Journal of the Korea Academia-Industrial Cooperation Society
Vol. 12, No. 10 pp. 4600-4605, 2011

http://dx.doi.org/10.5762/KAIS.2011.12.10.4600

경량 컨테이너 구조 환경에서 하이버네이트와 아이바티스의 기능별 수행 속도 비교 연구

이명호[1*]
[1]세명대학교 전자상거래학과

A Study on Comparison of Functional Performance Test of Hibernate and iBatis with 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 2.3에 대하여 기능별 수행 속도를 비교하여 프로젝트 규모별 장단점을 분석해 보는데 그 목적이 있다. 현재까지 경량 컨테이너 구조로 많이 사용되고 잘 알려진 구조로 스프링 프레임워크이 있다. 또한 데이터베이스의 생산성을 높여주기 위한 기법으로 ORM이 있다. 현재 많이 사용되는 ORM 도구로 하이버네이트와 아이바티스가 있다. 따라서 본 연구에서는 동일한 스프링 프레임워크 2.5 환경을 기반으로 하이버네이트 3.2와 아이바티스 2.3에서 파일럿 시스템을 설계하고 구현하여 CRUD별 수행 속도를 비교함으로써 프로젝트의 적용에 평가 지표를 제공하고자 한다.

Abstract The purpose of this paper is to compare performance test of Hibernate 3.2 and iBatis 2.3 in the identical platform environment of Spring Framework 2.5 and to analyze their strengths and weaknesses. Currently Spring Framework is mostly used and well known lightweight container architecture. Both Hibernate and iBatis are mostly used instruments of ORM which is a method to enhance productivity of database. Therefore, this paper aim to design and implement pilot system based on Spring Framework 2.5 using Hibernate 3.2 and iBatis 2.3 and compare performance speed of CRUD which can be served as performance evaluation index for future projects.

Key Words : Performance Test, Hibernate 3.2, iBatis 2.3, Spring Framework 2.5, CRUD

1. 서 론

지난 10년 동안 인터넷과 인터넷 관련 기업들의 비약적인 확산과 성장은 전 세계 상장 기준의 인터넷 기업을 조사한 결과 2010년 매출액은 2000년 대비 8배, 시가 총액은 5배 증가하였다. 향후 인터넷의 미래는 모바일, 소셜, 영상 및 스마트 등의 키워드를 중심으로 진화 발전될 전망이다[1]. 또한 웹 3.0의 인터넷 혁명의 파동에 대한 가설을 기반으로 사용자가 언제 어디서나 인터넷 접속을 통하여 정보기술의 자원을 제공받는 주문형 IT 서비스인 클라우드 컴퓨팅 서비스 시대가 도래하고 있다. 이러한 디지털 공간의 활동과 모바일 기기의 사용 확대에 따라 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[2]. 이러한 엔터프라이즈 애플리케이션 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 분산 환경을 지원하기 위

*교신저자 : 이명호(mhlee@semyung.ac.kr)
접수일 11년 09월 05일 수정일 11년 09월 23일 게재확정일 11년 10월 06일

하여 객체를 직렬화하는 과정에 따라 실행 속도의 저하, 엔터프라이즈 개발의 복잡성 증가로 인한 개발 생산성의 저하, 품질저하 및 대형 벤더사들의 EJB 컨테이너 사이의 이식성 저하 등의 단점이 발생한다. 이러한 Non EJB와 EJB 구조가 가지고 있는 문제점을 해결하고 자바 엔터프라이즈 개발을 편하게 해주는 오픈소스 경량 애플리케이션 프레임워크가 스프링이다[3-5]. 또한 대용량 엔터프라이즈 웹 애플리케이션 개발에서는 데이터베이스의 SQL 코드를 작성하는데 소요시간을 줄여 생산성을 높이는 것이 아주 중요한 문제로 대두된다. 이러한 문제를 해결할 수 있는 방법으로 가장 널리 사용되는 기법은 데이터베이스 테이블과 객체사이의 매핑을 자동으로 처리해주는 ORM 기법이다. 현재 ORM 도구로 널리 이용되고 있는 도구로는 인도와 중국 등 동남아 지역에서 많이 사용되고 있는 하이버네이트(Hibernate)와 특히 한국에서 많이 사용하고 있는 아이바티스(iBatis)가 있다[6-10].

