한국산학기술학회논문지
Vol. 12, No. 4 pp. 1919-1926, 2011

경량 컨테이너 구조 환경에서 하이버네이트 3.2와 아이바티스 2.3의 개발 생산성 비교 연구

이명호[1*]
[1]세명대학교 전자상거래학과

A Study on Comparison of Development Productivity of Hibernate 3.2 and iBatis 2.3 Based Lightweight Container Architecture

Myeong-Ho Lee[1*]
[1]**Department of eCommerce, Semyung University**

요 약 본 논문은 스프링 프레임워크 2.5의 동일한 플랫폼 환경에서 하이버네이트 3.2와 아이바티스 2.3과 연관된 객체지향 소프트웨어 개발에 대한 지침과 평가 지표를 제공하는데 목적이 있다. 현재까지 경량 컨테이너 구조로 많이 사용되고 잘 알려진 구조로 스프링 프레임워크가 있다. 또한 데이터베이스의 생산성을 높여주기 위한 기법으로 ORM이 있다. 현재 많이 사용되는 ORM 도구로 하이버네이트와 아이바티스가 있다. 따라서 본 연구에서는 가장 큰 특징과 변화를 가지고 있으며 안정된 스프링 프레임워크 2.5의 동일프레임워크 환경을 기반으로 하이버네이트 3.2와 아이바티스 2.3에서 파일럿 시스템을 설계하고 구현함으로써 개발 플랫폼 환경별 객관적인 소프트웨어 개발 생산성을 비교하고, 표준화에 따른 평가 지표를 제공하고자 한다.

Abstract This paper proposes an object-oriented software development guidance and an evaluation index for the productivity related to Hibernate 3.2 and iBatis 2.3 in same platform of Spring framework 2.5. Currently in production until the lightweight container architecture, known most commonly used architecture framework is Spring framework. Also intended to increase the productivity of database techniques are ORM. Hibernate and iBatis is an ORM tool is currently being used. In this study, Spring framework 2.5 is based on the framework of the same Hibernate 3.2 and iBatis 2.3 to design and implement the pilot system. In addition, comparison and standardization of software development productivity assessment is to provide guidance.

Key Words : Spring Framework 2.5, ORM, Hibernate 3.2, iBatis 2.3, Software Development Productivity

1. 서 론

인터넷의 급속한 확산과 웹 2.0 진화에 따른 정보기술 환경의 확장 요구에 부응하기 위하여 사용자가 언제 어디서나 인터넷 접속을 통하여 정보기술의 자원을 제공받는 주문형 IT 서비스인 클라우드 컴퓨팅(Cloud Computing) 서비스 시대가 도래하고 있다. 앞으로 3G/LTE 이동통신이나 무선랜 등의 무선통신 인프라 구축과 스마트폰, 태블릿 PC와 같은 모바일 인터넷 기기의 확산으로 모바일 앱(Mobile App)이나 모바일 웹(Mobile Web) 환경을 기반으로 하는 모든 정보기술 산업에서의 모바일화가 급속히 전개되고 있다. 기업의 업무용 모바일 오피스(Mobile Office)나 모바일 정보처리 서비스가 개인의 활발한 콘텐츠 생성과 자유로운 개인화 웹 환경을 제공하는 개인화 서비스가 매우 활성화될 전망이다. 따라서 플랫폼 종속적인 환경에서 개방형 플랫폼 개발 환경으로

본 연구는 2010년도 세명대학교 자체연구비 지원에 의해 연구되었음.
*교신저자 : 이명호(mhlee@semyung.ac.kr)
접수일 11년 02월 22일 수정일 11년 03월 15일 게재확정일 11년 04월 07일

의 확산과 표준화가 연계되는 성장으로 발전될 전망이다[1-4].

이러한 디지털 공간의 활동과 모바일 기기의 사용 확대로 개인과 조직의 활동 기록이 축적되면서 경영에 유용한 정보도 폭발적으로 증가하게 되었다. 웹사이트의 방문기록, 온라인 서비스의 이용기록, 검색사이트의 검색통계, 소셜미디어의 소통기록 등의 막대한 대용량 데이터가 발생하고 있다.

