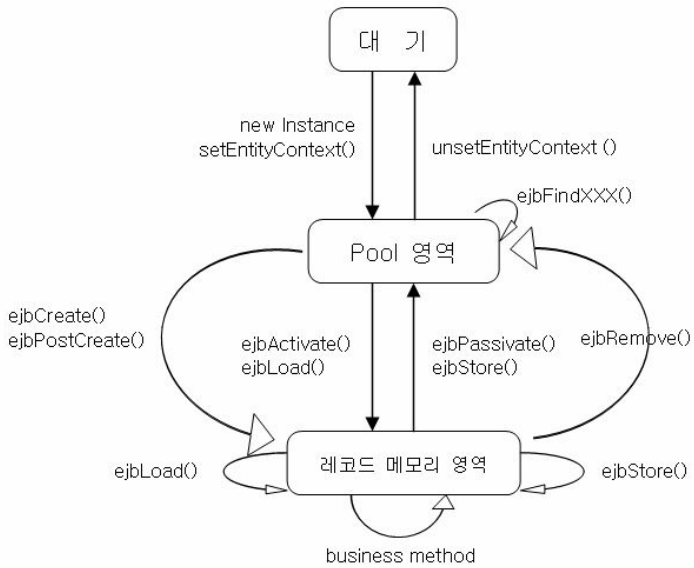
[그림 3-8] 엔티티빈의 개발 순서

인테그레이션 계층은 데이터베이스를 주로 처리하기 때문에 데이터베이스에서 가져온 결과를 객체로 관리하면 효율적이다. VO클래스는 데이터 베이스에서 목록을 가져올 때 사용자 정의 타입으로 객체의 값을 관리할 수 있다. VO클래스를 만드는 기준은 테이블을 컬럼 정보, 뷰 테이블의 컬럼 정보, 업무에 필요한 내용 등의 정보를 기준으로 작성된다.

홈인터페이스는 javax.ejb.EJBHome 인터페이스를 상속받은 인터페이스를 정의한다. 엔티티빈의 홈인터페이스의 역할은 데이터베이스의 레코드를 삽입, 삭제, 검색 조건 등에 관한 콜백 메서드를 선언할 수 있다.

리모트인터페이스는 javax.ejb.EJBObject 인터페이스를 상속받은 인터페이스를 정의한다. 리모트인터페이스는 특정 데이터베이스의 컬럼 또는 레코드 단위로 메모리에서 값을 변경하거나 가져오는 비즈니스 메서드를 setXxx(), getXxx()형태로 선언한다.

홈인터페이스와 리모트인터페이스는 클라이언트가 신호를 보내기 위해 알아야하는 목록을 선언하는 역할을 한다면 그 신호를 받아서 처리하는 구현부는 javax.ejb.EntityBean을 상속받은 빈클래스에 정의한다. 단 엔티티빈의 CMP로 정의하면 기본 SQL문을 자동으로 처리하기 때문에 생략 가능하지만 BMP로 정의하면 [그림 3-9]의 엔티티빈의 생명주기를 이해하여 SQL문을 특정메서드에 맞게 구현해야 한다.



[그림 3-9] 엔티티빈 생명주기

엔티티빈의 생명주기의 콜백 메서드에 역할은 [표 3-4]에서와 같다.

[표 3-4] 엔티티빈의 콜백메서드의 역할

메서드명	빈 클래스에서 구현	위치
ejbCreate([인자])	인자수를 기준으로 삽입하는 INSERT 문 작성	홈인터페이스 사용자 정의 선언
ejbPostCreate	ejbCreate를 보고 옵션으로 정의	
ejbFindXXX([인자])	인자를 조건절로 pk를 검색하는 SELECT문 작성	
ejbLoad()	메모리에 있는 pk를 기준으로 모든 레코드 정보를 검색하는 SELECT문 작성	엔티티빈에서 상속받은 메서드
ejbStore()	메모리에 있는 pk를 기준으로 레코드 정보 수정하는 UPDATE문 작성	
ejbRemote	메모리에 검색된 pk를 기준으로 특정 레코드를 삭제하는 DELETE문 작성	

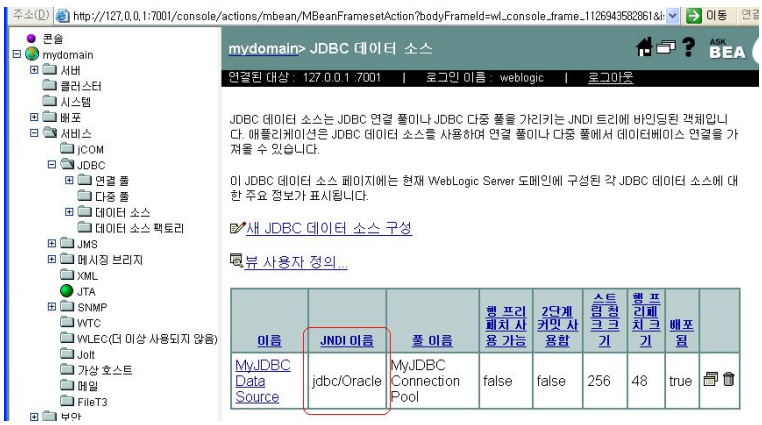
이와 같은 엔티티빈의 특성을 이해하여 데이터베이스를 처리하는 인테그레이션 계층을 구현한다. 본 논문에서는 회원정보처리, 주문서 컴포넌트는 BMP로 작성하고 제품관리는 CMP로 작성하였다. [그림 3-8]는 회원정보처리에 관한 컴포넌트에 대한 클래스 다이어그램이다.

MemberBMP 컴포넌트의 클래스 다이어그램에 해당하는 클래스를 작성한다.

- 모델 클래스 - MemberRec.java
- 홈 인터페이스 - MemberBMPHome.java
- 리모트 인터페이스 - MemberBMP.java
- 빈 클래스 - MemberBMPEJB.java

[그림 3-10] MemberBMP 컴포넌트의 클래스 다이어그램

[그림 3-11]은 데이터베이스 정보를 웹로직 관리 프로그램에서 설정 후 “jdbc/Oracle“ JNDI명으로 관리한다.



[그림 3-11] 웹로직 관리 프로그램에서 DBMS 정보 설정하기

[표 3-5]는 MemberBMPEJB.java에 작성된 콜백메서드의 일부 구현부이다. MemberBMP의 핵심 코드이다.

