웹 어플리케이션의 통합 테스트에서 인터페이스 분석

장진아 O, 서주영, 최병주, 박종빈*
이화여자대학교 컴퓨터학과, 삼성 SDS*
{jajang O, jyseo}@ewhain.net, bjchoi@ewha.ac.kr, bhin.park@samsung.com*

Interface Analysis of Web Applications in Integration Test

Jina Jang O, Jooyoung Seo, Byoungju Choi, Jongbhin Park*
Dept. of Computer Science and Engineering, Ewha Womans University, Samsung SDS.*

요 약

최근 웹 어플리케이션이 대형화되고 복잡해지면서 품질의 중요성은 더욱 높아지고, 신뢰도 높은 웹 어플리케이션의 개발을 위해 웹 프레임워크를 기반으로 한 개발이 일반적인 추세다. 이러한 개발 환경에서는 이질적인 프레임워크가 서로 올바르게 연동하는지를 평가하는 통합 테스트가 필수적이며, 이를 지원하기 위해 적절한 통합 테스트 항목을 식별하는 것이 중요하다. 본 논문에서는 웹 어플리케이션의 통합 테스트 단계를 위해 대표적인 웹 프레임워크 분석을 통해 통합 테스트 대상인 계층을 식별하고, 핵심 테스트 항목이 되는 인터페이스를 정의하였다.

1. 서론

최근 인터넷 사용 인구와 통신 기술의 급성장으로 웹 어플리케이션은 발전과 변화를 거듭하고 있다. 웹 기반 어플리케이션은 시간과 공간의 제약이 없는 사용성(anytime anywhere)이라는 웹의 강력한 장점을 바탕으로 비즈니스와 산업 전반에서 활발하게 사용되고 있다. 웹 어플리케이션의 수요가 늘어나고 웹 어플리케이션이 비즈니스 로직의 핵심 역할을 하면서, 웹 어플리케이션은 갈수록 다양한 기능을 갖는 복잡한 시스템이 되고 있다. 웹 어플리케이션이 대형화되고 복잡해지면서 품질의 중요성은 더욱 높아지고 있다 [1]. 따라서 신뢰도 높은 웹 어플리케이션을 효율적으로 개발하기 위해 다계층 아키텍처(Multi-tier Architecture)를 도입하고, 재사용되고 품질이 보증된 웹 어플리케이션 프레임워크(Web Application Framework)를 기반으로 개발하는 것이 일반적 추세이다 [2].

웹 어플리케이션 프레임워크는 어플리케이션의 기본 골격과 품질이 보증된 코드의 재사용을 통해 비즈니스 시스템 개발에 편리한 환경을 제공한다. 이러한 환경에서는 프레임워크가 제공하는 기능을 구현한 클래스나 재사용 단위인 컴포넌트를 대상으로 하는 단위 테스트보다는, 서로 다른 클래스나 컴포넌트 계층들이 통합되는 과정에서 오류가 발생하는지, 이질적인 프레임워크가 서로 연동하며 올바르게 원하는 기능을 수행하는지를 테스트 할 수 있는 통합 테스트가 전체 비즈니스 시스템의 품질 정량화를 위해 필수적이다. 특히 급변하는 웹 어플리케이션을 효율적으로 테스트하기 위해 웹 어플리케이션의 특징을 반영한 통합 테스트 항목의 선정이 매우 중요하다.

본 논문에서는 웹 어플리케이션의 통합 테스트를 위해 웹 어플리케이션의 대표적인 프레임워크를 분석해서 테스트 케이스 선정의 기반이 될 인터페이스를 정의한다. 즉 제안하는 웹 어플리케이션을 위한 인터페이스란, 비즈니스 로직 상의 이질적 계층 간 상호 협력하는 부분으로, 이들 계층의 통합 시 발생하는 결함 식별의 기준이 된다 [3]. 본 논문은 대표적인 웹 어플리케이션 프레임워크인 Spring과 Struts, Hibernate에 기반한 웹 어플리케이션을 분석 대상으로 하며, 이 때 웹 어플리케이션 프레임워크는 이들 프레임워크를 단일 오픈 프레임워크로 합친 삼성 SDS의 Anyframe JAVA [4]를 대상으로 한다.

본 논문은 다음과 같이 구성된다. 2장에서는 웹 어플리케이션의 특징과 테스트의 어려움에 대해 알아보고, 3장에서는 웹 어플리케이션의 아키텍처를 살펴본다. 4장에서는 3장에서 분석한 웹 어플리케이션 아키텍처를 기반으로 통합의 대상인 계층과 인터페이스를 정의하며, 5장에서 결론 및 향후 연구방향을 기술한다.