이러한 엔터프라이즈 환경에서는 이 기종 컴퓨터들 간에 프로그램을 분산시켜 부하를 줄여 시스템의 성능 저하와 네트워크 병목 현상을 줄일 수 있는 분산객체 구조가 필요하게 되었으며 대용량 데이터 처리에 필요한 분산객체의 성공모델로 먼저 알려진 것이 EJB이다. 그러나 EJB의 단점은 분산 환경을 지원하기 위하여 객체를 직렬화하는 과정 때문에 실행 속도의 저하가 발생하며, 개발 주기가 소스수정, 빌드, 배포, 테스트와 같은 복잡한 과정을 거치기 때문에 개발 생산성의 저하가 일어나며, 테스트의 어려움으로 제품의 품질저하, 변형된 패턴들로 인한 객체 지향적으로 개발하는데 제약사항도 발생하며, 대형 벤더사들의 EJB 컨테이너 사이의 이식성 저하 등이 발생한다. 이러한 Non EJB와 EJB 구조가 가지고 있는 문제점을 해결하고 장점들을 지원하기 위하여 새롭게 등장한 구조가 경량 컨테이너 구조(Lightweight Container Architecture)이다[5,6]. 이와 같이 경량 컨테이너 구조의 가장 중요한 6가지 기본 핵심가치로는 아키텍처 리팩토링에 의해서 확장할 수 있는 단순한 아키텍처 구성, 소프트웨어 개발 생산성 확보, 객체지향 중심적, 비즈니스 요구사항의 중요성, 기술과 아키텍처의 검증과정의 중요성, 그리고 테스트 가능성 등의 지향점을 추구하기 위한 결과물로 등장한 것이 스프링 프레임워크(Spring Framework)이다[2-4,7]. 또한 대용량 엔터프라이즈 웹 애플리케이션에서는 데이터베이스가 필수적이며 SQL 코드를 작성하는데 소요시간을 줄여 생산성을 높이는 것이 아주 중요한 문제로 대두 되었다. 이러한 문제를 해결할 수 있는 방법으로 가장 널리 사용되는 기법은 데이터베이스 테이블과 객체사이의 매핑을 자동으로 처리해 주는 ORM(Object Relational Mapping) 기법이다. 현재 ORM 도구로 널리 이용되고 있는 도구로는 Hibernate, iBatis, Toplink, JPA(Java Persistence API), OJB(ObJect relational Bridge) 등이 있다[8,9].

따라서 본 연구에서는 가장 큰 특징과 변화를 가지고 있으며 안정된 스프링 프레임워크 2.5에서 개발된 파일럿 시스템을 동일한 개발 프레임워크 환경을 기반으로 하이버네이트 3.2와 아이바티스 2.3에서 설계하고 구현함으로써 개발 플랫폼 환경별 객관적인 소프트웨어 개발 생산성을 비교하고, 표준화에 따른 평가 지표를 제공하고자 한다.

2. 개발 환경의 기본 개념

2.1 경량 컨테이너 구조의 고찰

경량 컨테이너 아키텍처의 가장 잘 알려진 구조로는 스프링 프레임워크이며 현재까지 표 1과 같은 일정으로 발표되었다.

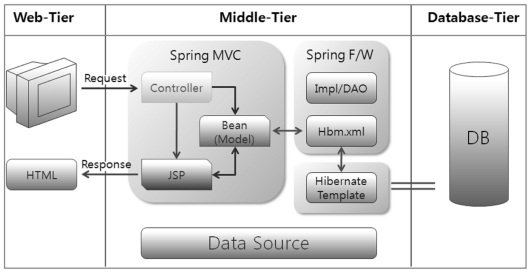
[표 1] 스프링 프레임워크의 발표 일정표

일 정	내 용
2002년 10월	Rod Johnson이 처음 소개
2003년 06월	Apache 2.0 라이센스로 릴리즈
2003년 06월	스프링 프레임워크 1.0 릴리즈
2006년 10월	스프링 프레임워크 2.0 릴리즈
2007년 11월	스프링 프레임워크 2.5 릴리즈
2008년 12월	스프링 프레임워크 3.0 M1 발표
2009년 01월	스프링 프레임워크 3.0 M2 발표
2009년 05월	스프링 프레임워크 3.0 M3 발표
2009년 08월	스프링 프레임워크 3.0 M4 발표