[표 3-5] 빈클래스의 데이터베이스 처리

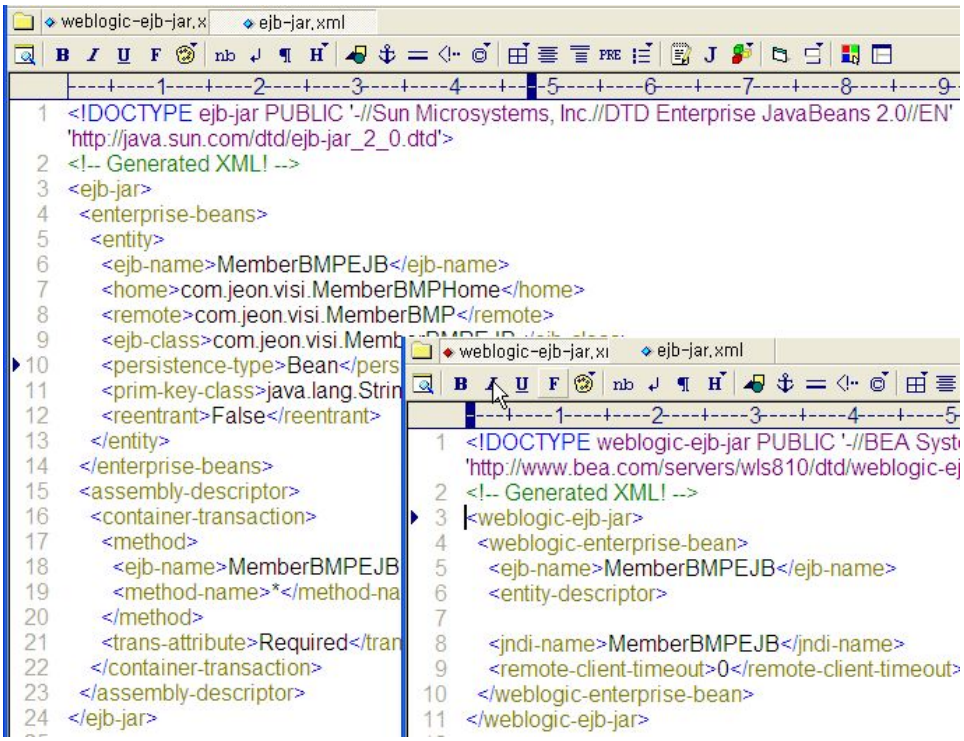
MemberBMPEJB.java

```
.....[생략].....
018 :  public class MemberBMPEJB implements EntityBean
019 :  {
020 :     //1.  레코드정보 모델 타입으로 한번에 저장
021 :     private MemberRec rec;
022 :     private EntityContext ctx;
023 :     private DataSource ds;
024 :
025 :     //2. 상속받은 멤버 재정의
026 :     public void setEntityContext(EntityContext ctx)     {
.....[생략].....
031 :          ds = (DataSource)ic.lookup("jdbc/Oracle");
.....[생략].....
034 :     }
041 :     //PK를 기준으로 레코드 정보 메모리에 올리기
042 :     public void ejbLoad()    {
.....[생략].....
047 :          Connection conn=ds.getConnection();
048 :          PreparedStatement pstmt=conn.prepareStatement(select pw,
name, sex, email, tel, indate from member where id=?);
049 :          pstmt.setString(1, rec.getId());
050 :          ResultSet rs=pstmt.executeQuery();
.....[생략].....
061 :     }
.....[생략].....
```

[그림 3-11]처럼 DBMS 정보를 웹로직 서버에서 관리하고 JNDI명만 오픈하기 때문에 프로그램에서는 [표 3-4]의 31라인처럼 JNDI명으로 lookup만 하면 서버를 통해서 DBMS 정보를 자동으로 처리한다. 그리고 프로그램에서는

47라인처럼 커넥션(Connection) 정보를 얻어와서 필요한 SQL문만 구현하면 된다. 서버에서 커넥션을 관리하기 때문에 사용자는 편하지만 웹로직 서버의 환경 설정이 종속된다는 단점이 있다.

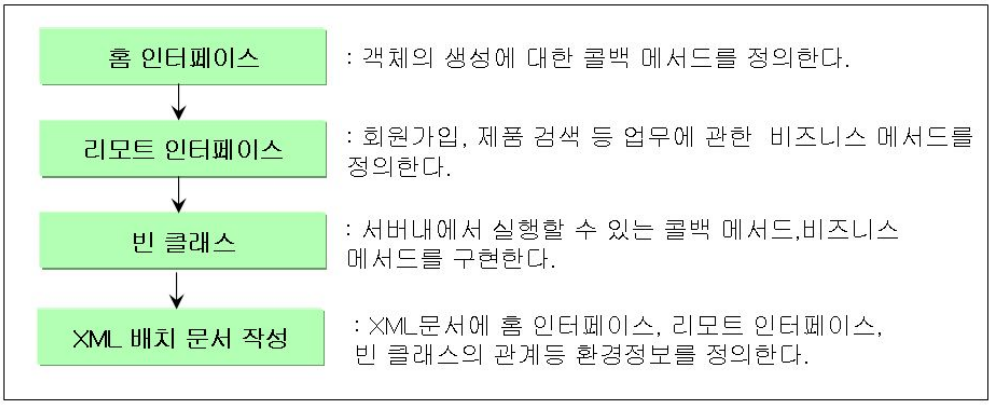
[그림 3-12]처럼 ejb-jar.xml환경 설정은 어떤 어플리케이션 서버라도 BMP의 공통 환경 설정이지만 JNDI의 환경 설정은 weblogic-ejb-jar.xml를 이용하여 작성된다.



[그림 3-12] 엔티티빈의 환경설정 XML 배치문서

3.2.2 세션빈을 이용한 비즈니스 계층 개발

다음 [그림 3-13]은 세션빈을 이용한 비즈니스 계층을 개발할 때 개발하는 순서이다.



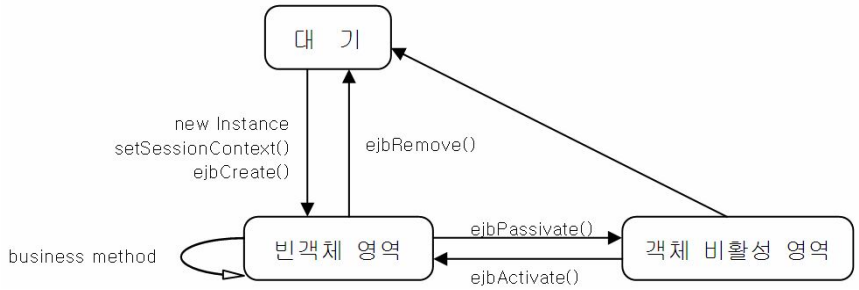
[그림 3-13] 세션빈의 개발 순서

비즈니스 계층은 프리젠테이션 계층에서 요구하는 요구사항을 직접 처리하거나 인테그레이션 계층을 연결하여 처리한 결과를 프리젠테이션 계층에 전송해주는 역할을 한다.

홈인터페이스는 javax.ejb.EJBHome 인터페이스를 상속받고 리모트인터페이스는 javax.ejb.EJBObject 인터페이스를 상속받아 정의하는것은 같지만 세션빈의 홈 인터페이스에서는 객체를 생성하는데 신호만 관리하는 create()만 선언하고 리모트인터페이스에서는 업무에 관한 비즈니스 메서드만 선언한다.