따라서 본 연구에서는 동일한 스프링 프레임워크 2.5 환경에서 하이버네이트와 아이바티스의 서로 다른 ORM 기법을 활용한 파일럿 프로그램을 설계하고 구현하여 CRUD(Create, Retrieve, Update, Delete)를 이용하여 기능별 정량적 수행 속도를 비교하도록 한다.

2. 개발 환경의 기본 개념

2.1 경량 컨테이너 구조의 고찰

자바 엔터프라이즈 애플리케이션 개발을 편하게 해주는 오픈소스 경량 컨테이너 아키텍처의 가장 잘 알려진 구조가 스프링 프레임워크이다[2,11]. 현재까지 표 1과 같은 일정으로 발표되었다.

[표 1] 스프링 프레임워크의 발표 일정표
[Table 1] Release of Spring Framework

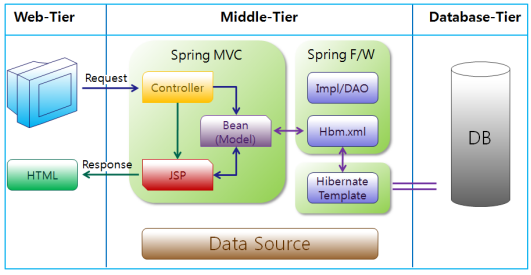
일 정	내 용
2003년 06월	스프링 프레임워크 1.0 릴리즈
2007년 11월	스프링 프레임워크 2.5 릴리즈
2008년 12월	스프링 프레임워크 3.0 M1 발표
2009년 08월	스프링 프레임워크 3.0 M4 발표
2011년 02월	스프링 프레임워크 3.1 M1 발표
2011년 06월	스프링 프레임워크 3.1 M2 발표

스프링 애플리케이션은 POJO를 이용해서 만든 애플리케이션 코드와 POJO가 어떻게 관계를 맺고 동작하는지를 정의해 놓은 설계 정보로 구성된다.

애플리케이션을 POJO로 개발할 수 있게 해주는 가능 기술인 스프링의 주요기술로는 IoC(Inversion of Control)/DI(Dependency Injection), AOP(Aspect Oriented Programming), PSA(Portable Service Abstraction)가 있다. IoC는 프로그램의 제어흐름 구조가 뒤바뀌는 것을 말한다. 여기에서는 모든 제어 권한을 자신이 아닌 다른 대상에게 위임하기 때문에 객체가 자신이 사용할 객체를 스스로 선택하지 않고 생성하지도 않는다. 자신도 어떻게 만들어지고 어디서 사용되는지를 알 수 없다. DI는 객체 레퍼런스를 외부로부터 제공(주입)받고 이를 통해 여타 객체와 다이내믹하게 의존관계가 만들어지는 것이 핵심이다. AOP는 객체지향 기술이 점점 복잡해져가는 애플리케이션의 요구조건과 기술적인 난해함을 모두 해결하기 어려운 한계와 단점을 극복하도록 도와주는 보조적인 프로그래밍 기술이다. PSA는 환경과 세부 기술의 변화에 관계없이 일관된 방식으로 기술에 접근할 수 있게 해주는 가능 기술이다[12].

