한국산학기술학회논문지
Vol. 10, No. 6, pp. 1265-1274, 2009

동일한 경량 컨테이너 구조 환경에서 스프링 프레임워크 2.0과 2.5의 개발 생산성 비교 연구

이명호[1*]
[1]세명대학교 전자상거래학과

A Study on Comparison of Development Productivity of Spring Framework 2.0 and 2.5 with Lightweight Container Architecture

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

요 약 본 논문은 스프링 프레임워크 2.0과 2.5와 연관된 객체지향 소프트웨어 개발 생산성에 대한 지침과 평가 지표를 제공하는데 목적이 있다. 스프링 프레임워크는 경량 컨테이너 아키텍처로 성공적인 오픈 소스 표준 모델로 알려져 있다. 그러나 동일한 플랫폼 상에서 스프링 프레임워크 2.0과 2,5에 대한 성능 평가 연구는 부족하였다. 또한 정량적 분석도 일부분의 LoC(Line of Code) 분석만 시도함에 따라 새로운 사양이 발표됨에도 구체적인 평가 지표와 지침이 부족하여 소프트웨어 생산성의 평가와 프로젝트의 새로운 시도에 제한이 있었다. 따라서 본 연구에서는 동일한 플랫폼 상에서 스프링 프레임워크의 새로운 버전의 개발 생산성 평가하기 위한 특정 지침을 제시하고, 이전의 사양과의 객관적인 소프트웨어 개발 생산성 지침을 제공하고자 한다.

Abstract This paper proposes an object-oriented software development guidance and an evaluation index for the productivity related to Spring Framework 2.0 and 2.5. Spring Framework is a known successful open source standard model for lightweight container architecture. However, there is no comparison research about the performance of Spring Framework 2.0 and 2.5 with same identical platform. Quantitative analysis is supported as a part of LoC(Line of Code) analysis. There is a limit to develop the updated software with no the specific evaluating index for the productivity of the software. This work proposes an specific index for evaluating the productivity of new version Spring Framework on a platform. Base on the result, the specific guidance of the developing software is obtained.

Key Words : Spring Framework 2.0 and 2.5, Lightweight Container Architecture, LoC

1. 서론

디지털 컨버전스 시대에서의 컴퓨터 아키텍처는 인터넷의 주도아래 근본적인 거대한 변화의 시대를 맞이하고 있다. 또한 웹이 진화하면서 데이터뿐만 아니라 응용 애플리케이션 프로그램까지 데스크톱에서 해방되어 외부 데이터 센터에 저장해 놓고 사용할 수 있는 클라우드 컴퓨팅(Cloud Computing) 환경의 시대를 예고하고 있다[4]. 따라서 운영환경 통합은 온-디멘드로, 기반구조의 통합은 그리드나 유틸리티로, 개발통합은 통합개발 환경으로, 데이터베이스 통합은 데이터 허브나 EAI로, 사용자 인터페이스 통합은 RIA로 통합화 및 표준화가 진화되고 있다[2,3]. 이러한 엔터프라이즈 환경에서는 이 기종 컴퓨터들 간에 프로그램을 분산시켜 부하를 줄여 시스템의 성능

*교신저자 : 이명호(mhlee@semyung.ac.kr)
접수일 09년 05월 01일 수정일 09년 06월 10일 게재확정일 09년 06월 17일

저하와 네트워크 병목 현상을 줄일 수 있는 분산객체 구조가 필요하게 되었다. 또한 복잡한 시스템 요구조건을 신속히 구현하기를 원하는 유비쿼터스 정보화 시대에서는 다양한 객체 지향 및 분산 객체 개발 방법론을 거쳐 현재는 컴포넌트 기반 개발 방법에 이르게 되었다[6]. 컴포넌트는 고유한 기능을 수행하는 독립적인 소프트웨어의 단위를 말하며, 인터페이스와 구현의 분리를 통해 캡슐화를 통하여 컴포넌트 제공자와 사용자 사이에서 독립성을 확보하여 소프트웨어의 재사용성을 높일 수 있게 하는 방법론이다. 컴포넌트 모델은 컴포넌트 설계와 구현 단계에서 표준 규약을 통하여 컴포넌트에 대한 일관성 있는 관리를 지원하며, 컴포넌트 패키징, 분산, 트랜잭션 관리, 통신, 보안 등의 서비스가 포함된다. 그러나 이러한 컴포넌트 모델의 분산 응용 프로그램을 운영하기 위하여 CORBA, DCOM, RMI 등이 개발되었지만 지속성있는 데이터를 표현하기 위한 표준화된 방법이 없었고, 트랜잭션, 보안, 멀티쓰레딩 등의 서비스를 위하여 개발자들이 직접 코드를 작성해야 하였다. 이러한 문제점들을 해결하기 위하여 현재 인정되고 있는 컴포넌트 모델의 표준은 MS사의 COM+, OMG의 CCM(CORBA Component Model), SUN사의 EJB(Enterprise JavaBeans) 등이 있지만, 이 중에서 대용량 분산 객체의 가장 성공모델로 알려진 것이 EJB이다. EJB는 자바 프로그램처럼 단독으로 실행되는 것이 아니라 EJB 컨테이너라는 소프트웨어에 설치되어야 실행될 수 있으며, EJB 컨테이너는 EJB 서버에 포함되어 있다.