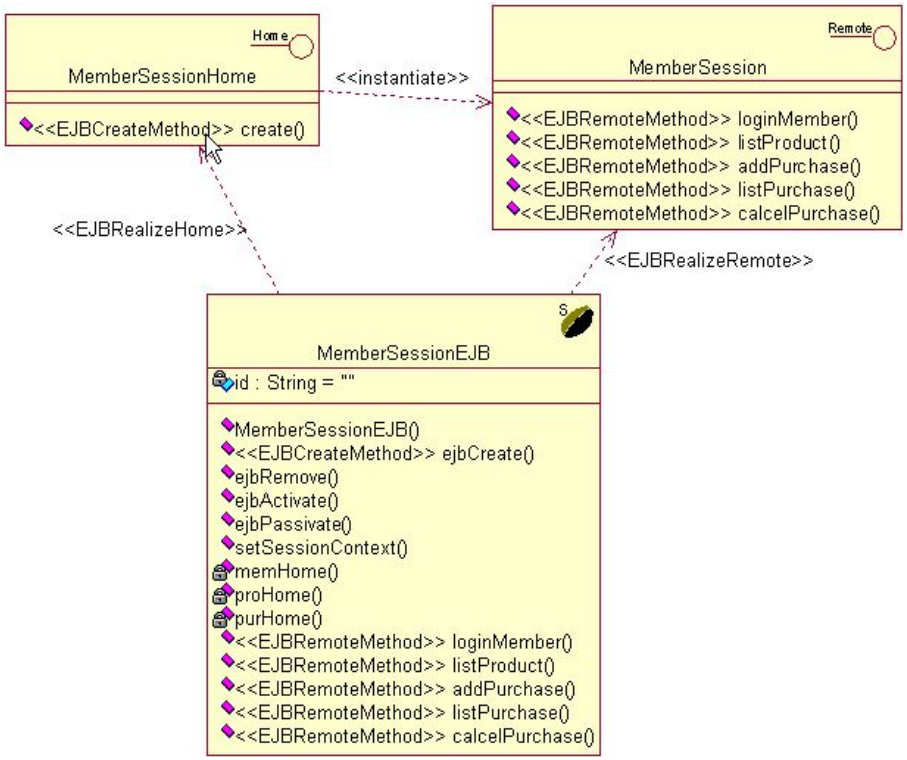
홈인터페이스와 리모트인터페이스의 신호를 받아서 처리하는 구현부는 javax.ejb.SessionBean 인터페이스를 상속받은 빈클래스를 정의한다. [그림 3-14]는 세션빈의 생명주기를 이해하면 세션빈의 역할을 이해하는데 도움이 된다.

세션빈의 콜백메서드는 엔티티빈과 다르게 사용자가 특별한 동작을 구현하지 않아도 서버에서 자동으로 처리해 준다. 그래서 실제 프로그래밍할 때는 상속받은 메서드를 특별한 구현부 없이 {}만으로 구색만 갖춘다.



[그림 3-14] 세션빈의 생명주기

퍼시스턴스 계층에서 작성한 엔티티빈을 고객 업무에 따라 필요한 동작을 한꺼번에 수행할 수 있도록 비즈니스 계층에 세션빈 컴포넌트를 작성한다. [그림 3-15]은 고객 업무에 필요한 정보만 관리하는 세션빈 컴포넌트이다.



[그림 3-15] MemberSS 컴포넌트의 클래스 다이어그램

MemberSS 컴포넌트의 클래스 다이어그램에 해당하는 클래스를 작성한다.

- 홈 인터페이스 - MemberSessionHome.java
- 리모트 인터페이스 - MemberSession.java
- 빈 클래스 - MemberSessionEJB.java

[표 3-6]는 MemberSessionEJB.java에 작성된 비즈니스메서드의 일부 구현부이다.

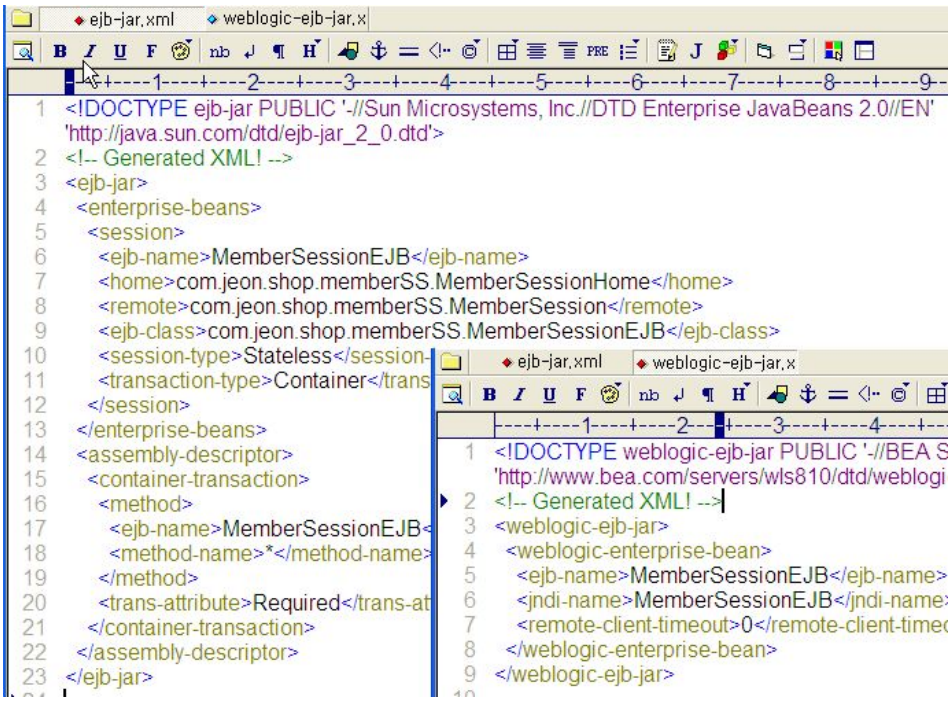
[표 3-6] 빈클래스에서 엔티티빈 연결 후 비즈니스 메서드 처리

MemberSessionEJB.java

```
.....[생략].....
054 :    /** AdminHome 엔티티 빈과 연결 메소드 */
055 :    private MemberBMPHome memHome()  {
056 :      MemberBMPHome memHome=null;
057 :      try{
058 :        Object obj= ctx.lookup("MemberBMPEJB");
059 :        memHome = (MemberBMPHome)
PortableRemoteObject.narrow(obj, MemberBMPHome.class);
060 :      }catch(Exception e){ System.out.println("Member 연결 오류"); }
061 :      return memHome;
062 :    }
.....[생략].....
094 :    //비즈니스 멤서드 오버라이딩...회원 로그인
095 :    public boolean loginMember(String id, String pw)   {
096 :      try{
097        :                                                    MemberBMP
memRemote=memHome().findByPrimaryKey(id);
098 :        if(memRemote !=null && pw.equals(memRemote.getPw()) )
099 :          return true;
100 :          }catch(Exception e){System.out.println(" loginMember() 오
류.....");
101 :      }
102 :      return false;
103 :    }
.....[생략].....
```

