한국산학기술학회논문지
Vol. 10, No. 3, pp. 589-595, 2009

경량 컨테이너 구조 환경의 스프링 프레임워크 2.5를 기반으로 호텔예약시스템의 설계 및 구현

이명호[1*]
[1]세명대학교 전자상거래학과

Design and Implementation of Hotel Reservation System Based Spring Framework 2.5 of Lightweight Container Architecture

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

요 약 본 논문은 스프링 프레임워크 2.5와 연관된 객체지향 소프트웨어 개발에 대한 지침과 평가 지표를 제공하는데 목적이 있다. Non EJB와 EJB 아키텍처가 가지고 있는 문제점을 해결하고 장점들을 지원하기 위하여 새롭게 등장한 아키텍처가 경량 컨테이너 아키텍처이다. 이 구조는 EJB 아키텍처와 같이 무겁지 않으면서도 EJB 컨테이너의 모든 가능을 제공하는 구조이다. 현재까지 경량 컨테이너 아키텍처로 현업에서 가장 많이 사용되고 잘 알려진 아키텍처로 스프링 프레임워크가 있다. 따라서 본 연구에서는 Non EJB와 EJB 아키텍처가 가지고 있는 단점들을 해결하고 장점들을 지원하기 위하여 개발된 최신 경량 컨테이너 아키텍처인 스프링 프레임워크 2.5를 기반으로 호텔예약시스템의 설계 및 구현을 통하여 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 2.5. Spring Framework is a known successful open source standard model for lightweight container architecture. Non EJB and the EJB architecture to resolve the problem with benefits to support the new architecture is a lightweight container architecture. This architecture, such as the EJB, but not heavy, to provide all of the architecture is possible. The lightweight container architecture is most often used in business spring framework is well-known architecture. Therefore, this research has the Non EJB and the EJB to solve the advantages and disadvantages developed to support the latest spring framework 2.5 lightweight container architecture based on the design and implementation of a hotel reservation system with the objective through the specification of the software previously to provide guidance to development productivity.

Key Words : Spring Framework 2.5, Lightweight Container Architecture, Non EJB, EJB

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. 따라서 운영환경 통합은 온-디멘드로, 기반구조의 통합은 그리드나 유틸리티로, 개발통합은 통합개발 환경으로, 데이터베이스 통합은 데이터 허브나 EAI(Enterprise Application Integration)로, 사용자 인터페이스 통합은

*교신저자: 이명호(mhlee@semyung.ac.kr)
접수일 09년 02월 02일 수정일 09년 03월 19일 게재확정일 09년 03월 23일

RIA(Rich Internet Architecture)로 통합화 및 표준화가 진화되고 있다[3][4]. 이러한 엔터프라이즈 환경에서는 이기종 컴퓨터들 간에 프로그램을 분산시켜 부하를 줄여 시스템의 성능 저하와 네트워크 병목 현상을 줄일 수 있는 분산객체 구조가 필요하게 되었으며, 이를 해결하기 위한 컴포넌트 기반 개발(CBD:Component-Based Development) 방법론까지 이르게 되었다[6]. 여기에서 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 것이다.

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[1][3][4][9]. 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 직렬화(Serialization)하는 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 아키텍처(Lightweight Container Architecture)이다. 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 프레임워크(Spring Framework)이다. 따라서 본 연구에서는 Non EJB와 EJB 아키텍처가 가지고 있는 문제점을 해결하고 장점들을 지원하기 위하여 개발된 스프링 프레임워크 2.5를 기반으로 경량 컨테이너 아키텍처를 설계 및 구현하여 이전의 사양과의 객관적인 소프트웨어 개발 생산성 지침을 제공하고자 한다.