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[8]. Non EJB와 EJB 아키텍처가 가지고 있는 문제점을 해결하고 장점들을 지원하기 위하여 새롭게 등장한 아키텍처가 경량 컨테이너 아키텍처이다. 이와 같이 경량 컨테이너 아키텍처의 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 프레임워크이다.

현재까지 플랫폼의 변화에 따른 개발 생산성에 대한 비교 연구는 2가지의 애플리케이션에 대하여 다른 J2EE 플랫폼에서의 개발 생산성을 비교한 연구였으며[11], 동일한 플랫폼 상에서 EJB 2.0과 EJB 3.0 사양에 대하여 정량적인 평가 지표에 따른 개발 생산성 연구는 있었다[3]. 그러나 스프링 프레임워크 2.0과 2.5에 대하여 같은 플랫폼 상에서 소프트웨어 개발 생산성 비교에 대한 연구가 미비하였으며, 정량적 분석도 일부분의 LoC(Line of Code) 분석만 시도함에 따라 새로운 사양이 발표됨에도 구체적인 평가 지표와 지침이 부족하여 소프트웨어 생산성의 평가와 프로젝트의 새로운 시도에 제한이 있었다.

따라서 본 연구에서는 Non EJB와 EJB 아키텍처가 가지고 있는 문제점을 해결하고 장점들을 지원하기 위하여 개발된 같은 플랫폼 상에서 스프링 프레임워크 2.0과 2.5 사양에 대하여 정량적인 평가 지표를 제시하여, 새로운 스프링 프레임워크 사양에 대한 정량적인 분석을 통하여 객관적인 소프트웨어 개발 생산성 연구에 대한 지침을 제공하고자 한다.

2. 스프링 프레임워크의 기본 개념

2.1 스프링 프레임워크의 고찰

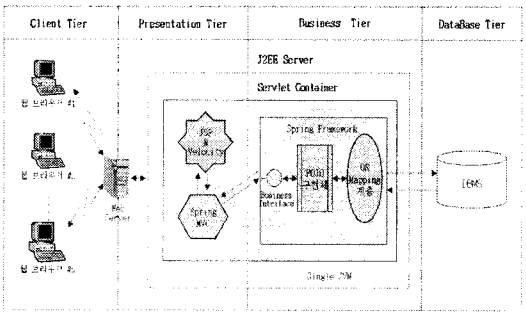
현재까지 경량 컨테이너 아키텍처의 가장 잘 알려진 구조로는 스프링 프레임워크이며, 첫 번째 버전은 2002년 10월 Rod Johnson이 Wrox 출판사에서 출간한 "Expert One-on-One J2EE Design and Development"에서 처음 소개되었으며, 프레임워크는 2003년 6월에 Apache 2.0 라이센스로 릴리즈 되었다. 2004년 3월에 첫번째 스프링 프레임워크 1.0 마일스톤이 릴리즈 되었고, 2006년 스프링 프레임워크 2.0이 릴리즈 되었다. 2007년 11월에 스프링 프레임워크 2.5가 릴리즈 되었으며. 2008년 12월 스프링 프레임워크 3.0 M1이 발표되었고, 2009년 1월 스프링 프레임워크 3.0 M2가 발표되었다.

그러나 스프링 프레임워크 2.5에서 기존 2.0 버전과 비교하여 새로운 특징의 변화가 있었다. 가장 큰 특징으로는 애노테이션(Annotation)을 이용한 의존성 삽입(DI : Dependency Injection)의 도입이다. 또한 현재까지 스프링 프레임워크 3.0에서도 2.5의 기능에 애노테이션 설정이 좀 유연하고 폭넓게 사용할 수 있도록 조금 발전한 것뿐이다[2,8].

따라서 본 연구에서는 가장 큰 특징과 변화를 가지고 있으며 안정된 스프링 프레임워크 2.5를 기반으로 파일럿 시스템을 설계하여 구현하도록 한다.