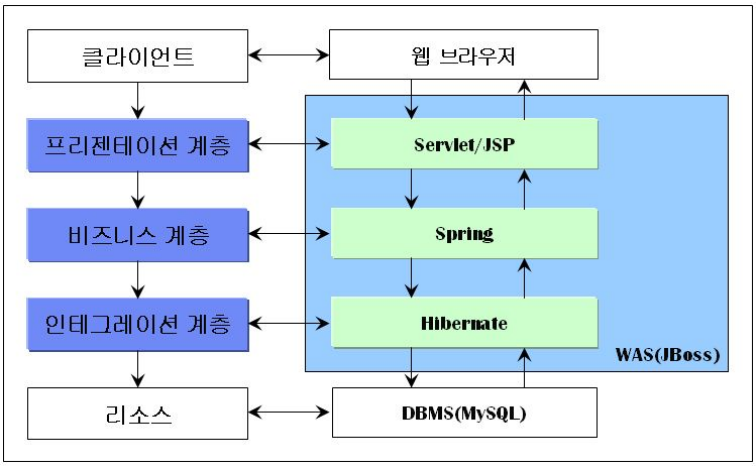
[표 3-6]의 MemberSessionEJB.java에서 비즈니스 메서드를 구현하기 전 55라인처럼 먼저 관련된 인테그레이션 계층의 컴포넌트를 룩업한다. 룩업하는 구현부가 중복됨으로 메서드를 선언해서 관리하면 용이하다. 94라인의 비즈니스 메서드를 구현하기전 memHome()를 호출하고 홈인터페이스에서 필요한 콜백메서드를 호출하면 자동으로 서버에서 관련된 클래스를 실행한 결과를 리턴해 준다. 비즈니스 계층에서는 그 결과만 리턴받아서 98라인처럼 업무를 처리한다.

[그림 3-16]은 엔티티빈의 환경설정 처럼 세션빈의 기본 환경설정 방식은 같다. 즉 EJB의 공통 환경 설정은 ejb-jar.xml에 하고 웹로직 서버에 종속된 환경설정은 weblogic-ejb-jar.xml에 한다.



[그림 3-16] 세션빈의 환경설정 XML 배치문서

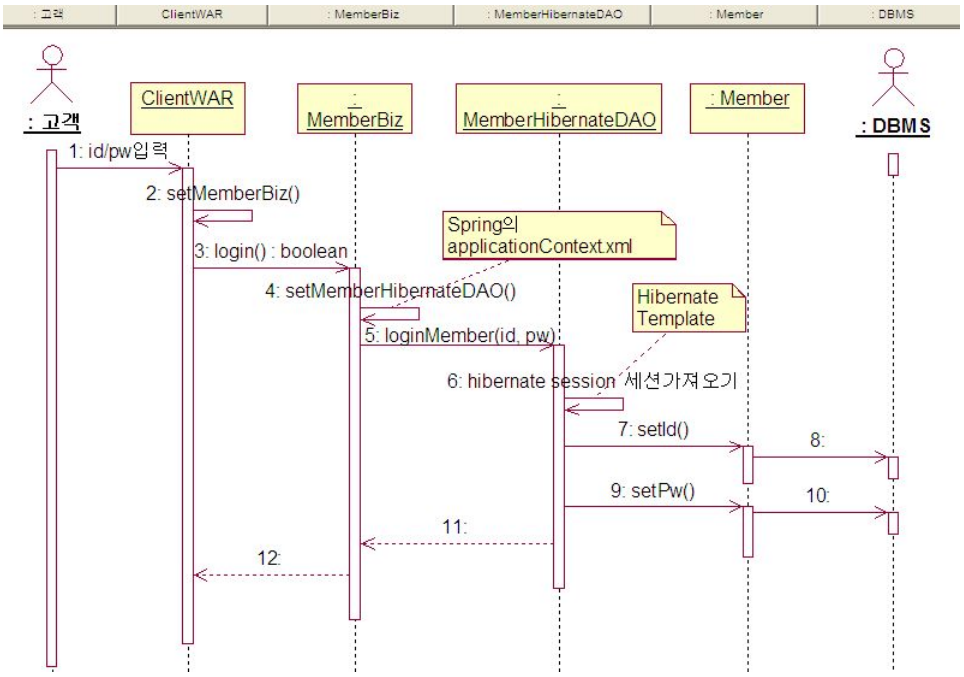
3.3 오픈소스 프레임워크를 이용한 개발[4][7][8][12][13]



[그림 3-17] J2EE 아키텍처의 각 계층에 배치된 오픈소스 프레임워크

[그림 3-17]은 J2EE 아키텍처의 각 계층에 배치할 수 있는 오픈소스 프레임워크와 관계를 설명하고 있다. 클라이언트 계층은 웹 브라우저가 담당하고 리소스 계층은 DBMS가 담당하기 때문에 개발 영역에서는 제외된다. JBoss 어플리케이션 서버에 프리젠테이션 계층은 Servlet/JSP를 이용하여 개발, 비즈니스 계층은 스프링을 기반으로 개발, 인테그레이션 계층은 하이버네이트를 기반으로 개발한다.

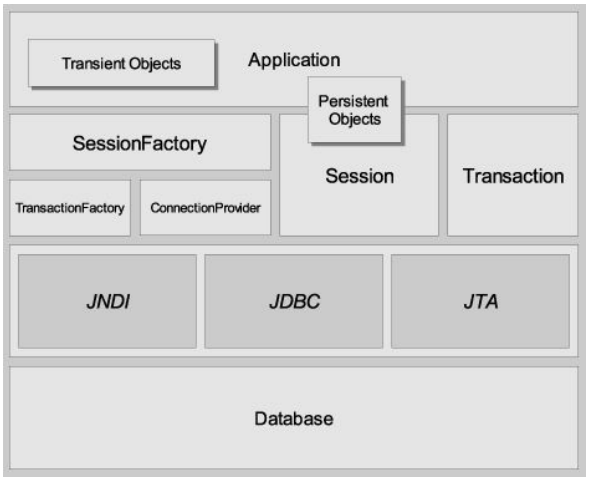
각 계층의 연결은 [그림 3-18]의 오픈소스 프레임워크를 기반으로 개발 시 처리되는 시퀀스 다이어그램을 보면 확인할 수 있다. 오픈소스 프레임워크를 기반으로 한 개발은 소스에 관계가 설정되어 있지 않고 시퀀스 다이어그램의 4, 6번처럼 환경설정 문서에 클래스와 인터페이스 관계 뿐 아니라 객체 생성 래퍼런스까지 관리한다.



[그림 3-18] 오픈소스 프레임워크를 이용한 시퀀스 다이어그램

3.3.1 하이버네이트를 이용한 인테그레이션 계층 개발

[그림 3-19]은 퍼시스턴스 계층에 사용할 하이버네이트 프레임워크의 아키텍처 구조이다.



[그림 3-19] 하이버네이트 프레임워크 아키텍처

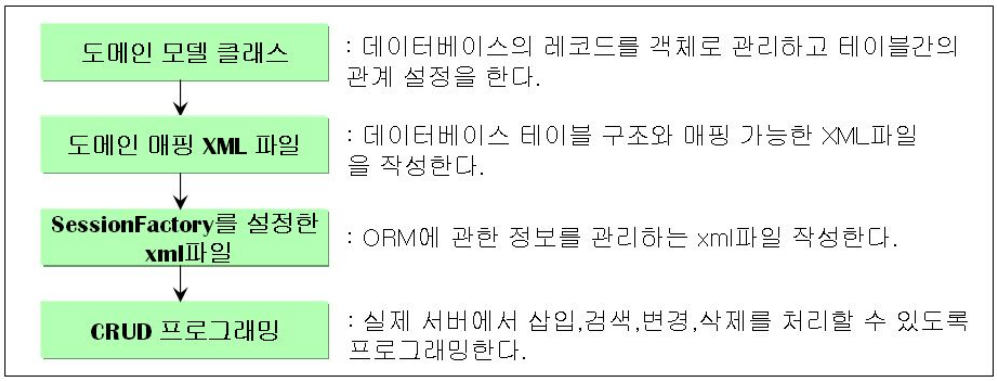
하이버네이트의 아키텍처는 Transient Objects, Application은 업무에 해당하는 것이고 JNDI, JDBC, JTA는 프레임워크에서 이미 제공하는 환경이다. Database는 리소스 계층에 해당하는 것이기 때문에 실제 개발할 때는 SessionFactory, Session, Transaction을 고려하여 개발하면 된다. [표 3-7]은 개발시 고려해야 하는 하이버네이트의 요소에 대한 설명이다.