2. 웹 어플리케이션 테스트

웹 어플리케이션은 전통적인 테스트 기법의 적용을 어렵게 만드는 다음과 같은 특징들이 있다 [5]:

첫째, 웹 어플리케이션은 다수의 사용자가 이용하는 서비스이기 때문에 동시다발적인 사용자의 요청을 처리할 수 있어야 한다.

둘째, 웹 어플리케이션은 다양한 클라이언트 환경에서 사용하는데 적합해야 한다.

셋째, 웹 어플리케이션은 콘텐츠 뿐만 아니라 기술적인 측면에서 일반적인 소프트웨어에 비해 빠른 업데이트가 요구된다.

따라서, 웹 어플리케이션을 효과적으로 테스트하기 위해서는 이러한 웹 어플리케이션 특징을 반영하는 테스트 방안이 필요하다. 즉, '빈번한 사용자 요청 처리, 다양한 클라이언트 환경 지원, 잦은 업데이트에 의한 변경'과 같은 웹 어플리케이션의 특징에 따라 테스트되어야 할 계층이 구조적으로 식별되어야 하며, 이들의 상호작용 위치인 인터페이스가 반드시 테스트되어야 한다.

그러나 웹 어플리케이션 테스트에 관한 기존 연구들은 전통적인 테스트 방법을 웹 어플리케이션에 적용한 것이 대부분이다 [6].

웹 어플리케이션의 특징을 반영한 경우도 주로 코드 기반 단위 테스트나 네트워크와 데이터베이스에 대한 부하 테스트와 같은 성능 테스트에 초점이 맞춰져 있다 [7]. 그러나 웹 어플리케이션의 크기와 복잡도가 증가함에 따라 알고리즘이나 자료구조의 선택보다 전체 어플리케이션 시스템의 구조를 결정하는 일이 더욱 중요해졌고, 특히 웹 어플리케이션은 다계층 아키텍처와 웹 어플리케이션 프레임워크를 기반으로 기본 골격과 코드를 재사용하여 구현되기 때문에, 단위 테스트 보다는 거시적인 관점으로 서로 다른 계층을 통합하는 과정에서 오류가 발생하지 않는지, 각 계층을 구현하는 이질적인 프레임워크가 서로 연동하며 올바르게 원하는 기능을 수행하는지를 테스트 할 수 있는 통합 테스트가 무엇보다 중요하다. 그러나 웹 어플리케이션의 통합 테스트에 관련된 연구는 많지 않다.

본 논문에서는 위와 같은 문제를 해결하기 위하여 웹 어플리케이션의 특징에 의해 계층화되는 웹 어플리케이션 아키텍처를 분석하고, 이들 계층을 핵심 통합 대상으로 식별하며, 이들 사이의 상호작용을 통합 테스트 단계에서 관심을 가지고 살펴봐야 하는 테스트 항목인 인터페이스로 제안한다.

3. 웹 어플리케이션 아키텍처

웹 어플리케이션의 크기와 복잡도가 증가함에 따라 자체 알고리즘이나 자료구조의 선택보다 전체 어플리케이션 시스템에 적합한 아키텍처를 선정하고 이해하는 일이 핵심적인 개발 이슈가 되고 있다. 많은 웹 어플리케이션 시스템이 프로그램의 일관성을 유지하고, 성능 및 품질을 향상 시키기 위해 다계층 아키텍처(Multi-tier Architecture)를 응용해서 웹에 적용한 모델 2 접근법을 도입하고, 웹 어플리케이션 프레임워크(Web Application Framework)를 사용하고 있다.

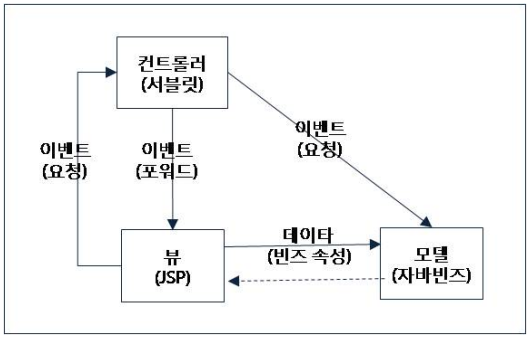
본 장에서는 일반적인 웹 어플리케이션 아키텍처를 이해하기 위해 모델 2 접근법을 알아보고, 가장 보편적으로 사용되는 계층화된 웹 어플리케이션 아키텍처를 살펴본다. 특히 본 논문은 Spring과 Struts, Hibernate를 사용하는 오픈 프레임워크인 Anyframe JAVA를 기반으로 웹 어플리케이션이 갖는 이질적 계층들을 식별하고 이들 사이의 인터페이스를 정의한다. 사실 웹 어플리케이션은 다양한 구조로 구현할 수 있기 때문에 모든 웹 어플리케이션을 포괄하는 인터페이스 분석은 현실적으로 불가능하다. 따라서 본 논문에서는 가장 일반적으로 사용되는 프레임워크인 Spring과 Struts, Hibernate를 기반으로 구현한 웹 어플리케이션을 테스트 대상으로 한정하고 아키텍처를 분석하였다.