2.2 스프링 프레임워크의 구성

스프링 프레임워크 2.0과 달리 2.5의 가장 큰 특징은 애노테이션을 이용한 의존성 삽입이다. 스프링이 시작한 의존성 삽입 기술은 피코 콘테이너, EJB3.0, SEAM, Google Guice 등의 다양한 프레임워크와 기술 스펙으로 발전해 왔다[5]. 의존성 삽입 기술은 스프링 1.0부터 2.0까지 계속 발전해오고 있는 기술이다. 의존성 삽입은 Constructor Injection, Setter Injection, Interface Injection 등의 크게 3가지 유형을 가진다[1,8,10]. Constructor Injection은 생성자를 이용해서 의존성을 설정해주는 방법이고, Setter Injection은 Setter 메서드를 이용하여 의존성을 설정해 주는 방법이다. 스프링은 자바 빈 규칙을 이용한 Setter Injection을 주로 사용한다. 또한 Factory Bean 기능이 추가되어 빈의 생성 방식이 유연하게 만들 수 있는 길을 열어 주었다. 스프링 프레임워크 2.0에서는 이러한 의존성 삽입의 범위를 스프링이 직접 관리하지 않는 객체에게로 확대하는 기능이 추가되었으며, 스프링 프레임워크 2.5에서는 단지 기존의 XML 설정 기능을 그대로 애노테이션으로 적용한 수준이 아니라 애노테이션 방식의 특징을 최대한 살리면서 다른 의존성 삽입 기술에서 제공하고 있는 편리한 설정방식을 대폭 도입하게 되었다[5]. 따라서 본 연구에서 동일한 경량 컨테이너 아키텍처 환경에서 스프링 프레임워크 2.0과 2.5 사양으로 파일럿 시스템을 구현한 구성도를 살펴보면 그림 1과 같다.



[그림 1] 스프링 프레임워크의 구성도

3. 개발 생산성 비교 방안

3.1 테스트 환경

본 연구에서는 스프링 프레임워크 2.0과 2.5 사양을 기반으로 하는 소프트웨어 개발 생산성을 비교 분석하기 위한 방안으로 표 1과 같은 동일한 개발환경과 데이터베이스 스키마를 이용하여 스프링 프레임워크 2.0과 2.5 환경에서의 파일럿 프로그램을 개발한 후, 이 프로그램 통하여 각 항목별 평가 지표를 이용하여 사양별 정량적으로 개발 생산성을 비교하도록 한다.

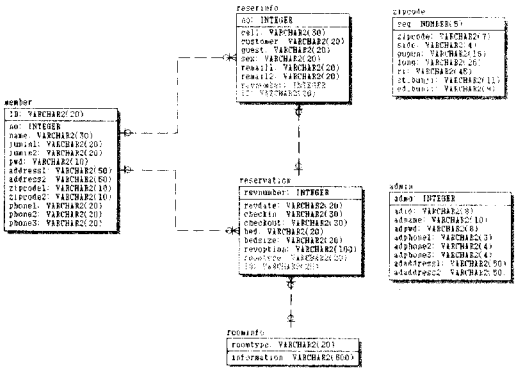
[표 1] 스프링 프레임워크 2.0과 2.5의 개발 환경

항 목	Spring 2.0	Spring 2.5
OS	Windows XP Professional	Windows XP Professional
Platform	JDK 1.6	JDK 1.6
WAS	JBOSS-4.2.3GA	JBOSS-4.2.3GA
DB	Oracle 10g	Oracle 10g
IDE	MyEclipse 6.0	MyEclipse 6.0
CASE	Rational Rose 2003	Rational Rose 2003

따라서 본 연구에서 사용한 정량적 소프트웨어 개발 생산성 평가로는 스프링 프레임워크 2.0과 2.5에서의 Controller와 ~ManageImpl에서의 서비스 코드 설정 평가 지표로 사양별 파일 개수 및 LoC의 평가와 사양별 XML의 비교 평가 등을 선정하여 분석한다.

3.2 데이터베이스 스키마

동일한 경량 컨테이너 주조 환경에서 소프트웨어 생산성 비교를 위하여 개발될 파일럿 시스템의 데이터베이스 스키마는 스프링 프레임워크 2.0과 2.5 사양에서 그림 2와 같이 동일한 데이터베이스 스키마를 이용하여 비교 분석한다.



[그림 2] 데이터베이스 스키마 구조

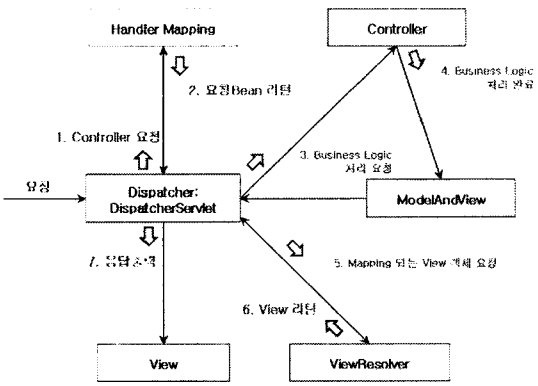
데이터베이스 스키마 구조에서 각 엔티티의 기능을 요약하면 표 2와 같다.