스프링 프레임워크의 구조는 IoC와 의존성 삽입 기능과 빈 팩토리가 존재하는 스프링에서 가장 핵심적인 부분인 Core 패키지, Core 패키지에서 제공되는 핵심적인 기본 위에서 구축되고 Beans 패키지의 특성을 그대로 상속받아 국제화, 이벤트 처리, 리소스 로딩, 그리고 투명한 콘텍스트 생성 등의 기능을 포함하는 Context 패키지, JDBC 추상화를 지원하고 선언적 트랜잭션 관리와 POJO 지원과 특정 인터페이스에 대해서 구현한 클래스 제공하는 DAO 패키지, JPA, JDO, Hibernate, 그리고 iBatis 같은 OR Mapping API 연동을 위한 기능 제공하는 ORM 패키지, Aspect- Oriented Programming에 호환되는 AOP의 지원과 소스 레벨의 메타데이터 기능을 사용하여 코드에 다양한 기능을 부여할 수 있는 AOP 패키지, 기본적인 웹 관련 기능을 제공과 파일업로드, IoC 초기화, Struts 1과 Struts 2와 통합 가능한 Web 패키지, 그리고 MVC 모델에 맞게 구현되는 MVC 패키지로 구성된다[10-12].

2.2. 하이버네이트의 구성

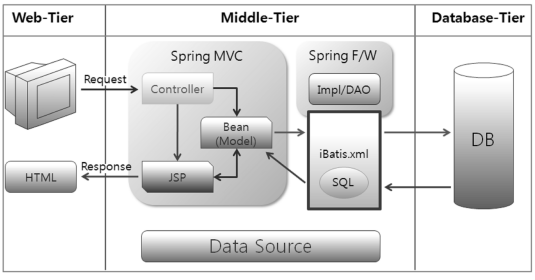
하이버네이트는 자바 환경에서 ORM을 하는 도구이다. ORM은 객체지향 개념을 가진 실체 클래스를 RDB에 있는 테이블로 대응시키는 것을 말한다[7,8]. 가상의 객체지향 DB를 효과적으로 만들어 RDB를 OOP 언어의 개념으로 연계하는 프로그램 기술이다. 그림 1은 본 연구에서 개발 플랫폼상의 하이버네이트 구성도를 도식화 한 것이다.



[그림 1] 하이버네이트의 구성도

2.3 아이바티스의 구성

아이바티스는 SqlMaps라는 별명을 가지고 있으며 SQL과 자바 객체를 매핑 기능을 수행하는 툴이다. 그러나 객체의 상속이나 다양한 컬렉션 클래스의 지원 등의 많은 부분에서 풍부한 객체 모델링을 제공하고 있지는 못하다[13]. 따라서 아이바티스를 이용하기 위해서는 DB로부터 쿼리한 결과를 전달하기 위한 매개체가 필요하며 매개체로 Map이나 유저가 정의한 클래스(객체)가 이용된다. 그림 2는 본 연구에서 개발 플랫폼상의 아이바티스 구성도를 도식화 한 것이다.



[그림 2] 아이바티스의 구성도

3. 개발 생산성 비교 방안

3.1 테스트 환경

본 연구에서는 하이버네이트 3.2와 아이바티스 2,3 사양을 기반으로 하는 소프트웨어 개발 생산성을 비교 분석하기 위한 방안으로 표 2와 같은 동일한 개발환경과 데이터베이스 스키마를 이용하여 스프링 프레임워크 2.5 환경에서의 파일럿 프로그램을 개발한 후, 이 프로그램 통하여 각 항목별 평가 지표를 이용하여 사양별 정량적으로 개발 생산성을 비교하도록 한다.

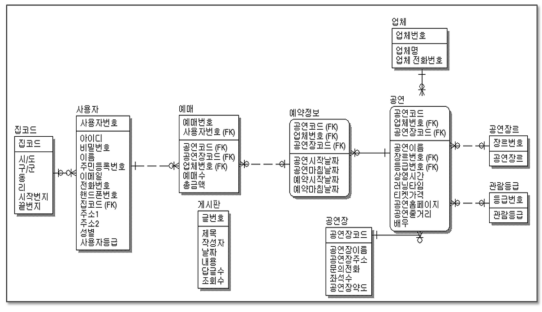
[표 2] 파일럿 테스트의 개발 환경

항 목	Hibernate 3.2	iBatis 2.3
OS	Windows Vista	Windows Vista
Framework	Spring F/W	Spring F/W
Platform	JavaSE6.0/JavaEE1.5	JavaSE6.0/JavaEE1.5
WAS	JBoss-5.0.0GA	JBoss-5.0.0GA
DB	Oracle 10g XE	Oracle 10g XE
IDE	Eclipse 3.4	Eclipse 3.4
CASE	Rational Rose 2006	Rational Rose 2006

따라서 본 연구에서 사용한 정량적 소프트웨어 LoC의 평가와 사양별 XML의 비교 평가 그리고 개발방식의 평가 등을 선정하여 분석한다[14].