2.2 하이버네이트의 구성

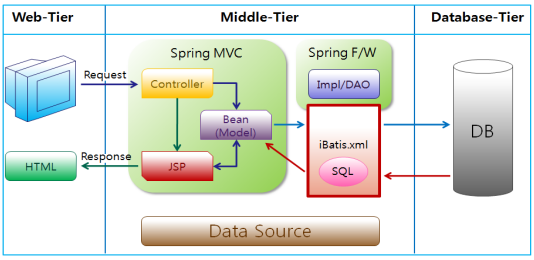
하이버네이트는 자바 환경에서 객체지향 개념을 가진 실체 클래스를 관계형 데이터베이스에 있는 테이블로 대응시키는 도구이다. 가상의 객체지향 데이터베이스를 효과적으로 만들어 관계형 데이터베이스를 객체지향 언어의 개념으로 연계하는 프로그램 기술이다. 그림 1은 본 연구에서 개발 플랫폼 상의 하이버네이트 구성도를 도식화 한 것이다[6,7].



[그림 1] 하이버네이트의 구성도
[Fig. 1] Architecture of Hibernate

2.3 아이바티스의 구성

아이바티스는 SQL과 자바 객체를 매핑 기능을 수행하는 툴이다. 그러나 객체의 상속이나 다양한 컬렉션 클래스의 지원 등의 많은 부분에서 풍부한 객체 모델링을 제공하고 있지는 못하다[10]. 그림 2는 본 연구에서 개발 플랫폼 상의 아이바티스 구성도를 도식화 한 것이다.



[그림 2] 아이바티스의 구성도
[Fig. 2] Architecture of iBatis

3. 기능별 수행 속도 비교 방안

3.1 테스트 환경

본 연구에서는 하이버네이트 3.2와 아이바티스 2.3 사양을 기반으로 하는 기능별 속도를 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 기능별 수행 속도를 비교하도록 한다.

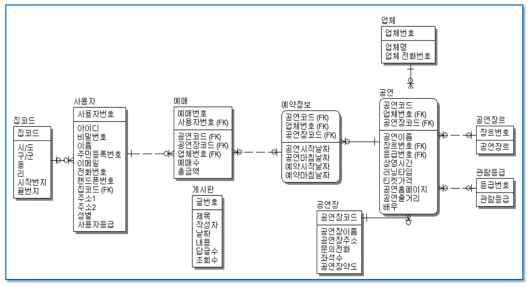
[표 2] 파일럿 테스트의 개발 환경
[Table 2] Development Environment of Pilot Test

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 기능별 수행 속도 비교 선행 조건으로는 단일 테이블(Board-게시판)과 Join 테이블(PerfInfo-공연정보)을 샘플로 선정한 후, 등록, 수정, 삭제, 조회, 등록 후 조회, 수정 후 조회 등 6개 기능별 각각의 수행 속도를 측정하도록 한다. 다음으로 매핑 기능 수행 전부터 수행 직후의 시간을 산출한다. 1천, 5천, 1만, 3만, 5만 건의 데이터를 기준으로 측정한다. 각 건당 3회 이상 기능 수행 후 평균값을 표기한다. 그리고 속도는 millisecond(1/1000 초)로 추산하여 비교 분석한다.

3.2 데이터베이스 스키마

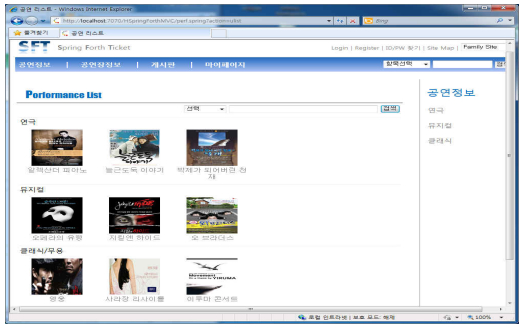
동일한 경량 컨테이너 구조 환경에서 기능별 수행 속도 비교를 위하여 개발될 파일럿 시스템의 데이터베이스 스키마는 스프링 프레임워크 2.5 사양에서 그림 3과 같이 동일한 데이터베이스 스키마를 이용하여 비교 분석한다[2].