[표 2] 엔티티의 기능

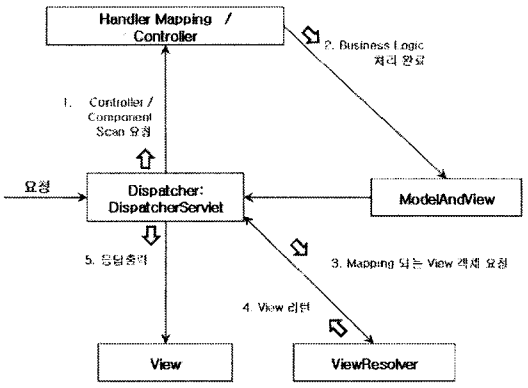
엔티티명	설 명
Admin	시스템을 관리하는 관리자 정보
Member	시스템에 가입된 회원의 정보
Reservation	회원이 예약한 예약 정보
Reserinfo	예약자와 실제 투숙자 구별을 위한 예약자 정보
Roominfo	객실의 변동 사항을 관리하는 정보
ipcode	전국의 우편번호 정보

3.3 스프링 MVC의 동작 패턴

웹 애플리케이션을 개발하기 위하여 초기 개발 속도 및 쉬운 접근성으로 요청의 최초 진입점을 JSP로 시작하는 모델 1 방식과 요구사항에 대한 대응 속도의 느림과 유지보수의 어려움, 정교한 사용자 인터페이스 개발의 어려움의 문제점을 극복한 요청의 최초 진입점이 컨트롤러인 서블릿으로 진입하는 대안인 모델 2 방식 있다. 모델 2 방식의 근간이 되는 개념이 MVC이다[1]. 스프링 MVC에서 모델은 사용자 인터페이스 티어 및 비즈니스 티어 사이에서 주고받는 데이터를 말한다. 다음 그림 3과 그림 4는 스프링 MVC 2.0과 2.5의 동작 패턴을 도식화한 그림이다.



[그림 3] 스프링 MVC 2.0의 동작 패턴

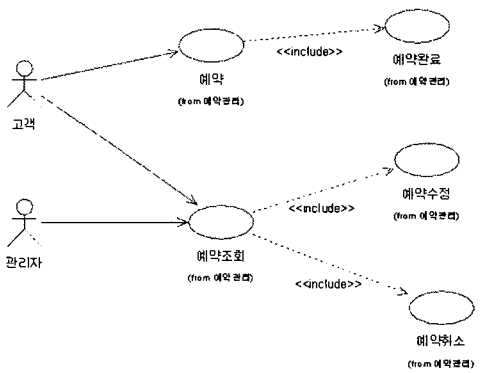


[그림 4] 스프링 MVC 2.5의 동작 패턴

3.4 유스케이스 다이어그램

파이럿 시스템의 예약관리에 대한 요구사항 정의 활동에서 파악된 액터와 유스케이스를 유스케이스 다이어그램으로 표현해 보면 그림 5와 같은 예약관리의 유스케이스 모델이 된다.

예약관리의 유스케이스 모델에 기술된 유스케이스 이름만으로는 유스케이스가 나타내려고 하는 기능을 명확하게 정의될 수 없기 때문에 이에 대한 보다 자세한 정보를 기술한 문서 산출물인 유스케이스 명세서를 기술해 보면 표 3과 같다.



[그림 5] 예약관리의 유스케이스 다이어그램

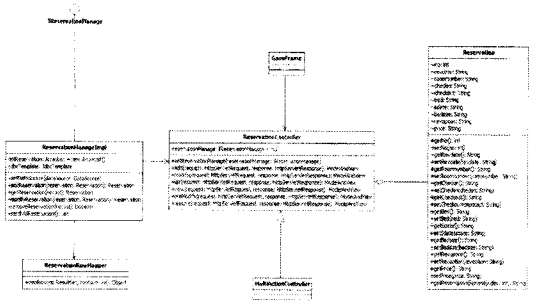
유스케이스 명세서에 기술된 이벤트 흐름은 유스케이스가 나타내는 시스템의 기능이 액터와 시스템 간의 상호작용으로 진행되는 과정을 보여준다.

[표 3] 예약관리 유스케이서 명세서

예약관리 유스케이스 명세서	
개 요	회원은 호텔에 투숙하기 위해 예약한다.
관련 액터	회원
우선순위	상
선행조건	시스템에 로그인 되어 있어야 한다.
이벤트 흐름	기본흐름 ①메인 화면에서 예약을 클릭한다. ②시스템은 예약정보 입력 페이지를 보여준다. ③양식에 따라 예약 날짜, 체크인 날짜, 체크아웃 날짜, 요청사항 등을 입력한 후 확인을 누른 후 예약자 정보 입력 페이지로 이동한다. ④회원은 예약자 정보를 입력(예약자, 투숙객, 성별, 휴대폰, e-mail, 주소, 요청 사항 등)후 확인버튼을 클릭한다. ⑤입력이 완료 되면 예약 완료 화면 예약번호를 보여준다. ⑥회원이 완료 버튼을 누르면 회원전용 메인 화면으로 돌아간다.
	대안흐름 A1) 예약정보 및 예약자 정보 입력:필수 입력 사항을 누락 하였을 경우. ① 필수 입력사항 누락시 '누락 되었습니다.' 라는 메시지를 뿌려주고 ③으로 돌아간다. ② 사용자는 ③부터 다시 수행한다.
후행조건	예약 정보와 예약자 정보가 입력된다.
기타 요구사항	없음

3.5 클래스 다이어그램

비기능적인 요구사항과 플랫폼을 고려한 후, 설계 활동을 통한 분석 클래스를 구체화하여 설계 클래스를 도출한다. 따라서 본 연구의 파일럿 시스템에서 중요한 예약관리의 설계객체 모델인 클래스 다이어그램은 그림 6과 같다.