3.2 데이터베이스 스키마

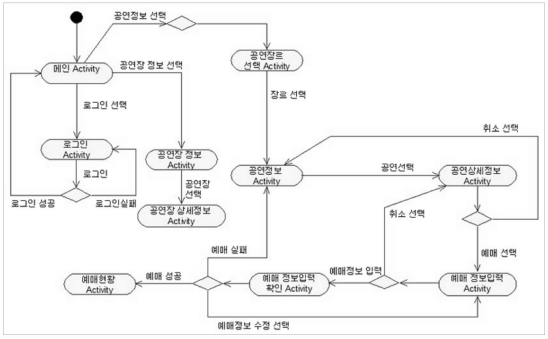
동일한 경량 컨테이너 구조 환경에서 소프트웨어 생산성 비교를 위하여 개발될 파일럿 시스템의 데이터베이스 스키마는 스프링 프레임워크 2.5 사양에서 그림 3과 같이 동일한 데이터베이스 스키마를 이용하여 비교 분석한다.



[그림 3] 데이터베이스 스키마의 구성도

3.3 화면흐름 모델

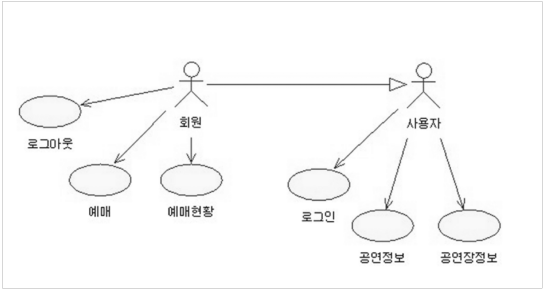
그림 4는 파일럿 시스템의 전체 화면 흐름 모델을 액티비티 다이어그램으로 나타낸 것이다.



[그림 4] 개발 환경의 화면 흐름 모델

3.4 유스케이스 다이어그램

파일럿 시스템의 요구사항 정의 활동에서 파악된 액터와 유스케이스를 유스케이스 다이어그램으로 표현해 보면 그림 5와 같은 유스케이스 모델이 된다.



[그림 5] 전체 유스케이스 다이어그램

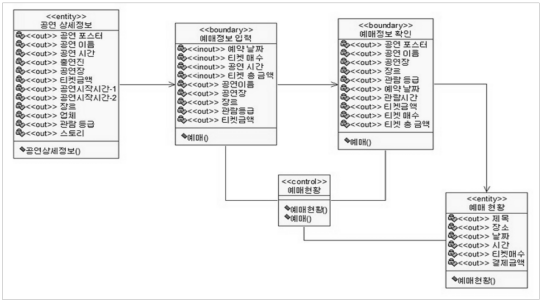
파일럿 시스템의 유스케이스 모델에 기술된 유스케이스 이름만으로는 유스케이스가 나타내려고 하는 기능을 명확하게 정의될 수 없기 때문에 이에 대한 보다 자세한 정보를 기술한 문서 산출물인 공연장 관리 유스케이스 명세서를 기술해 보면 표 3과 같다. 유스케이스 명세서에 기술된 이벤트 흐름은 유스케이스가 나타내는 시스템의 기능이 액터와 시스템 간의 상호작용으로 진행되는 과정을 보여준다.

3.5 클래스 다이어그램

유스케이스를 분석하여 경계클래스, 제어클래스, 그리고 실체클래스를 파악하는 유스케이스 분석기법이 있다. 이들 간의 관계를 클래스 다이어그램으로 표시하면 분석객체 모델이 된다[8]. 따라서 본 연구의 파일럿 시스템에서 중요한 예매관리의 분석객체 모델인 클래스 다이어그램은 그림 6과 같다.