2. 스프링 프레임워크의 기본 개념

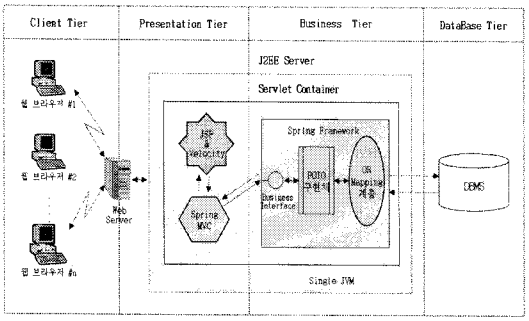
2.1 스프링 프레임워크의 고찰

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 프레임워크 2.5의 구성

스프링 프레임워크 2.5의 가장 큰 특징은 애노테이션을 이용한 의존성 삽입이다. 스프링이 시작한 의존성 삽입 기술은 피코 컨테이너, EJB3.0, SEAM, 구글 주스(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[2][8]. Constructor Injection은 생성자를 이용해서 의존성을 설정해주는 방법이고, Setter Injection은 Setter 메서드를 이용하여 의존성을 설정해 주는 방법이다. 스프링은 자바 빈 규칙을 이

용한 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 경량 컨테이너 아키텍처 환경에서 스프링 프레임워크 2.5 사양으로 파일럿 시스템을 구현한 구성도를 살펴보면 그림 1과 같다.



[그림 1] 스프링 프레임워크의 구성도

3. 비주얼 모델링의 분석 및 설계

3.1 개발 환경

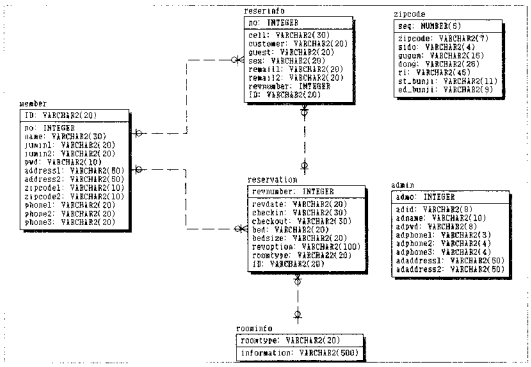
본 연구에서는 대용량 분산 객체 시스템을 개발하기 위한 스프링을 기반으로 표 1과 같은 개발환경을 이용하여 스프링 프레임워크 2.5 환경에서의 프로그램을 분석하고 설계한 후 파일럿 시스템을 구현하도록 한다.

[표 1] 스프링 프레임워크 2.5의 개발 환경

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

3.2 DB 스키마의 설계

대용량 분산 객체 처리를 위한 스프링 프레임워크 2.5 사양의 파일럿 시스템은 테이블로 영속적인 정보를 저장하기 위하여 데이터 모델이 먼저 정의되어야 한다. 따라서 엔티티 클래스를 기반으로 데이터 모델을 작성한다. 본 연구에서 파일럿 시스템으로 제시한 데이터 모델인 데이터베이스 스키마 구조는 그림 2와 같이 관리자정보, 회원정보, 예약정보, 예약자정보, 그리고 객실관리정보 등으로 설계하여 구현한다.



[그림 2] 데이터베이스 스키마 구조

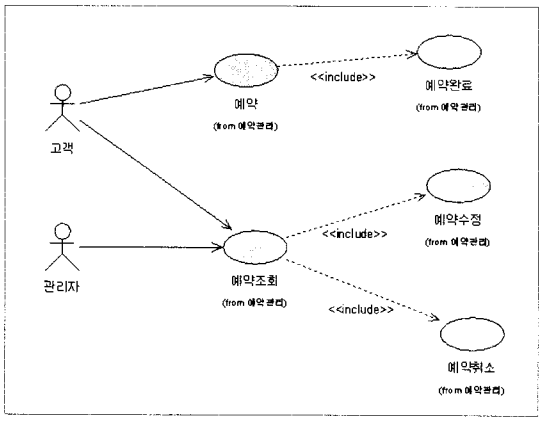
데이터베이스 스키마 구조에서 각 엔티티의 기능을 요약하면 표 2와 같다.