[그림 6] 예약관리 클래스 다이어그램

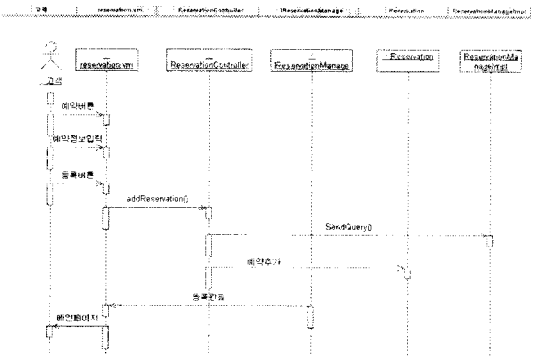
다음 표 4는 파일럿 시스템 중 예약관리에 대한 클래스 정의를 요약한 표이다.

[표 4] 예약관리에 대한 클래스 정의

클래스 명	타입	설 명
IReservation Manage	public interface	예약관리에 필요한 유스케이스를 실현하는 오퍼레이션을 모아놓은 인터페이스
Reservation ManageImpl	public class	IReservationManage에 모여진 오퍼레이션이 실제 수행 되는 클래스
Reservation RowMapper	public class	오퍼레이션의 실행 결과를 임시 저장 하는 클래스
Reservation Controller	public class	클라이언트의 호출을 받아 오퍼레이션의 수행을 조절 하는 클래스
MultiAction Controller	public class	ReservationController의 Spring Controller 설정을 위한 클래스
Reservation	public class	Reservation 정보를 가지고 있는 객체 클래스

3.6 시퀀스 다이어그램

설계 유스케이스 실현 모델은 파악된 설계 클래스들이 어떻게 메시지를 주고받으면서 시스템의 요구사항을 제공할 수 있는지를 표현한 시퀀스 다이어그램이다. 그림 7은 본 연구의 파일럿 시스템에서 중요한 예약관리의 설계 유스케이스 실현 모델인 시퀀스 다이어그램을 도식화한 것이다.

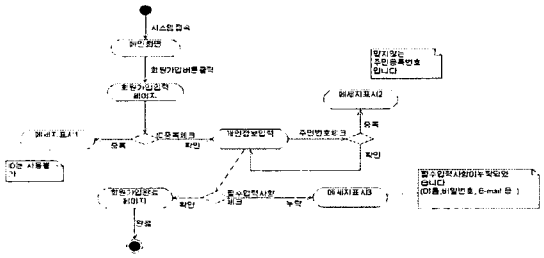


[그림 7] 예약관리 클래스 다이어그램

3.7 파일럿 시스템의 구현

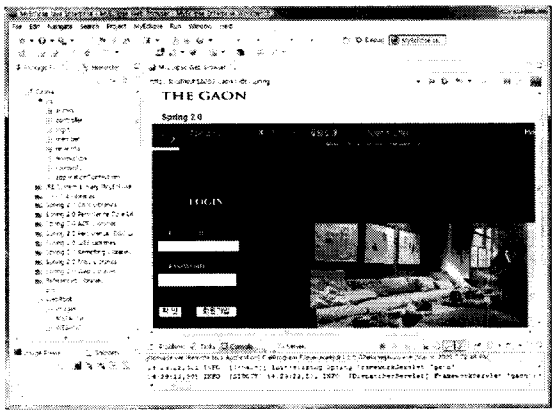
경량 컨테이너 아키텍처 환경에서 스프링 프레임워크 2.0과 2.5의 파일럿 시스템은 각 유스케이스에 사용되는 화면 간의 전환을 화면 흐름 모델로 설계함으로써 명시적으로 분석하였다. 다음 그림 8은 회원가입 관련 화면 흐름 모델과 관련된 사례로써 예약관리 유스케이스의 사

용자 인터페이스를 위하여 사용되는 화면 흐름 관계를 나타낸다.



[그림 8] 회원가입을 위한 화면 흐름도

이상과 같은 데이터베이스 스키마를 기반으로 분석 및 설계를 통하여 동일한 플랫폼 개발 환경에서 스프링 프레임워크 2.0과 2.5의 파일럿 시스템을 구현하기 위한 메인 화면은 그림 9와 그림 10과 같다.



[그림 9] 스프링 프레임워크 2.0의 파일럿 시스템

[그림 10] 스프링 프레임워크 2.5의 파일럿 시스템

4. 스프링 프레임워크의 평가

4.1 XML의 평가

스프링에서는 스프링 MVC의 DispatcherServlet에서 컨트롤러를 사용하여 클라이언트의 요청을 처리한다. 스프링 MVC는 1개 이상의 DispatcherServlet을 설정할 수 있으며, 이것은 기본적으로 웹 애플리케이션의 /WEB-INF/ 디렉터리에 위치한 [서블릿이름]-servlet.xml 파일로부터 스프링의 정보를 읽어온다. 서로 다른 DispatcherServlet이 공통 빈을 필요로 하는 경우에는 ContextLoaderListener를 사용하여 공통으로 사용될 빈을 설정할 수 있다. ContextLoaderListener는 context ConfigLocation 컨텍스트 파리미터를 명시하지 않으면 /WEB-INF/application -Context.xml을 설정 파일로 사용한다[7]. 따라서 본 연구에서 스프링 프레임워크 2.0과 2.5에서 개발된 파일럿 시스템의 중요한 XML 현황은 표 5와 같다.