[표 3] 공연장 관리 유스케이스 명세서

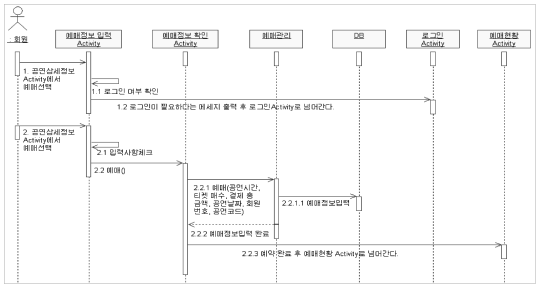
개 요	사용자는 공연장을 시스템에 등록, 조회, 수정, 삭제한다.
관련액터	사용자(운영자, 관리자)
관련요구사항	
우선순위	상
선행조건	사용자는 시스템에 로그인 되어 있어야 한다.
이벤트 흐름	기본 흐름 1.사용자는 메인화면에서 마이페이지 항목을 선택한다. 2.시스템은 마이페이지 메인 화면을 보여준다. 3.사용자는 공연장 관리 항목을 선택한다. 4.시스템은 공연장 관리 메인화면을 보여준다. 5.사용자는 공연장 관리 메인화면을 이용하여 공연장 정보를 등록, 검색, 조회, 수정, 삭제할 수 있다.
	등록 1.사용자는 공연장 관리 메인 화면에서 등록 버튼을 클릭한다. 2.시스템은 공연장 등록 화면을 보여준다. 3.사용자는 공연장 정보를 입력하고 등록을 선택한다.(A1) ■ 공연장 정보는 이름, 주소, 약도, 공연장 주요 사진, 공연장 소개, 전화번호, 좌석수, 좌석배치도, 공연 기간 등으로 구성된다. 4.시스템은 입력된 공연장 정보를 저장한다. 5.시스템은 등록된 공연장 정보를 확인할 수 있게 공연장 조회 화면을 보여준다. [대안 흐름] (A1) 사용자가 입력항목을 전부 입력하지 않고 등록 버튼을 클릭할 경우 1.시스템은 입력오류 메시지를 보여주고 해당 입력항목으로 돌아간다. 2.사용자는 필수 입력항목을 재입력하고 등록 버튼을 클릭한다.
	검색 1.사용자는 공연장 관리 메인 화면에서 공연장을 검색 조건으로 입력하고 검색을 선택한다. 2.시스템은 입력된 공연장 이름에 해당하는 공연장의 정보를 구하여 공연장 관리 메인 화면에 표시한다. ■ 입력된 이름을 포함하는 공연장들만 검색된다. “%공연장%”으로 검색을 수행한다. ■ 공연장 관리 메인 화면에 표시되는 공연장 정보는 공연장 이름, 주소, 전화번호이다. ■ 사용자는 검색된 공연장의 이름을 선택하여 공연장에 대한 자세한 정보를 조회할 수 있다.
	조회 1.사용자는 공연장 관리 메인 화면에서 검색된 공연장 표시 목록에서 공연장 이름을 선택한다. 2.시스템은 해당 공연장의 상세 정보를 공연장 조회 화면에 표시한다. ■ 공연장 조회 화면에 표시된 공연장 정보는 이름, 주소, 약도, 공연장 주요 사진, 공연장 소개, 전화번호, 좌석수, 좌석배치도, 공연 기간 등이다.
	수정 1.사용자는 수정할 공연장을 클릭한다. 2.시스템은 수정화면을 보여주며 수정할 공연장에 대한 정보를 표시한다. ■ 수정이 가능한 정보는 공연장 이름, 주소, 약도, 공연장 주요 사진, 공연장 소개, 전화번호, 좌석수, 좌석배치도, 공연 시간, 공연 기간, 티켓 가격, 관람시간, 관람등급 이다. 3.사용자는 새로운 공연장 정보를 입력하고 수정 완료 선택한다. 4.시스템은 입력받은 새 공연장 정보로 기존의 정보를 대체한다. 5.시스템은 변경된 공연장 정보를 확인할 수 있게 공연장 조회 화면을 보여준다.
	삭제 1.사용자는 공연장 조회 화면에 표시된 공연장에 대한 정보를 삭제하기 위하여 삭제를 선택한다. 2.시스템은 해당 공연장에 대한 정보를 삭제하고(A1), 해당 공연장에 대한 정보가 삭제되었다는 메시지를 보여준다. [대안 흐름] (A1)공연중이거나 예매 진행중인 공연장인 경우 1.공연중이기 때문에 삭제가 불가능하다는 메시지를 보여준다.
후행조건	요청된 공연장 정보에 대한 등록/삭제/수정이 완료되었다.
기타요구사항	최대 1,000건 정도의 공연장에 대한 정보를 처리할 수 있어야 한다.