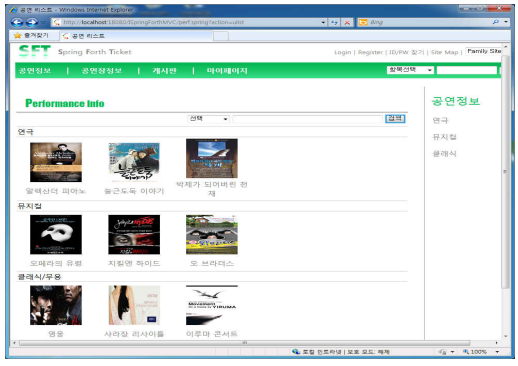
[그림 3] 데이터베이스 스키마의 구성도
[Fig. 3] Architecture of DB Schema

3.3 파일럿 시스템의 구현

이상과 같은 데이터베이스 스키마를 기반으로 분석 및 설계를 통하여 동일한 플랫폼 개발 환경에서 하이버네이트 3.2와 아이바티스 2.3의 파일럿 시스템을 구현하여 등록, 수정, 삭제, 조회 기능의 수행 속도를 비교하기 위한 공연정보 화면은 그림 4와 그림 5와 같다.



[그림 4] 하이버네이트의 공연정보
[Fig. 4] Performance Information of Hibernate



[그림 5] 아이바티스의 공연정보
[Fig. 5] Performance Information of iBatis

4. 기능별 수행 속도 비교 평가

4.1 등록 기능의 평가

게시판의 단일 테이블과 공연정보의 Join 테이블을 파일럿으로 선정하여 등록 기능의 수행 속도를 비교해 보면 표 3과 같다. 수행 속도는 매핑 기능의 수행 전부터 수행 직후의 시간을 산출하여 3회 이상의 수행한 후 평균값으로 나타내었다.

[표 3] 등록 기능의 수행 속도 비교
[Table 3] Performance Test of INSERT

자료 건수(건)		Hibernate(*ms*)	iBatis(*ms*)
Board (단일)	1,000	3,653	2,648
	5,000	13,799	10,548
	10,000	23,344	17,520
	30,000	70,423	51,674
	50,000	140,046	77,738
PerfInfo (Join)	1,000	9,374	1,797
	5,000	44,501	5,898
	10,000	91,423	11,412
	30,000	268,804	33,959
	50,000	432,624	57,757

표 3에서 보는 봐와 같이 단일 테이블일 경우에는 아이바티스가 하이버네이트보다 30%이상 수행 속도가 좋으며, Join 테이블일 경우에는 등록 기능의 수행 속도가 아이바티스가 매우 탁월했다. 특히 자료 건수가 많을수록 아이바티스가 하이버네이트보다 수행 속도가 좋은 것으로 나타났다.

4.2 수정 기능의 평가

게시판의 단일 테이블과 공연정보의 Join 테이블을 파일럿으로 선정하여 수정 기능의 수행 속도를 비교해 보면 표 4와 같다.

[표 4] 수정 기능의 수행 속도 비교
[Table 4] Performance Test of UPDATE

자료 건수(건)		Hibernate(*ms*)	iBatis(*ms*)
Board (단일)	1,000	1,491	1,051
	5,000	8,869	5,292
	10,000	12,441	10,891
	30,000	39,340	32,405
	50,000	71,379	53,435
PerfInfo (Join)	1,000	6,010	1,170
	5,000	34,863	5,469
	10,000	70,891	11,012
	30,000	200,847	34,995
	50,000	326,851	55,462

수정 기능의 수행 속도에서는 단일 테이블일 경우 아이바티스가 하이버네이트보다 10,000건 일 때 최소 14%에서 5,000건일 때 최고 68%로 수행 속도가 좋았다. Join 테이블일 경우에는 모든 건 수에서 아이바티스가 수정 기능의 수행 속도가 매우 탁월했다.

4.3 삭제 기능의 평가

게시판의 단일 테이블과 공연정보의 Join 테이블을 파일럿으로 선정하여 삭제 기능의 수행 속도를 비교해 보면 표 5와 같다.