3.1 모델 2 접근법

모델 2 접근은 모델-뷰-컨트롤러 디자인 패턴(Model-View-Controller Design Pattern, 이하 MVC)을 응용해서 웹에 적용한 것이다. MVC는 웹 어플리케이션의 설계방법을 표현하는 모델로써, 어플리케이션을 모델, 뷰, 컨트롤러 계층으로 각각의 역할을 나눠서 작업하고자 하는 일을 분담하는 구조이다. 이러한 구조를 통해 설계, 개발의 분업화를 이룰 수 있으며 각 계층간 의존도를 최소화하여 어플리케이션 변경 시에 유연하게 대처할 수 있다. 웹 어플리케이션에서 MVC는 다음과 같이 적용된다.

- 모델 계층 : 비즈니스 로직의 구현을 통한 서비스 제공 및 데이터베이스와의 연동을 담당하는 부분으로 자바빈즈를 이용한다.
- 뷰 계층 : 유저 인터페이스를 담당하는 부분으로 JSP를 이용한다.
- 컨트롤러 계층 : 모델과 뷰 계층 사이에서 데이터를 전달하는 역할을 담당하는 부분으로 서블릿을 이용한다.

MVC를 웹 어플리케이션에 적용한 모델 2 구조는 [그림 1]과 같다.



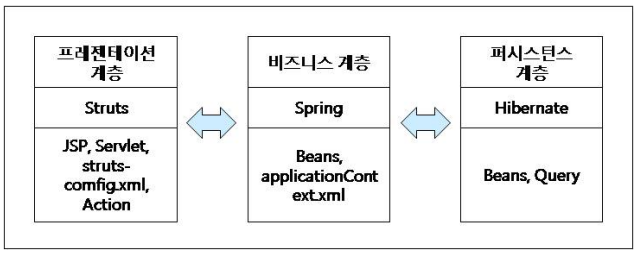
[그림 1] 모델 2 구조

3.2 계층화한 웹 어플리케이션 아키텍처

대규모의 웹 어플리케이션은 모델 2 개발 방식에 따라 계층화(Layering)하여 개발하는 것이 일반적이다. 보통 웹 어플리케이션은 프레젠테이션 계층(Presentation layer), 비즈니스 계층(Business layer), 퍼시스턴스 계층(Persistence layer)의 3개 계층으로 나눈다 [8].

- 프레젠테이션 계층 : 사용자의 요청을 받고, 어플리케이션의 처리 결과를 보여주는 모델 2 접근법의 뷰 계층에 해당하는 일을 한다. 정보를 처리하는 일에는 관여하지 않으며, 정보 자체를 사용자에게 보여줄 뿐이다. 또한 사용자의 요청을 비즈니스 계층으로 전달하는 모델 2 접근법의 컨트롤러 계층에 해당하는 일도 수행한다.
- 비즈니스 계층 : 사용자가 원하는 동작을 실제로 처리하는 모델 2 접근법의 모델 계층에 해당한다. 필요한 비즈니스 로직을 구현하고 프레젠테이션 계층의 요청에 서비스를 제공한다.
- 퍼시스턴스 계층 : 데이터베이스와 연동을 담당하는 모델 2 접근법의 모델 계층에 해당하는 부분이다.

이 때, 각 계층은 완전히 독립적이어야 하며, 일반적으로 각 계층 사이에서는 인터페이스를 통하여 통신한다. 이렇듯 계층화된 웹 어플리케이션의 아키텍처는 [그림 2]와 같다.



[그림 2] 계층화한 웹 어플리케이션 아키텍처

[그림 2]의 웹 어플리케이션 아키텍처에 따라 개발을 효율적으로 수행하도록 지원하는 수단의 하나가 웹 어플리케이션 프레임워크다. 프레임워크를 사용하면 개발자는 업무고유의 처리와 화면을 작성하는 것만으로도 편리하게 웹 어플리케이션을 완성할 수 있다.