[표 3-7] 하이버네이트의 기능 요소

패키지 명	기능
SessionFactory	SessionFactory는 JNDI, JDBC, JTA 등 데이터베이스 처리와 관련된 정보를 사용하여 Session을 생성한다.
Session	데이터베이스 커넥션과 연결되며 자바 객체의 영속성을 관리한다.
Transaction	JDBC나 JTA 등 실제 하부 구현에 상관없이 애플리케이션이 동일한 코드를 사용하여 트랜잭션을 처리할 수 있도록 한다.

하이버네이트는 퍼시스턴스를 제공하는 것은 물론이고 효율적인 캐싱 계층과 JMX(Java Manage eXtensions)를 지원한다. 또한 데이터베이스로 객체를 얻어올 수 있는 쿼리 언어인 HQL(Hibernate Query Language)도 제공한다.

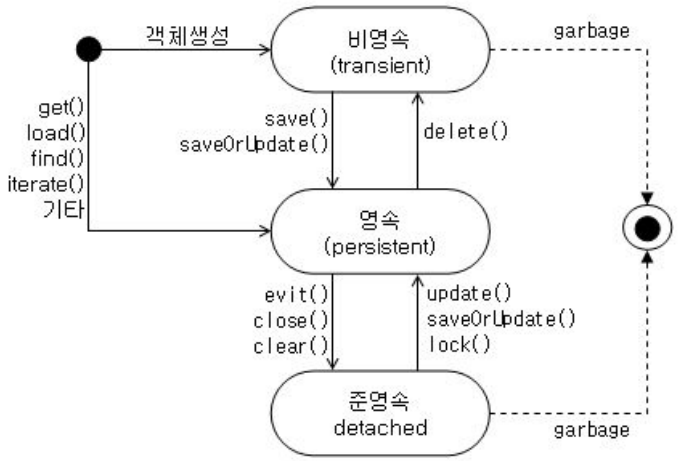
[그림 3-20]은 하이버네이트를 이용한 인테그레이션 계층을 개발할 때 개발하는 순서이다.



[그림 3-20] 하이버네이트의 개발 순서

오픈소스 프레임워크에서도 인테그레이션 계층은 데이터베이스를 주로 처리한다. 처리하는 레코드의 정보를 관리해야 하기 때문에 도메인 모델 클래스를 먼저 작성한다. VO 클래스와 차이점은 레코드의 관계를 설정하고 도메인 매핑 XML문서를 작성하여 같이 관리하게 된다. ORM에 관련된 정보도 XML문서로 관리하고 스프링을 비즈니스 로직으로 사용할 경우엔 CRUD 프로그램은 이후 스프링 프로그램으로 대치할 수 있다.

하이버네이트는 환경문서 정보를 읽고 객체의 생명주기를 관리한다. [그림 3-21]은 하이버네이트의 생명주기이다.



[그림 3-21] 하이버네이트 생명주기

하이버네이트가 관리하는 영속 객체는 비영속, 영속, 준영속의 3가지 상태를 가지고 있다. 비영속 상태는 객체가 생성은 되었지만 하이버네이트에 의해 관리되지 않는 경우를 말한다. 즉 객체는 하이버네이트와 연결되지 않았으며 데이터베이스 테이블에도 매핑된 데이터가 존재하지 않는다. 비영속 상태의 객체를 Session.save(), Session.saveOrUpdate(), Session.persist() 메서드를 사용하여 저장하면 영속 객체가 된다. 이때 Session.get(), Sesstion.load()를 이용하여 조회 기능을 사용할 수 있다. Session이 종료되거

나 Session.evict() 메서드가 호출하여 영속 객체를 캐시에서 삭제하면 준영속 상태의 객체가 된다. 준영속 객체의 프로퍼티 값이 변경되더라도 데이터베이스에 반영되지 않는다. [표 3-8]은 하이버네이트의 영속객체에 대한 정보이다.

[표 3-8] 하이버네이트의 영속객체

구분	내용
비영속 상태	퍼시스턴트 객체를 처음 만들었을 때의 상태이다. 데이터베이스 테이블에 관련 데이터가 없고 연관된 Session이 없다.
영속 상태	현재 활성화된 Session과 연결된 퍼시스턴트 객체이다. 이 상태의 퍼시스턴트 객체는 고유성을 가지며, 프로퍼티 값의 변경이 Session을 통해 자동으로 데이터베이스에 반영된다.
준영속 상태	영속 상태의 퍼시스턴트 객체가 Session과 연결이 끊긴 상태이다.

하이버네이트를 이용한 퍼시스턴스 계층 개발은 다음과 같이 작성한다.

- 도메인 모델 클래스 작성 - Member.java, Poduct.java, Purchase.java
- 도메인 매핑 XML파일 작성 - member.hbm.xml, product.hbm.xml, purchase.hbm.xml
- SessionFactory설정 xml파일 작성 - hibernate.cfg.xml
- CRUD 프로그래밍 - 비즈니스 계층이 스프링 프레임워크 임으로 비즈니스 계층에서 구현한다.

도메인 모델 클래스는 멤버 데이터로 데이터베이스 컬럼 정보와 연결되는 데이터 타입을 선언하고 관계는 HashSet 객체로 데이터를 선언한다. 또한 도메인 매핑 XML 파일을 작성한다.

[표 3-9] 도메인 모델 소스

Member.java

```
.....[생략].....
09 : private String id, pw, name;    //아이디, 비밀번호, 이름
10 : private char sex;          //성별
11 : private String email, tel, indate;       //이메일, 전화번호, 가입날짜
12 : private Set purchases=new HashSet<Purchase>();  //관계
...
50 : public String getId(){return this.id;}       //관계설정
51 : public void setId(String id){this.id=id;}
52 : public void addPurchase(Purchase purchase){
53 :      purchases.add(purchase);
54 :      purchase.setMember(this);
55 : }
56 : public Set<Purchase> getPurchases(){ return purchases;}
.....[생략].....
```

Product.java

```
.....[생략].....
05 : private String proNo;  //일련번호
06 : private String proName;  //제품이름
07 : private int proPrice;       //제품 가격
08 : private int proQuantity; //수량
09 : private Set purchase=new HashSet<Purchase>();
.....[생략].....
```

Purchase.java

```
.....[생략].....
05 :  private int purNo;                    //구매서 일련번호
06 :  private Member member;             //구매 회원 정보 참조
07 :  private Product product;             //구매제품정보
08 :  private String purBayDate;          //구매 날짜
09 :  private int purQuantity;               //구매수량
10 :  private String purState;//발송상태(W대기, S발송중, E완료, C취소)
.....[생략].....
```