[그림 6] 예매관리의 분석객체 모델

3.6 시퀀스 다이어그램

시퀀스 다이어그램을 사용하여 분석 객체들 간의 메시지 송수신을 통한 유스케이스 실현 과정이 분석 유스케이스 실현 모델이다[8]. 그림 7은 본 연구의 파일럿 시스템에서 중요한 예매관리의 분석 유스케이스 실현 모델인 시퀀스 다이어그램을 도식화 한 것이다.



[그림 7] 예매관리의 분석 유스케이스 모델

3.7 파일럿 시스템의 구현

이상과 같은 데이터베이스 스키마를 기반으로 분석 및 설계를 통하여 동일한 스프링 프레임워크 2.5 플랫폼 개발 환경에서 하이버네이트 3.2와 아이바티스 2.3의 파일럿 시스템을 구현하기 위한 메인 화면은 그림 8 및 그림 9와 같다.

[그림 8] 하이버네이트 3.2의 파일럿 시스템

[그림 9] 아이바티스 2.3의 파일럿 시스템

4. ORM의 비교 평가

4.1 LoC의 평가

LoC의 비교 선행 조건으로는 모든 공백 라인과 주석은 삭제하였으며, 클래스나 메서드의 오픈 브레이스는 해당 라인 끝에 클로즈 브레이스는 한라인에 정렬하였다. 매개변수는 개수와 상관없이 한 라인에 나열하였다. 표 4는 스프링 프레임워크 2.5 환경에서 하이버네이트 3.2와 아비바티스 2.3에서 개발된 파일럿 시스템의 LoC 현황을 나타낸 것이다.

[표 4] 파일럿 시스템의 LoC 비교

항 목		Hibernate 3.2	iBatis 2.3
Board	Board.java	7	7
	BoardImpl.java	59	82
CompInfo	CompInfo.java	4	3
	CompInfoImpl.java	46	48
Genre	Genre.java	3	2
	GenreImpl.java	36	41
GradeShow	GradeShow.java	3	2
	GradeShowImpl.java	36	42
Login	Login.java	✗	6
	LoginImpl.java	✗	29
Member	Member.java	14	12
	MemberImpl.java	139	86
PerfDays	PerfDays.java	10	7
	PerfDaysImpl.java	78	112
PerfInfo	PerfInfo.java	26	23
	PerfInfoImpl.java	100	62
PlaceInfo	PlaceInfo.java	9	7
	PlaceInfoImpl.java	41	52
ReserveInfo	ReserveInfo.java	21	13
	ReserveInfoImpl.java	72	54
Zipcode	Zipcodejava	8	8
	ZipcodeImpl.java	29	14
계		741	712

표 4에서 보는 바와 같이 아이바티스가 하이버네이트

보다 4%의 LoC 감소가 됨을 알 수 있다.

4.2 xml의 평가

하이버네이트는 xml 초기 설정으로 재사용이 가능하고 아이바티스는 데이터를 가져오는 형식마다 매핑정보를 xml에 추가해야 한다. 표 5는 스프링 프레임워크 2.5 환경에서 하이버네이트 3.2와 아이바티스 2.3에서 개발된 파일럿 시스템의 xml 현황 비교를 나타내었다.

[표 5] 파일럿 시스템의 xml 비교

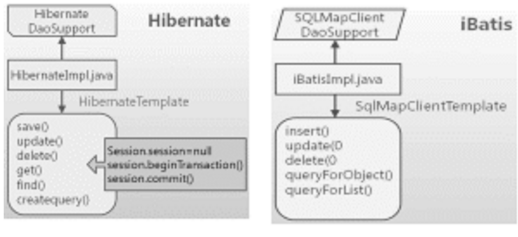
항 목		Hibernate 3.2	iBatis 2.3
Board	Board.xml	19	30
CompInfo	CompInfo.xml	18	32
Genre	Genre.xml	17	25
GradeShow	GradeShow.xml	18	27
Login	Login.xml	Member에 포함	24
Member	Member.xml	38	57
PerfDays	PerfDays.xml	25	64
PerfInfo	PerfInfo.xml	51	208
PlaceInfo	PlaceInfo.xml	28	48
ReserveInfo	ReserveInfo.xml	37	90
Zipcode	Zipcode.xml	21	22
계		272	627

표 5에서 보는 바와 같이 xml의 LoC 비교를 보면 하이버네이트가 아이바티스보다 57%의 감소가 됨을 알 수 있다.