이러한 구조를 지원하는 가장 보편적으로 사용되는 프레임워크는 Spring과 Struts, Hibernate이 있다. 본 논문에서는 이들 프레임워크를 하나로 합쳐서 오픈 프레임워크 형태로 제공하는 Anyframe JAVA를 대상으로 웹 어플리케이션 아키텍처를 분석하였다.

Spring은 대규모의 어플리케이션을 효율적으로 개발 및 유지보수 하기 위해 만들어졌다. Spring은 어플리케이션 프레임워크로써 웹에 국한된 프레임워크가 아닌 모든 어플리케이션을 대상으로 하는 프레임워크이다. 따라서 웹 어플리케이션에서도 프레젠테이션 계층 보다는 비즈니스 계층에 더 관심을 두어 발전하고 있다.

반면 Struts는 모델 2 접근법에 따라 만들어진, 모델, 뷰, 컨트롤러의 3가지 주요 컴포넌트를 갖는 순수한 웹 프레임워크다. 즉 Struts는 프레젠테이션 계층을 표현하는데 더 많이 사용된다.

따라서 웹 어플리케이션 아키텍처를 구축하는데 Spring과 Struts, Hibernate을 함께 사용할 수 있으며, 일반적으로는 Spring으로 비즈니스 계층을 구현하고 Struts로 JSP 계층과 프레젠테이션 계층을 개발한다. 퍼시스턴스 계층은 Spring의 하위기능 또는 Hibernate를 사용해서 구현할 수 있다.

4. 웹 어플리케이션 인터페이스 테스트

4.1 인터페이스 테스트

본 논문에서는 웹 어플리케이션의 통합 테스트 항목 선정을 위해 인터페이스 기반 테스트 방법을 사용한다.

본 논문에서 정의하는 인터페이스 기반 테스트 방법은 일반적으로 이질적인 계층 사이의 함수 호출과 리턴의 단순 상호작용 만을 의미하는 것이 아니라 전역 변수를 통한 데이터 전달, 어플리케이션 프로그램 인터페이스(Application Program Interface), 데이터베이스 쿼리 등을 포함한다. 즉, 보다 넓은 의미에서 서로 다른 계층 간에 어떠한 형태로든 상호관계를 갖는 부분을 인터페이스라 정의하고, 이를 통합 테스트 항목으로 삼는 기법을 의미한다.

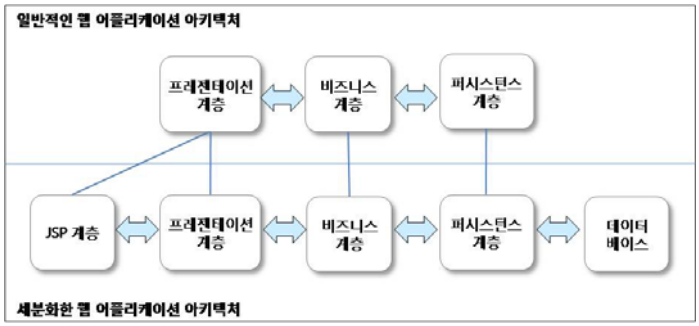
본 연구실에서는 이러한 인터페이스 기반 테스트 기법을 '어플리케이션, 운영체제 커널, 디바이스 드라이버, HAL(Hardware Abstraction Layer), 하드웨어'의 이질적 계층이 밀접히 연관되고 최적화되는 특징을 갖는 임베디드 시스템의 소프트웨어 테스트 기준으로 정의하였고, 이를 자동화 테스트 도구인 Justatia [9]를 통해 제안하는 인터페이스 테스트의 중요성과 실현 가능성을 보인

바 있다. 본 논문에서는 임베디드 소프트웨어 테스트에 적용했던 인터페이스 테스트의 핵심 이론을 '프레젠테이션, 비즈니스, 퍼시스턴스, 데이터베이스'와 같은 이질적 계층을 갖는 Anyframe JAVA를 기반으로 하는 웹 어플리케이션에 확장한다.

4.2 세분화한 웹 어플리케이션 아키텍처

본 논문에서는 웹 어플리케이션의 계층 간 통합이 일어나는 위치를 테스트 관점에서 식별하기 위해 3장에서 살펴본 일반적인 웹 어플리케이션 아키텍처를 다음과 같이 5개의 이질적인 계층으로 보다 세분화 하였다.