[표 5] 스프링 프레임워크의 XML 현황

스프링 프레임워크 버전		2.0		2.5	
항목 / XML	설정 항목	사용 여부	LoC	사용 여부	LoC
Dispatcher Servlet	Component-Scan	x	-	O	1
	Handler Mapping	O	2	x	-
	Controller Bean 설정	O	35	x	-
	Schema Type	x (DTD)	3	O (2.5)	8 (추가확장가능)
Total(단위 : Line)		40		9	
Application Context	Component-Scan	x	-	O	5
	Resource설정 및 Bean 등록	O	35	X	-
	Schema Type	O (2.0)	4	O (2.5)	8
	Annotation Config	x	-	O	1
Total(단위 : Line)		39		14	

표 5에서 보는 봐와 같이 스프링 프레임워크 2.5가 2.0에 비하여 DispatcheServlet에서는 77.5%의 LoC 감소가 보이며, ApplicationContext에서는 64.1%의 LoC 감소가 나타났다.

(1) servlet.xml 설정 방식

스프링 프레임워크 2.0의 servlet.xml XML 문서 타입은 DTD 방식으로 다음과 같다.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN"
"http://www.springframework.org/dtd/spring-beans-2.0.dtd" >
```

스프링 프레임워크 2.5의 servlet.xml XML 문서 타입은 스키마 방식으로 다음과 같다.

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:p="http://www.springframework.org/schema/p"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd">
```

HandlerMapping 설정을 살펴보면, BeanNameUrl, SimpleUrl, AbstractUrl, AbstractBeanNameUrl Mapping, DefaultAnnotation Handler Mapping 방식 중 한 가지를 설정한다. 스프링 프레임워크 2.0에서 BeanNameUrl Handler Mapping 방식 사용예를 보면 다음과 같다.

```
<bean id="handlerMapping"
class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMappin
g" />
```

스프링 프레임워크 2.5는 애노테이션 선언과 컴포넌트 스캔을 사용하기 때문에 디폴트 애노테이션 방식을 사용한다. 따라서 애노테이션 선언과 Component-Scan을 사용하기 때문에 DefaultAnnotationHandler Mapping 방식을 사용한다.

```
<bean class=
"org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerM
apping" p:alwaysUseFullPath="true" />
```

스프링 프레임워크 2.0에서의 컨트롤러 빈의 설정은 각 컨트롤러 마다 설정해주어야 하며 사용 예는 다음과 같다.

```
<bean id="memberController" name="/member.spring"
class="controller.MemberController">
  <property name="methodNameResolver" ref="paramResolver" />
  <property name="member">
        <ref bean="memberManage" />
  </property>
</bean>
```

스프링 프레임워크 2.5에서의 컴포넌트 스캔 설정을 살펴보면, controller 패키지를 스캔 범위로 지정해 줌으로써 컴포넌트 선언 컨트롤러 빈을 자동 등록되게 한다.

```
<context:component-scan base-package="controller" />
```

(2) applicationcontext.xml 설정 방식

스프링 프레임워크 2.0의 applicationcontext.xml 설정에서 XML 문서 타입인 스키마 방식은 다음과 같다.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">
```

스프링 프레임워크 2.5에서의 applicationcontext.xml 스키마 방식은 다음과 같다.

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:p="http://www.springframework.org/schema/p"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-2.5.xsd">
```

스프링 프레임워크 2.0에서 리소스 설정 및 빈 등록 사용예를 보면 다음과 같다.

```
<bean id="memberManage" class="member.MemberManageImpl"
    abstract="false" lazy-init="default" autowire="default"
    dependency-check="default">
  <property name="dataSource">
    <ref bean="dataSource" />
  </property>
</bean>
```

또한 스프링 프레임워크 2.5에서의 컴포넌트 스캔 설정을 살펴보면, 서비스 패키지를 스캔 범위로 지정해 줌으로써 컴포넌트 선언 서비스 빈의 자동 등록 및 리소스 설정을 하도록 한다.

<context:component-scan base-package="member" />
<context:component-scan base-package="reservation" />

이상과 같이 스프링 프레임워크 2.5에서 2.0 이전 버전들 보다 XML 스키마 형의 변화로 인한 설정의 간소화와 Component-Scan으로 인한 빈 설정의 편리성, 그리고 Controller, Service, Resource, Handler-Mapping 등의 선언을 자동으로 등록 가능함에 따라 컴포넌트 추가 확장 시 효율성이 증가 등의 장점이 있다. 그러나 스키마 형을 확장 할수록 코드의 라인수가 증가하는 단점도 있다.