[표 5] 삭제 기능의 수행 속도 비교
[Table 5] Performance Test of DELETE

자료 건수(건)		Hibernate(*ms*)	iBatis(*ms*)
Board (단일)	1,000	2,922	1,022
	5,000	15,839	4,501
	10,000	27,708	10,535
	30,000	87,849	31,662
	50,000	138,540	52,343
PerfInfo (Join)	1,000	5,771	1,289
	5,000	28,644	6,121
	10,000	61,973	11,888
	30,000	176,383	35,265
	50,000	284,627	62,550

삭제 기능의 수행 속도에서는 단일 테이블이나 Join 테이블에서 아이바티스가 하이버네이트보다 탁월한 수행 속도를 보였다. 특히 Join 테이블일 경우에 삭제 기능의 수행 속도가 매우 빨랐다.

4.4 조회 기능의 평가

게시판의 단일 테이블과 공연정보의 Join 테이블을 파일럿으로 선정하여 조회 기능의 수행 속도를 비교해 보면 표 6과 같다.

[표 6] 조회 기능의 수행 속도 비교
[Table 6] Performance Test of RETRIEVE

자료 건수(건)		Hibernate(*ms*)	iBatis(*ms*)
Board (단일)	1,000	696	540
	5,000	1,186	793
	10,000	1,716	1,186
	30,000	3,812	2,122
	50,000	5,346	3,073
PerfInfo (Join)	1,000	925	678
	5,000	1,723	953
	10,000	2,792	1,309
	30,000	6,365	2,683
	50,000	10,214	3,994

조회 기능의 수행 속도에서는 단일 테이블일 경우 아이바티스가 하이버네이트보다 1,000건 일 때 최소 29%에서 30,000건일 때 최고 78%로 수행 속도가 빨랐다. Join 테이블일 경우에는 모든 건 수에서 아이바티스가 조회 기능의 수행 속도가 매우 좋았다.

4.5 등록후 조회 기능의 평가

게시판의 단일 테이블과 공연정보의 Join 테이블을 파일럿으로 선정하여 등록후 조회 기능의 수행 속도를 비교해 보면 표 7과 같다.

[표 7] 등록후 조회 기능의 수행 속도 비교
[Table 7] Performance Test of RETRIEVE after INSERT

자료 건수(건)		Hibernate(*ms*)	iBatis(*ms*)
Board (단일)	1,000	106	61
	5,000	481	258
	10,000	972	493
	30,000	3,021	1,523
	50,000	4,643	2,568
PerfInfo (Join)	1,000	249	190
	5,000	930	413
	10,000	2,225	810
	30,000	5,423	2,116
	50,000	9,469	3,488

등록후 조회 기능의 수행 속도에서는 단일 테이블일 경우 아이바티스가 하이버네이트보다 1,000건 일 때 최소 74%에서 30,000건일 때 최고 98%로 수행 속도가 높았다. Join 테이블일 경우에는 모든 건 수에서 아이바티스가 등록후 조회 기능의 수행 속도가 탁월했다.

4.6 수정후 조회 기능의 평가

게시판의 단일 테이블과 공연정보의 Join 테이블을 파일럿으로 선정하여 수정후 조회 기능의 수행 속도를 비교해 보면 표 8과 같다.

[표 8] 수정후 조회 기능의 수행 속도 비교
[Table 8] Performance Test of RETRIEVE after UPDATE

자료 건수(건)		Hibernate(*ms*)	iBatis(*ms*)
Board (단일)	1,000	98	47
	5,000	479	247
	10,000	953	493
	30,000	2,895	1,476
	50,000	4,682	2,475
PerfInfo (Join)	1,000	204	66
	5,000	914	357
	10,000	1,811	680
	30,000	5,511	2,030
	50,000	9,728	3,356

수정후 조회 기능의 수행 속도에서는 단일 테이블일 경우 아이바티스가 하이버네이트보다 10,000건 일 때 최소 93%에서 1,000건일 때 최고 109%로 수행 속도가 높았다. Join 테이블일 경우에는 모든 건 수에서 아이바티스가 수정후 조회 기능의 수행 속도가 매우 탁월했다.