[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 유스케이스 다이어그램

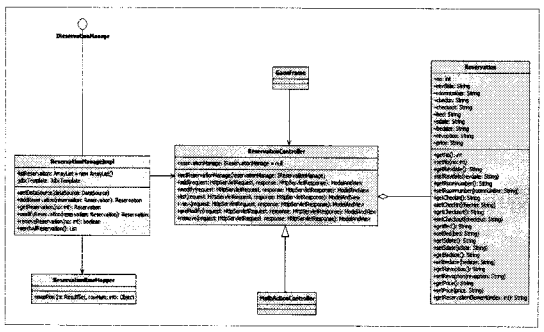
파이럿 시스템의 기본적인 요구사항을 기술한 문제 기술서와 유스케이스 명세서를 기반으로 예약관리에 대한 요구사항 정의활동으로 모델링한 결과인 유스케이스 다이어그램으로 표현해 보면 그림 3과 같은 유스케이스 모델이 된다.



[그림 3] 예약관리의 유스케이스 다이어그램

3.4 클래스 다이어그램

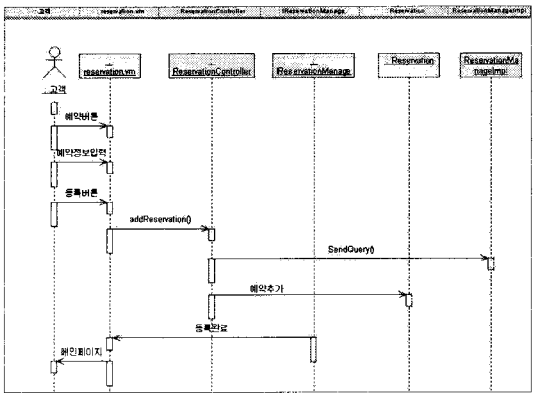
비기능적인 요구사항과 플랫폼을 고려한 후, 설계 활동을 통한 분석 클래스를 구체화하여 설계 클래스를 도출한다. 따라서 본 연구의 파일럿 시스템에서 중요한 예약관리의 설계객체 모델인 클래스 다이어그램은 그림 4와 같다.



[그림 4] 예약관리 클래스 다이어그램

3.5 시퀀스 다이어그램

위의 클래스 다이어그램은 시스템에서 어떤 클래스들이 있고, 각 클래스에서는 어떤 연산을 제공하고 있는가를 표현한다. 그러나 객체들 사이의 상호작용인 전달되는 메시지의 흐름은 시퀀스 다이어그램으로 표현된다. 따라서 본 연구의 파일럿 시스템에서 중요한 예약관리의 분석 유스케이스 실현 모델인 시퀀스 다이어그램은 그림 5와 같다.



[그림 5] 예약관리 시퀀스 다이어그램

4. 스프링 프레임워크 2.5의 평가

4.1 서비스 코드의 평가

일반적으로 소프트웨어 개발 생산성을 평가할 때 LOC 평가 방법을 자주 사용한다. 따라서 본 연구에서도 스프링프레임워크 2.5 사양에서의 패키지에 대한 LOC를 평가해 보면 표 3과 같다. 서비스 코드 설정 부분의 직접 선언은 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. Request Mapping을 클래스에 적용하게 되면 해당 클래스는 지정한 URL만을 처리하게 되기 때문에 메서드에 적용되는 RequestMapping 애노테이션은 더 이상 URL을 명시할 수 없는 단점도 존재한다.

[표 3] 인터페이스와 클래스 파일의 평가

Package명	설 정 사 항	LOC
Member	Auto-Wired, Component Controller 선언, Request-Mapping	243
Reservation	Auto-Wired, Component Controller 선언, Request-Mapping	195
ReservInfo	Auto-Wired, Component Controller 선언, Request-Mapping	173
Roominfo	Auto-Wired, Component Controller 선언, Request-Mapping	68
Admin	Auto-Wired, Component Controller 선언, Request-Mapping	192
Total (단위 : Line)		871

그러나 스프링 프레임워크 2.5에서의 서비스 코드 설정 시의 장점으로는 표 4와 같이 소스코드와 함께 설정을 할 수 있고, Field, Multi ActionController의 메서드 등의 편리한 설정 방식을 지원한다. 소스코드와 함께 있기에 리팩토링시 편리하다. 소스코드의 직접 설정으로 인한 XML의 설정의 간소화하며, 의존관계의 자동설정으로 인한 편의성이 증가한다. 또한 Controller 인터페이스를 구현하지 않은 클래스도 애노테이션을 이용하여 Controller로 사용이 가능하다.