[표 3-10] 도메인 매핑 XML파일

member.hbm.xml

```
.....[생략].....
<!-- 데이터베이스 테이블의 컬럼 정보에 관계된 매핑 XML 파일-->
<hibernate-mapping>
  <class name="com.jeon.Member" table="MEMBER">
      <id name="id" type="string" unsaved-value="null" >
          <column name="ID" sql-type="varchar(10)" not-null="true"/>
          <generator class="identity"/>
       </id>
      <property name="pw">
           <column name="PW" sql-type="varchar(10)" not-null="true"/>
       </property>
.....[생략].....
    <set name="purchase" table="PURCHASE" inverse="true"
cascad="all">
        <key column="ID" /> <one-to-many class="Purchase" />
    </set>
  </class>
</hibernate-mapping>
```

product.hbm.xml

```
.....[생략].....
    <set name="purchase" table"PURCHASE" inverse="true"
cascad="all">
        <key column="PURNO" /><one-to-many class="Purchase" />
    </set>
  </class>
.....[생략].....
```

purchase.hbm.xml

```
.....[생략].....
  <class name="com.jeon.Member" table="MEMBER">
  <class name="com.jeon.Product" table="PRODUCT">
     .....[생략].....
       <one-to-many class="member" column="ID" />
       <one-to-many class="product" column="PURNO" />
   </class>
.....[생략].....
```

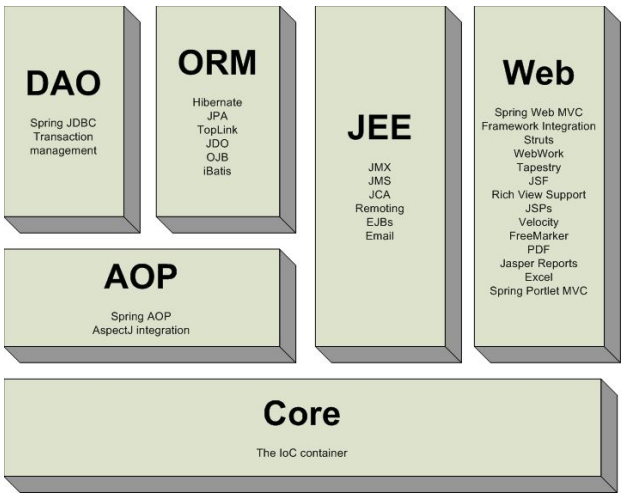
[표 3-11] SessionFactory설정 XML파일

hibernate.cfg.xml

```
.....[생략].....
<hibernate-configuration>
  <session-factory>

    <!-- DBMS 정보 설정 -->
    <property name="connection.driver_class">
      com.mysql.jdbc.Driver
    </property>
    <property name="connection.url">
      jdbc:mysql://127.0.0.1:3306/javadb?characterEncoding=euckr
    </property>
    <property name="connection.username">javaid</property>
    <property name="connection.password">javapw</property>
    <!-- JDBC connection pool (use the built-in) -->
    <property name="connection.pool_size">5</property>
    <!-- SQL dialect -->
    <property name="dialect">org.hibernate.dialect.MySQLInnoDBDialect
    </property>
    <!-- Enable Hibernate's automatic session context management -->
    <property name="current_session_context_class">thread</property>
    <!-- Disable the second-level cache -->
    <property name="cache.provider_class">
        org.hibernate.cache.NoCacheProvider</property>
    <!-- Echo all executed SQL to stdout -->
    <property name="show_sql">true</property>
    <!-- Drop and re-create the database schema on startup -->
    <property name="hbm2ddl.auto">create</property>

    <!-- 데이터베이스 테이블 매핑 관련 xml문서 경로지정 -->
    <mapping resource="jeon/com/model/member.hbm.xml"/>
    <mapping resource="jeon/com/model/product.hbm.xml"/>
    <mapping resource="jeon/com/model/purchase.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

3.3.2 스프링을 이용한 비즈니스 계층 개발

[그림 3-22]은 비즈니스 계층에서 사용할 스프링 프레임워크의 아키텍처 구조이고 [표 3-12]은 스프링의 아키텍처 내의 각 패키지에 대한 설명이다.

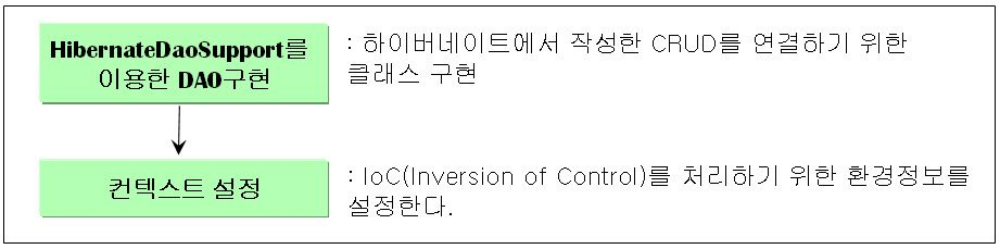


[그림 3-22] 스프링 프레임워크 아키텍처

[표 3-12] 스프링의 기능 요소

패키지 명	기능
Core	스프링의 핵심으로 기능과 그 설정을 분리하기 위한 IoC기능이 구현된 BeanFactory를 제공한다.
Context	스프링의 기본기능. EJB의 JNDI처럼 스프링에서 자바빈의 접근 방법을 제공한다.
DAO	DAO 패키지는 JDBC에 대한 추상화 계층으로 지루한 JDBC 코딩이나 예외처리를 없애고 트랜잭션 관리 기능도 제공한다.
ORM	ORM 제품들을 스프링의 기능과 조합해서 사용할 수 있게 한다.
AOP	AOP 얼라이언스와 호환되는 AOP구현 패키지이다.
Web	웹 애플리케이션 개발에 필요한 기본기능 웹워크, 스트럿츠와 통합하기 위한 패키지
WebMVC	스트럿츠와 같은 일반적인 WAF의 기능을 스프링 버전으로 구현하고 있는 패키지이다. 필수사항은 아니다.

스프링은 IoC와 DI(Dependency Injection)를 지원하는 경량 컨테이너이다. AOP를 기반으로 트랜잭션을 지원하고 POJO를 지원하기 때문에 널리 사용된다. 스프링 프레임워크를 사용할 때는 모든 패키지를 사용하여 개발할 수 있지만 본 연구에서는 스프링은 비즈니스 계층에서만 담당하기 때문에 DBMS의 접근은 ORM 프레임워크를 사용하고 사용자 UI는 Servlet/JSP를 이용하여 구현한다. 그래서 본 논문에서는 스프링의 Core에 해당하는 IoC와 Context, AOP에 대한 패키지만 커스터마이징 하여 비즈니스 계층을 개발한다.



[그림 3-23] 스프링의 개발 순서

스프링을 이용한 비즈니스 계층 개발은 다음과 같이 작성한다.

- HibernateDaoSupport를 이용한 DAO구현-MemberHibernateDAO.java
- 컨텍스트 설정 - applicationContext_dao.xml, applicationContext_domain.xml

[표 3-13] 퍼시스턴스 계층의 빈 설정

applicationContext_dao.xml
.....[생략].....
<beans>
<bean id="memberTarget" **class="com.jeon.dao.MemberHibernateDAO"**>
<property name="sessionFactory" ref="sessionFactory" />
</bean>
</beans>

[표 3-14] HibernateDaoSupport를 이용한 DAO구현

MemberHibernateDAO.java