이상과 같이 대부분의 기능별 수행 속도 면에서는 아이바티스가 하이버네이트보다 효율적이다. 그러나 아이바티스는 기존의 개발 방식을 가진 사람들에게는 편리하지만 프로젝트 규모가 커지고 DB 테이블이 더 많아진다면 소스 코드가 길어지는 단점이 있다. 하이버네이트의 경우 객체 매핑 테이블 정의 등과 같이 초기설정의 어려움이 존재하지만 한 번의 초기 설정 후에는 사용의 편리함이 있다.

5. 결 론

비약적인 디지털 공간의 활동과 모바일 기기의 사용 확대에 따라 엔터프라이즈 애플리케이션 환경에서 막대한 대용량 데이터가 발생되고 있다. 이러한 대용량 데이터 처리 문제를 해결할 수 있는 방법으로 가장 널리 사용되는 기법으로 데이터베이스 테이블과 객체사이의 매핑을 자동으로 처리해 주는 ORM 기법이 점차 중요한 문제로 대두되고 있다. 현재까지 가장 널리 알려진 ORM 솔루션으로 하이버네이트와 ORM 매퍼로 현업에서 널리 이용되고 있는 도구로 아이바티스가 있다. 그러나 현재까지 경량 컨테이너 아키텍처의 성공 모델로 알려진 스프링 프레임워크환경에서 하이버네이트와 아이바티스 간의 기능별 수행 속도 평가 연구가 부족하여 프로젝트의 규모와 환경에 따라 어떠한 도구를 사용하는 것이 소프트웨어 생산성의 평가에 도움이 되는지에 대한 평가 지표에 제한이 있었다.

따라서 본 연구에서는 동일한 스프링 프레임워크 2.5 환경에서 서로 다른 하이버네이트와 아이바티스의 ORM 기법을 적용한 파일럿 프로그램을 설계하고 구현하여 CRUD를 이용한 기능별 정량적 수행 속도를 비교하였다. 또한 이와 같은 결과를 통하여 데이터를 처리할 프로젝트 규모별 ORM 기법의 생산성 평가를 하였다. 향후에는 동일한 데이터 스키마를 이용하여 스프링 프레임워크 2.5/3.0/3.1이나 EJB 3.0/3.1의 기능별 수행 속도 비교 연구가 지속되어야 할 것이다.

References

[1] K. D. Kwon, “The Past 10 years, Topography Change of Internet Industry”, SERI Management Note, No.114, pp. 1-10, 2011.

[2] M. H. Lee, “A Study on Comparison of Development Productivity of Hibernate 3.2 and iBatis 2.3 Based Lightweight Container Architecture”, Journal of The Korea Academia- Industrial cooperation Society, Vol.12 No.4, pp. 1919-1926, 2011.

[3] M. H. Lee, “A Study on Comparison of Development Productivity of Spring Framework 2.0 and 2.5 with Lightweight Container Architecture”, Journal of The Korea Academia- Industrial cooperation Society, Vol.10 No.6, pp. 1265-1274, 2009.

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

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

[6] H. S. Chae, “Object Oriented CBD Development Bible”, Hanbitmedia, pp. 35-76, 2005.

[7] B. G. Choi, “Hibernate 3 Programming”, Kame, pp. 14-414, 2007.

[8] J. S. Park, “Spring Framework Workbook”, Hanbitmedia, pp. 26-377, 2006.

[9] B. G. Choi, “Spring 2.5 Programming for Web Developer”, Kame, pp. 24-440, 2008.

[10] C. Begin, B. Goodin and L. Meadors, “iBatis in Action”, Manning, pp. 3-302, 2007.

[11] R. Johnson, et al., “Professional Java Development with the Spring Framework”, Wrox, pp. 1-303, 2005.

[12] I. M. Lee, “Toby's Spring 3”, Acorn, pp. 56-686, 2010.

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