[표 4] 서비스 코드인 Controller의 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);
    }
```

4.2 XML 평가

스프링에서는 스프링MVC의 DispatcherServlet에서 컨트롤러를 사용하여 클라이언트의 요청을 처리한다. 스프링MVC는 1개 이상의 DispatcherServlet을 설정할 수 있으며, 이것은 기본적으로 웹 애플리케이션의 /WEB-INF/ 디렉터리 에 위치한 [서블릿이름]-servlet.xml 파일로부터 스프링의 정보를 읽어온다. 서로 다른 DispatcherServlet이 공통 빈을 필요로 하는 경우에는 ContextLoaderListener를 사용하여 공통으로 사용될 빈을 설정할 수 있다.

ContextLoader Listener는 contextConfigLocation 컨텍스트 파리미터를 명시하지 않으면 /WEB-INF/applicationContext.xml을 설정 파일로 사용한다[7]. 따라서 본 연구에서 스프링 프레임워크 2.5에서 개발된 파일럿 시스템의 중요한 XML 현황은 표 5와 같다.

[표 5] 스프링 프레임워크 2.5의 XML 현황

항목 / XML	설정 항목	사용여부	LoC
Dispatcher Servlet	Component-Scan	○	1
	Handler Mapping	X	-
	Controller Bean 설정	X	-
	Schema Type	○ (2.5)	8 (추가확장기능)
Total(단위 : Line)		9	
Application	Component-Scan	○	5
Context	Resource설정 및 Bean 등록	X	-
	Schema Type	○ (2.5)	8
	Annotation Config	○	1
Total(단위 : Line)		14	

4.2.1 servlet.xml 설정 방식

servlet.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">
```

HandlerMapping 설정을 살펴보면, BeanNameUrl, SimpleUrl, AbstractUrl, AbstractBeanNameUrl Mapping, DefaultAnnotationHandler Mapping 방식 중 한 가지를 설정한다. 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.DefaultAnnotationHandler
Mapping" p:alwaysUseFullPath="true" />
```

Component-Scan 설정을 살펴보면, controller 패키지를 스캔 범위로 지정해 줌으로써 컴포넌트 선언 컨트롤러 Bean을 자동 등록되게 한다.

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

4.2.2 applicationcontext.xml 설정 방식

applicationcontext.xml Schema 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">
```

또한 Component-Scan 설정을 살펴보면, 서비스 패키

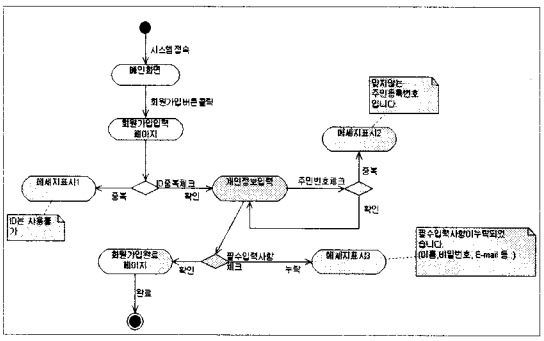
지를 스캔 범위로 지정해 줌으로써 컴포넌트 선언 서비스 빈의 자동 등록 및 Resource 설정을 하도록 한다.

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

이상과 같이 스프링 프레임워크 2.5에서 이전 버전들보다 XML 스키마 형(Schema Type)의 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.