```
.....[생략].....
public class MemberHibernateDAO extends HibernateDaoSupport
implements UserDAO {
//회원 로그인
public Member loginMember(String id, String pw)
      throws DataAccessException {
      Member login =
(Member)getHibernateTemplate().get(Member.class,id,pw);
      return login;
}
//등록된 제품 리스트
public Collection listProduct()throws DataAccessException {
   return getHibernateTemplate().find("from Product");
}
//제품 구매하기
public boolean addPurchase(PurchaseRec pRec)throws DataAccessException
{
   getHibernateTemplate().save(pRec);
   return 1;
}
//고객이 구매한 제품 정보 보기
public Collection listPurchase(String id)throws DataAccessException {
   return  getHibernateTemplate().get(Purchase.class, id);
}
.....[생략].....
```

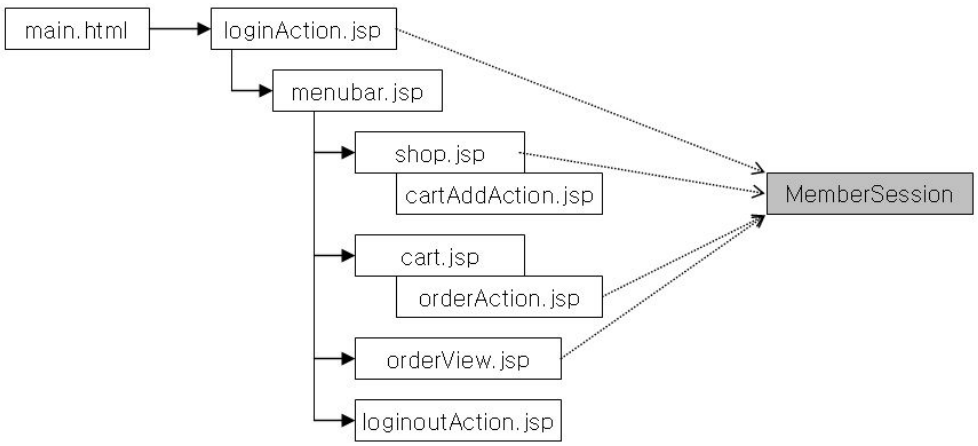
스프링에서 CRUD프로그래밍을 위해 환경설정은 applicationContext_dao.xml에 환경설정하고 DAO구현은 HibernateDaoSupport를 상속받아 환경정보(getHibernateTemplate())를 읽어 생명주기 메서드를 호출한다. 스프링 컨테이너가 유용한 이유 중에 하나는 IoC을 제공하기 때문이다. [표 3-15]는 XML환경 설정으로 도메인 모델을 조립하는 정보이다.

[표 3-15] IoC를 이용한 도메인 조립

applicationContext_domain.xml
.....[생략].....
<!--객체 조립 설정-->
<aop:config>
<aop:pointcut id="requiredTx"
expression="execution(*com.jeon.domain.MemberIDFacade.*(..))" />
<aop:advisor advice-ref="txAdvice" pointcut-ref="requiredTx" />
.....[생략].....
</aop:config>
.....[생략].....

3.4 Servlet/JSP를 이용한 프리젠테이션 계층 개발

프리젠테이션은 테스트를 위해 MVC Model 2 아키텍처를 기반으로 [그림 3-24]와 같이 페이지 설계를 한다. 상용 소프트웨어와 오픈소스 소프트웨어에 대한 개발환경에서 모두 프리젠테이션 계층에서 Servlet/JSP를 특별한 설정 없이 사용가능하다. 다만 자바빈(MemberSession.java)에서 UI에서 사용하는 메서드명은 같고 세션빈 컴포넌트와 스프링 프레임워크를 연결하는 구현부만 정의한다.



[그림 3-24] ShopWAR 컴포넌트의 페이지 설계

상용 프레임워크를 기반으로 개발한 소프트웨어와 오픈소스 프레임워크를 기반으로 개발한 소프트웨어를 테스트 하기위해 UI를 공통으로 작성함으로 각 컴포넌트를 연결하는 자바빈은 loginMember(String id, String pw)와 같이 특정 업무에 대한 동작을 정의할 때는 같은 인터페이스 명을 사용한다. 단 구현부는 [표 3-16]과 [표3-17]과 같이 각각의 프레임워크를 접근하는 코드가 구현되어 있어야 한다.

[표 3-16] 상용 프레임워크를 접근하기 위한 자바빈

MemberSession.java

```
.....[생략].....
/**MemberBMP lookup*/
private MemberSessionHome memHome(){
  MemberSessionHome memHome=null;
    try{
      Object obj= ctx.lookup("MemberSessionEJB");
      memHome = (MemberSessionHome)
PortableRemoteObject.narrow(obj, MemberSessionHome.class);
    }catch(Exception e){
      System.out.println("Member 엔티티 빈과 연결 부분에서 오류 발생");
    }
    return memHome;
}
/**고객 로그인*/
  public boolean loginMember(String id, String pw) {
    try{
      MemberSession memRemote=memHome().create();
      if(memRemote.loginMember(id, pw) )
        return true;
    }catch(Exception e){
      System.out.println(" MemberSession  loginMember().....");
    }
    return false;
  }
.....[생략].....
```

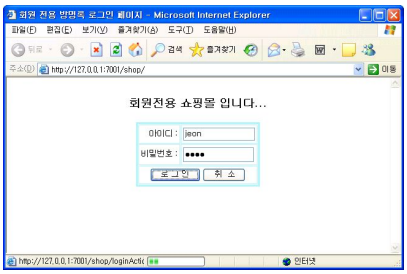
[표 3-17] 오픈소스 프레임워크를 접근하기 위한 자바빈

MemberSession.java

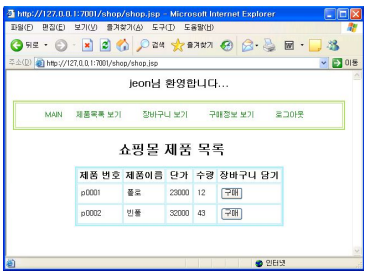
```
import org.springframework.web.context.WebApplicationContext;
import org.springframework.web.context.support.WebApplicationContextUtils;
.....[생략].....
public static MemberBiz getMemberBizrService(ServletContext ctx){
   WebApplicationContext wac =