4.3 개발방식의 평가

(1) 클래스의 비교

개발방식의 차이점으로 클래스 부분을 비교해 보면 그림 10과 같이 하이버네이트는 xml 정보를 사용하여 DB 테이블이 아닌 자바 클래스로 호출한다. 반면에 아이바티스는 xml 쿼리를 이용하여 쿼리를 보내어 가져온 정보를 매핑을 통하여 처리하는 것을 알 수 있다.



[그림 10] ORM별 클래스 비교

(2) 단일 테이블의 비교

개발방식의 차이점으로 단일 테이블 소스 부분에서 게시판 관련 BoardImpl.java을 비교해 보면 표 6과 같이 하이버네이트와 아이바티스의 개발방식의 차이를 요약하여 볼 수 있다.

[표 6] 게시판관련 개발방식의 비교표

메서드	Hibernate 3.2	iBatis 2.3
Template	getHibernateTemplate()	getSqlMapClientTemplate()
add()	save(board); : void	insert("insertBoard", board); :void
modify()	update(board); : void	update("updateBoard", board); :int
remove()	delete(board); : void	delete("deleteBoard", no); :int
get()	get(board); : object	queryForObject("getBoardForm"); :object
searchAll()	find("from Board order by board_no");	queryForList("getBoardList", null);
search()	session.createQuery("from Board where writer like=?").setParameter(value, object).list();	queryForList("getBoardListById", writer);

단일 테이블의 xml 소스 부분에 대하여 하이버네이트(Board.hbm.xml)와 아이바티스(Board.xml)의 개발방식의 차이를 살펴보면 그림 11과 그림 12와 같다.

[그림 11] 하이버네이트의 Board.hbm.xml

[그림 12] 아이바티스의 Board.xml

(3) POJO 클래스의 비교

POJO 클래스 부분의 개발방식을 비교해 보면 하이버네이트는 객체 안에 객체가 존재하는 구조로 이루어져 있으며 아이바티스는 다른 테이블에서 데이터를 가져올 때마다 변수를 직접 생성하는 방식을 취하고 있다.

따라서 하이버네이트의 경우 객체 매핑 테이블 정의 등과 같이 초기설정의 어려움이 존재하지만 한 번의 초기 설정 후에는 사용이 편리하다. LoC 비교시와 같이 초기 설정 후에 HQL에서 Query를 자동 처리하기 때문에 추가 설정이 적기 때문에 유지보수가 비교적 용이함을 알 수 있다. 아이바티스는 기존의 개발 방식을 가진 사람들에게는 편리하지만 프로젝트 규모가 커지고 DB 테이블이 더 많아진다면 소스 코드가 길어지는 단점이 있다. 객체 지향적 관점과 유지보수 측면에서는 하이버네이트가 편리하고 하이버네이트의 지식이 부족하고 SQL 작성에 자신이 있다면 아이바티스를 선택하는 것이 효율적이다.

5. 결 론

Non EJB와 EJB 구조가 가지고 있는 중량 컨테이너 구조의 문제점을 해결하고 장점들을 지원하기 위하여 새롭게 등장한 구조가 경량 컨테이너 구조이다. 경량 컨테이너 구조의 핵심인 단순한 아키텍처 구성, 소프트웨어 개발 생산성 확보, 객체지향 중심적, 비즈니스 요구사항의 중요성, 기술과 아키텍처의 검증과정의 중요성, 그리고 테스트 가능성 등의 지향점을 추구하기 위한 결과물로 등장한 것이 스프링 프레임워크이다. 대용량 엔터프라이즈 웹 애플리케이션에서는 데이터베이스의 생산성을 높이는 문제를 해결할 수 있는 방법으로 가장 널리 사용되는 기법은 데이터베이스 테이블과 객체사이의 매핑을 자동으로 처리해 주는 ORM 기법이다. 이중에서 가장 널리 이용되고 있는 솔루션이 하이버네이터와 아이바티스가 있다.