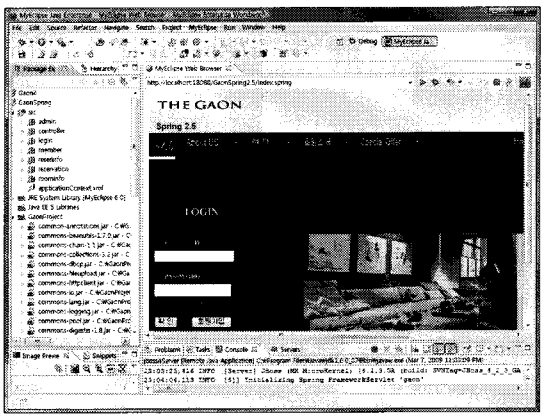
5. 파일럿 시스템의 구현

경량 컨테이너 아키텍처 환경에서 스프링 프레임워크 2.5의 파일럿 시스템은 각 유스케이스에 사용되는 화면 간의 전환을 화면 흐름 모델로 설계함으로써 명시적으로 분석하였다. 다음 그림 6은 회원가입 관련 화면 흐름 모델과 관련된 사례로써 예약관리 유스케이스의 사용자 인터페이스를 위하여 사용되는 화면 흐름 관계를 나타낸다.



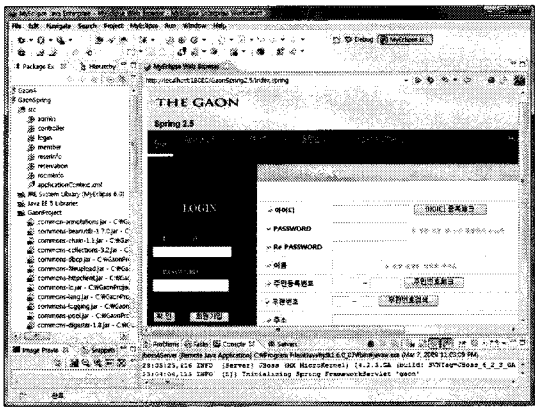
[그림 6] 회원가입을 위한 화면 흐름도

이상과 같은 개발 환경과 데이터베이스 스키마를 기반으로 스프링 프레임워크 2.5 환경에서 호텔예약시스템을 구현한 메인 화면은 그림 7과 같다.



[그림 7] 호텔예약시스템의 메인 화면

호텔예약시스템 중에서 구현된 회원가입을 위한 실행 화면을 살펴보면 다음 그림 8과 같다.



[그림 8] 회원가입의 실행 화면

6. 결 론

스프링 프레임워크에서는 특정한 인터페이스에 종속되지 않는 Bean과 같은 클래스인 POJO(Plain Old Java Object)를 관리하는 스프링 컨테이너에게 제어 역행화(IoC:Inversion of Control)를 통한 제어권을 넘겨서 EJB 컨테이너에서 지원하던 매력적인 기능들을 지원하고 있다. 그리고 POJO 기반이기 때문에 특정 환경구축을 위한 클래스를 Import하지 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 2.5 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 스프링 프레임워크 2.5를 기반으로 파일럿 프로젝트의 분석 및 설계를 통하여 구현 지침을 제시하였으며, 또한 스프링 프레임워크 2.5에 대한 성능 평가 기반으로 정량적인 분석을 통하여 객관적인 소프트웨어 개발 생산성 연구에 대한 지침을 제시하였다. 향후에는 AOP(Aspect Oriented Programming)나 EJB 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] 김병곤, "Enterprise Java Beans 3.0," 가메출판사, pp. 26-340, 2006.

[2] 박재성, "Spting 프레임워크 워크북," 한빛미디어, pp. 26-377, 2006.

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

[4] 이명호, "EJB2.0과 EJB3.0의 소프트웨어 개발 생산성 비교 연구", 한국산업경영시스템학회지, 제31권 제3호, pp. 1-7, 2008.

[5] 이일민, "자바 기술의 미래를 비추는 거울 스프링 프레임워크 2.5," 마이크로소프트웨어, pp. 136-143, 2008.

[6] 채홍석, "객체지향 CBD 개발 Bible," 한빛미디어, pp. 35-76, 2006.

[7] 최범균, "웹 개발자를 위한 스프링 2.5 프로그래밍," 가메출판사, pp. 24-440, 2008.

[8] Road Johnson, "Expert One-on-One J2EE Design and Development", Wrox, pp. 441-673, 2002.

[9] John Steams, Roberto 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