4.2 서비스 코드의 평가

(1) Controller 부분의 설정 평가

일반적으로 소프트웨어 개발 생산성을 평가할 때 LoC 평가 방법을 자주 사용한다. 따라서 본 연구에서도 스프링 프레임워크 2.0과 2.5 사양에서의 패키지에 대한 Controller 서비스 코드에 대한 설정사항에 대한 평가를 보면 표 6과 같다.

[표 6] Controller의 서비스 코드 설정의 평가

버 전	2.0		2.5	
Package명	설정사항	LoC	설정사항	LoC
Member	X	242	Auto-Wired, Component Controller 선언, Request-Mapping	243
Reservation	X	178	Auto-Wired, Component Controller 선언, Request-Mapping	195
ReservInfo	X	163	Auto-Wired, Component Controller 선언, Request-Mapping	173
Roominfo	X	59	Auto-Wired, Component Controller 선언, Request-Mapping	68
Admin	X	179	Auto-Wired, Component Controller 선언, Request-Mapping	192
Total (단위 : Line)		811		871

스프링 프레임워크의 Controller 부분의 서비스 코드에서는 표 7과 표 8에서와 같이 스프링 프레임워크 2.0에서는 빈 등록을 위한 설정과 Mapping 설정을 직접 XML에서 하지만, 스프링 프레임워크 2.5에서는 소스코드에 직접 선언하기 때문에 XML의 설정이 간소화와 의존관계의 자동설정으로 인한 편리성이 증가되지만 소스코드의 직접선언으로 복잡성이 야기될 수 있다.

[표 7] 2.0에서의 Controller의 설정 내용

```
package controller;

import java.util.List;

public class MemberController extends MultiActionController {

    private IMemberManage memberManage = null;

    public void setMember(IMemberManage memberManage) {
        this.memberManage = memberManage;
    }

    public ModelAndView add(HttpServletRequest request,
            HttpServletResponse response) {
```

[표 8] 2.5에서의 Controller의 설정 내용

```
package controller;

import java.util.List;

@Component
@org.springframework.stereotype.Controller
public class MemberController {

    @Autowired
    private IMemberManage memberManage = null;

    public void setMember(IMemberManage memberManage) {
        this.memberManage = memberManage;
    }

    @RequestMapping(value = "/addmember.spring", method = RequestMethod.POST)
    public ModelAndView add(HttpServletRequest request,
            HttpServletResponse response) {
```

(2) ~ManageImpl 부분의 설정 평가

스프링 프레임워크 2.0과 2.5 사양에서의 패키지에 대한 ~ManageImpl 부분의 서비스 코드에 대한 설정사항에 대한 평가를 보면 표 9와 같다.

[표 9] ~ManageImpl의 서비스 코드 설정의 평가

버 전	2.0		2.5	
Package명	설정사항	LoC	설정사항	LoC
Member	X	139	Component Resource, Service	144
Reservation	X	105	Component Resource, Service	110
ReservInfo	X	95	Component Resource, Service	103
Roominfo	X	58	Component Resource, Service	66
Admin	X	100	Component Resource, Service	114
Total (단위 : Line)		497		537

스프링 프레임워크의 ~ManageImpl 부분의 서비스 코드에서는 표 10과 표 11에서와 같이 스프링 프레임워크 2.0에서는 리소스를 빈으로 등록하여 XML에 설정을 해야 하지만, 스프링 프레임워크 2.5에서는 리소스와 서비스의 직접선언으로 XML의 설정이 필요 없어진다.

[표 10] 2.0에서의 ~ManageImpl의 설정 내용

```
package member;

import java.util.ArrayList;

public class MemberManageImpl implements IMemberManage {

    private ArrayList listMember = new ArrayList();
    private JdbcTemplate jdbcTemplate;

    public void setDataSource(DataSource dataSource) {
        this.jdbcTemplate = new JdbcTemplate(dataSource);
    }
```

[표 11] 2.5에서의 ~ManageImpl의 설정 내용

```
package member;

import java.util.ArrayList;

@Component
@Service
public class MemberManageImpl implements IMemberManage {

    private ArrayList listMember = new ArrayList();
    JdbcTemplate jdbcTemplate;

    @Resource
    public void setDataSource(DataSource dataSource) {
        this.jdbcTemplate = new JdbcTemplate(dataSource);
    }
```

이상과 같이 스프링 프레임워크 2.0에서 보다 스프링 프레임워크 2.5에서의 서비스 코드 설정 시의 장점으로는 소스코드와 함께 설정을 할 수 있고, Field, Multi ActionController의 메서드 등의 편리한 설정 방식을 지원한다. 소스코드와 함께 있기에 리팩토링 시 편리하다. 소스코드의 직접 설정으로 인한 XML의 설정의 간소화하며, 의존관계의 자동설정으로 인한 편의성이 증가한다. 또한 Controller 인터페이스를 구현하지 않은 클래스도 애노테이션을 이용하여 Controller로 사용이 가능하다. 그러나 직접선언으로 인한 코드의 복잡성을 야기 시킬 수 있으며, 형으로 빈을 찾아서 설정하는 문제도 있다. 이것은 같은 타입의 빈이 여러 개 존재할 경우 Autowiring 작업의 실패 원인이 되기도 한다. RequestMapping을 클래스에 적용하게 되면 해당 클래스는 지정한 URL만을 처리하게 되기 때문에 메서드에 적용되는 RequestMapping 애노테이션은 더 이상 URL을 명시할 수 없는 단점도 존재한다.