그러나 현재까지 경량 컨테이너 아키텍처의 성공 모델로 알려진 스프링 프레임워크 환경에서 ORM 솔루션별 정량적인 성과지표 개발 및 사례의 부족으로 실무 프로젝트의 업그레이드나 새로운 기술 사양의 적용이 미비하였다. 그 이유는 스프링 프레임워크의 기술 변화 및 다양한 ORM의 적용 기술 미비로 쉽게 새로운 사양들을 현업에 적용하지 못한 것이다. 또한 스프링 프레임워크 환경에서 ORM 별로 소프트웨어 개발 생산성 비교나 수행 속도 평가 연구도 부족하여 소프트웨어 생산성의 평가와 프로젝트의 새로운 시도에 제한이 있었다.

따라서 본 연구에서는 가장 큰 특징과 변화를 가지고 있으며 안정된 스프링 프레임워크 2.5에서 개발된 파일럿 시스템을 동일한 개발 프레임워크 환경을 기반으로 하이버네이트 3.2와 아이바티스 2.3에서 설계하고 구현함으로써 개발 플랫폼 환경별 객관적인 소프트웨어 개발 생산성을 비교하고, 표준화에 따른 평가 지표를 제공하였다. 향후에는 동일한 데이터 스키마를 이용하여 모바일 오피스에 대한 개발 생산성 연구나 EJB 3.1과 스프링 프레임워크 3.0의 소프트웨어 생산성 분석 연구가 지속되어야 할 것이다.

참고문헌

[1] 이명호, “EJB2.0과 EJB3.0의 소프트웨어 개발 생산성 비교 연구”, 한국산업경영시스템학회지, 제31권 제3호, pp. 1-7, 9월, 2008.

[2] 이명호, “경량 컨테이너 구조 환경의 스프링 프레임워크 2.5를 기반으로 호텔예약시스템의 설계 및 구현”, 한국산학기술학회논문지, 제10권 제3호, pp. 589-595, 3월, 2009.

[3] 이명호, “동일한 경량 컨테이너 구조 환경의 스프링 프레임워크 2.0과 2.5의 개발 생산성 비교 연구”, 한국산학기술학회논문지, 제10권 제6호, pp. 1265-1274, 6월, 2009.

[4] 이명호, “JPetStore 주문 시스템 기반으로 Spring 2.5와 Seam 2.0의 개발 생산성 비교 연구”, 한국산학기술학회논문지, 제11권 제7호, pp. 2610-2615, 7월, 2010.

[5] R. Johnson, “Expert One-on-One J2EE Design and Development”, Wrox, pp. 441-673, October, 2002.

[6] R. Johnson, and J. Hoeller, “Expert One-on-One J2EE Development without EJB”, Wrox, pp. 1-141, June, 2004.

[7] 이일민, “자바 기술의 미래를 비추는 거울 스프링 프레임워크 2.5”, 마이크로소프트웨어, pp. 136-143, 1월, 2008.

[8] 채홍석, “객체지향 CBD 개발 Bible”, 한빛미디어, pp. 35-76, 8월, 2005.

[9] 최범균, “하이버네이트 3 프로그래밍”, 가메출판사, pp. 14-414, 4월, 2007.

[10] 박재성, “Spring 프레임워크 워크북”, 한빛미디어, pp. 26-377, 1월, 2006.

[11] 최범균, “웹 개발자를 위한 스프링 2.5 프로그래밍”,

가메출판사, pp. 24-440, 3월, 2008.
[12] R. Johnson, et al., “Professional Java Development with the Spring Framework”, Wrox, pp. 1-303, July, 2005.
[13] C. Begin, B. Goodin and L. Meadors, “iBatis in Action”, Manning, pp. 3-302, January, 2007.
[14] J. Steams, R. Chinnici, and Sahoo, “An Introduction to the Java EE 5 Platform, "http://java.sun.com/developer /technicalArticles/J2EE/intro_ee5/index.html”, 2006.

이 명 호(Myeong-Ho Lee) **[종신회원]**

- 1984년 2월 : 아주대학교 산업공학과 (공학사)
- 1986년 2월 : 아주대학교 대학원 산업공학과 (공학석사)
- 2001년 2월 : 아주대학교 대학원 산업공학과 (공학박사)
- 2002년 3월 ~ 현재 : 세명대학교 전자상거래학과 부교수

<관심분야>
물류정보시스템, WAS 프로그래밍, 모니터링 시스템