WebApplicationContextUtils.getRequiredWebApplicationContext(ctx);
   return (MemberBiz) wac.getBean("memberService");
}
/**고객 로그인*/
   public boolean loginMember(String id, String pw) {
      MemberBiz biz=this.getMemberBizrService(ctx)
      try{

         if(biz.loginMember(id, pw) )
            return true;
      }catch(Exception e){
         System.out.println(" MemberBiz  loginMember().....");
      }
      return false;
   }
.....[생략].....
```

3.5 실행 결과 및 평가

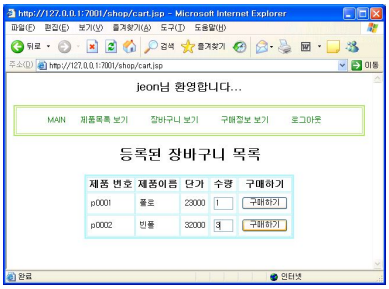
본 연구에서는 도서 쇼핑몰에 대한 요구사항 중 아키텍처를 설계하고 설계된 컴포넌트중 회원가입, 도서구매, 구매확인 컴포넌트를 상세 설계하여 상용 프레임워크와 오픈소스 프레임워크를 기반으로 각각 개발하였다. 공통 UI를 구현하여 상용 프레임워크와 오픈소스 프레임워크에서 요구사항에 맞는 실행 결과를 다음과 같이 확인할 수 있었다.



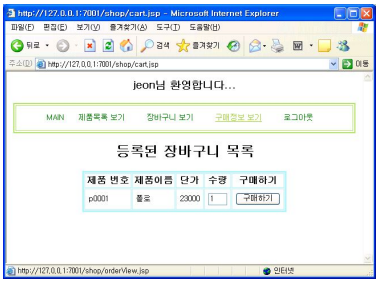
[그림 3-25] 로그인



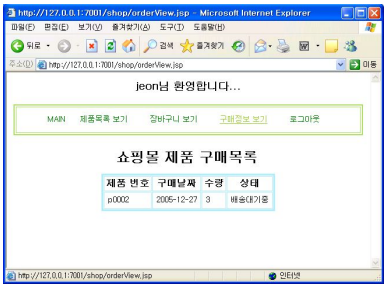
[그림 3-26] 쇼핑몰 목록보기



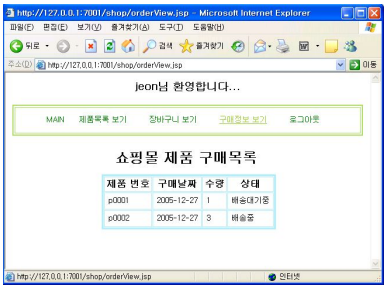
[그림 3-27] 장바구니



[그림 3-28] 장바구니 처리 결과



[그림 3-29] 구매 목록



[그림 3-30] 배송 상태 확인

도서 쇼핑몰 구축을 개발한 결과 오픈소스 프레임워크를 이용한 개발이 상용 프레임워크를 이용한 개발 절차보다 훨씬 간단하고 효율적이라는 것을 확인할 수 있었다. 개발 평가 항목은 [표 3-18]에서 확인할 수 있다.

[표 3-18] 개발 평가

비교항목	상용 프레임워크	오픈소스 프레임워크
비용	유료	무료
이식성	WAS에 종속된 환경설정	WAS에 독립된 환경설정
개발방법론	마르미III 기반	
아키텍처 설계	J2EE 아키텍처 기반 프레임워크 배치	
계층 개발절차	인테그레이션 계층-->비즈니스 계층-->프리젠테이션 계층	
업무 개발절차	로그인(24단계)	로그인(12단계)

본 연구에서의 사례연구에서 살펴볼 수 있듯이 마르미III를 응용한 개발 프로세스와 J2EE 아키텍처를 기반으로 프레임워크를 배치한 설계방식은 상용 프레임워크를 이용한 개발 뿐 아니라 오픈소스 프레임워크를 이용한 개발에서도 적용할 수 있어 개발 지침을 제시할 수 있었다.

그러나 오픈소스 프레임워크는 관련 소프트웨어를 무상으로 사용할 수 있고 환경설정을 서버에 종속되지 않게 관련 라이브러리를 hibernate3.jar 등과 같은 jar파일로 배포하여 lib/디렉토리에서만 유지하면 어느 서버에서나 해석이 가능하여 이식성이 좋다. 또한 개발 절차도 [그림 3-7], [그림 3-18]에서 살펴볼 수 있듯이 로그인 업무를 처리할 때 24단계에서 12단계를 줄일 수 있어 훨씬 간단해지고 효율적임을 확인할 수 있다. 평균 개발단계는 1/2로 감소함을 확인할 수 있다.

제 4 장 결론 및 향후 연구과제

본 연구에서는 상용 프레임워크와 오픈소스 프레임워크를 기반으로 회원 전용 도서쇼핑몰을 개발하였다. 개발된 소프트웨어에서 확인할 수 있듯이 상용 프레임워크를 기반으로 작성된 소프트웨어를 오픈소스 프레임워크를 기반으로 요구사항을 구현하는데 전혀 문제가 없었다. 오히려 상용 프레임워크를 기반으로 소프트웨어를 작성하면 환경 설정 및 코딩이 일부 서버에 종속되기 때문에 이식성이 매우 떨어지지만 오픈소스 프레임워크를 기반으로 작성된 소프트웨어는 소스 코드도 단조로워지고 서버에 종속하지 않는 환경 설정으로 이식성이 매우 좋았다.

이러한 연구로 오픈소스 프레임워크를 기반으로 소프트웨어를 개발하는데 대한 인식의 전환의 계기가 되었고 향후 오픈소스 프레임워크를 이용한 소프트웨어 개발 시 활용 가능한 절차 및 지침을 제시할 수 있었다.

향후 연구 과제는 오픈소스 프레임워크의 표준화에 대한 연구와 각 개발 계층에 사용할 프레임워크의 선정에 대한 세부 프로세스 및 프레임워크의 평가 기준에 대한 연구가 필요하다.

참고문헌

[1] Martin Fink, The Business and Economics of Linux and Open Source, Prentice Hall PTR, 2002.

[2] Michel Ruffin and Christof Ebert, “Using Open Source Software in Product Development: A Primer”, IEEE Software, Vol.21,no.1,pp82~86, 2004.

[3] OSD, http://www.opensource.org/osd.html

[4] Deepak Alur, Dan Malks, John Crupi Core J2EE Patterns, Prentice Hall 2003.

[5] 소스포지, http://sourceforge.net/search/?type_of_search=soft&type_of_search=soft&words=java

[6] 다우기술, http://support.daou.co.kr/sol_dir/sol_view.jsp?s_seq=2

[7] Rod Johnson, Expert One-on-One J2EE Design and Development, WROX PRESS, 2003.

[8] 박재성, Spring 프레임워크 워크북, 한빛미디어, 2006.

[9] TOGAF, http://www.opengroup.org/togaf/, 2004.

[10] 한국전자통신연구원, 마르미III Ver. 4.0, 2002.

[11] 박현준, 오픈소스 퍼시스턴스 프레임워크 비교, 국민대학교 BIT 전문대학원, 2004.

[12] 최범균, 하이버네이트 3 프로그래밍, 가메출판사, 2007.

[13] Paul Clements , Rick Kazman , Len Bass , Software Architecture in Practice (2/E), Addison-Wesley Professional, 2003.

[14] JOHN VLISSIDES , Erich Gamma , Ralph Johnson , Richard Helm, Design Patterns: Elements of Reusable Object-Oriented Software , Addison-Wesley Professional, 1995.

[15] 전혜영, 김성진, 클릭하세요 웹로직으로 배우는 EJB, 대림, 2006.