5. 결론

프로젝트를 실패로 만들지 않기 위하여 가장 먼저 생각할 것은 단순성, 생산성, 객체지향성, 고객이 원하는 핵심 요구사항에 중요성, 실증적인 방법으로 기술을 도입 가능성, 그리고 테스트 가능성 등의 핵심가치이다[9]. 이러한 지향점들을 추구하기 위하여 등장한 것이 스프링 프레임워크이다. 스프링 프레임워크에서는 특정한 인터페이스에 종속되지 않는 빈과 같은 클래스인 POJO를 관리하는 스프링 컨테이너에게 제어 역행화를 통한 제어권을 넘겨서 EJB 컨테이너에서 지원하던 매력적인 기능들을 지원하고 있다. 그리고 POJO 기반이기 때문에 특정 환경구축을 위한 클래스를 이입하지 않으며, 애노테이션 사용으로 인한 개발 편의성이 증가되는 장점이 있다. 그러나 설정이 바뀔 때 마다 컴파일이 필요하며, 각 티어간 연결이 인터페이스를 통해 이루어지기 때문에 인터페이스의 생성이 필요로 하는 단점도 있다. 그러나 현재까지 경량 컨테이너 아키텍처의 성공 모델로 알려진 스프링 프레임워크 2.0과 2.5 사양의 정량적인 성과지표 개발 및 사례의 부족으로 이전 사양으로 운영 중인 실무 프로젝트의 업그레이드나 새로운 기술 사양의 적용이 미비하였다. 그 이유는 기본적인 스프링 프레임워크의 기술 변화의 속도가 빠르고 표준 사양의 복잡도가 높음에 따라 쉽게 새로운 사양들을 현업에 적용하지 못한 것이다. 또한 스프링 프레임워크의 소프트웨어 개발 생산성 비교에 대한 연구도 부족한 상태이며, 스프링 프레임워크의 새로운 사양이 발표됨에도 현재까지 구체적인 분석 및 설계 기반에 따른 구현 지침이 부족하여 소프트웨어 생산성의 평가와 프로젝트의 새로운 시도에 제한이 있었다.

따라서 본 연구에서는 대용량 분산객체 시스템 처리를 위하여 동일 환경의 스프링 프레임워크 2.0과 2.5를 기반으로 파일럿 프로젝트의 분석 및 설계를 통하여 구현 지침을 제시하였으며, 또한 스프링 프레임워크 2.0과 2.5에 대한 성능 평가 기반으로 정량적인 분석을 통하여 객관적인 소프트웨어 개발 생산성 연구에 대한 지침을 제시하였다. 향후에는 AOP나 ORM 기반 구조로 스프링을 사용한 연구와 동일한 데이터 스키마를 이용하여 EJB 3.0과 스프링 프레임워크 3.0의 소프트웨어 생산성 분석 연구가 지속되어야 할 것이다.

참고문헌

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

[2] 이명호, "EJB 3.0 표준을 기반으로 대용량 분산객체 처리의 설계 및 구현", 대한설비관리학회지, 제13권 제2호, pp. 45-51, 6월, 2008.

[3] 이명호, "EJB2.0과 EJB3.0의 소프트웨어 개발 생산성 비교 연구", 한국산업경영시스템학회지, 제31권 제3

호, pp. 1-7, 9월, 2008.
[4] 이명호, "경량 컨테이너 구조 환경의 스프링 프레임 워크 2.5를 기반으로 호텔예약시스템의 설계 및 구현", 한국산학기술학회논문지, 제10권 제3호, pp. 589-595, 3월, 2009.
[5] 이일민, "자바 기술의 미래를 비추는 거울 스프링 프레임워크 2.5", 마이크로소프트웨어, pp. 136-143, 1월, 2008.
[6] 채홍석, "객체지향 CBD 개발 Bible", 한빛미디어, pp. 35-76, 8월, 2005.
[7] 최범균, "웹 개발자를 위한 스프링 2.5 프로그래밍", 가메출판사, pp. 24-440, 3월, 2008.
[8] R. Johnson, "Expert One-on-One J2EE Design and Development", Wrox, pp. 441-673, October, 2002.
[9] R. Johnson, and J. Hoeller, "Expert One-on-One J2EE Development without EJB", Wrox, pp. 1-141, June, 2004.
[10] R. Johnson, et al., "Professional Java Development with the Spring Framework", Wrox, pp. 1-303, July, 2005.
[11] 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