- JSP 계층 : 사용자의 요청을 받고 응답을 내보내는 직접적인 통로로 사용되는 JSP 계층을 프레젠테이션 계층으로부터 분리하였다. JSP 계층은 화면 상의 처리를 담당할 뿐만 아니라 사용자의 입력과 요청을 받아들이고 결과를 알려주는 UI로써의 기능을 함께 수행한다.
- 프레젠테이션 계층 : 기존의 프레젠테이션 계층을 중 비즈니스 로직과의 연동을 담당하는 부분을 프레젠테이션 계층으로 구분한다.
- 비즈니스 계층 : 필요한 비즈니스 로직을 구현하고 프레젠테이션 계층의 요청에 서비스를 제공하는 기존의 역할을 담당한다.
- 퍼시스턴스 계층 : 추가적으로 식별한 데이터베이스 계층과 연동하는 기능을 수행한다.
- 데이터베이스 계층 : 웹 어플리케이션의 내부구성요소는 아니지만 퍼시스턴스 계층과 직접적으로 연동하는 자원인 데이터베이스를 간접적인 웹 어플리케이션 구성요소로 인정하여 웹 어플리케이션 아키텍처에 함께 표현한다.



[그림 3] 세분화한 웹 어플리케이션 아키텍처

웹 어플리케이션의 계층화 아키텍처는 빈번한 사용자 요청 처리, 다양한 클라이언트 환경 지원, 잦은 업데이트에 의한 변경과 같은 웹 어플리케이션의 단점을 최소화 시키는 구조다. 웹 어플리케이션을 기능별로 나누고 이를 계층 단위로 구현함으로써 처리 속도와 효율을 높이고, 수정 및 변경으로 인한 영향을 최소화 하였다. 따라서 웹 어플리케이션의 특징을 반영한 위의 구조를 바탕으로 계층 간의 상호작용 위치인 인터페이스를 분석하고자 한다.

4.3 웹 어플리케이션의 인터페이스

(1) 웹 어플리케이션의 비즈니스 로직

웹 어플리케이션은 기존의 소프트웨어와는 달리 사용자가 어플리케이션을 접하는 환경은 JSP화면에 해당하는 웹 브라우저로 국한되며, 개발자가 사전에 사용자의 입력을 제한해 어플리케이션이 정확한 동작을 하는 흐름만을 선택하도록 제약을 두고 있다. 즉, 웹 어플리케이션은 사용자의 입력을 받는 JSP 화면 단위의 비즈니스 로직을 하나의 실행 흐름으로 한다. [그림 4]와 같이 JSP 화면에서 시작한 사용자 입력은 프레젠테이션 계층과 비즈니스 계층, 퍼시스턴스 계층을 거쳐 데이터베이스로 연결되는 일련의 실행 흐름을 갖는다. 이러한 JSP 화면 단위의 실행 흐름을 하나의 비즈니스 로직이라 정의하며, 이 때 비즈니스 로직 상의 서로 다른 계층 사이의 인터랙션을 통합 테스트 항목인 인터페이스로 식별한다.

[그림 4] 웹 어플리케이션의 비즈니스 로직

(2) 웹 어플리케이션의 인터페이스

본 논문은 웹 어플리케이션을 위한 인터페이스를 비즈니스 로직 상의 이질적인 계층 간 통합 시 발생하는 결함을 찾기 위해 식별한 계층 간 상호 협력하는 부분으로 정의한다. 인터페이스 정의에 따라 앞서 3장의 웹 어플리케이션 아키텍처에 기반해 인터페이스를 분석한 결과는 다음과 같다.

- JSP와 프레젠테이션 계층의 통합
- 프레젠테이션 계층과 비즈니스 계층의 통합
- 비즈니스 계층과 퍼시스턴스 계층의 통합
- 퍼시스턴스 계층과 데이터베이스의 통합

따라서 웹 어플리케이션의 통합 테스트 항목은 위에서 분석한 4개의 통합에서 존재하는 인터페이스가 된다. 웹 어플리케이션을 구성하는 이질적인 계층 간의 오류 없는 통합 여부는 웹 어플리케이션의 성공적인 통합을 위해 반드시 점검해야 한다. 즉, 이들 인터페이스는 웹 어플리케이션을 위해 핵심적으로 테스트되어야 할 위치이며, 테스트 커버리지의 기준이 되어야 한다.

4. 결론 및 향후 과제

웹 어플리케이션이 비즈니스 로직의 핵심 역할을 하면서, 웹 어플리케이션은 갈수록 많은 기능을 가진 복잡한 시스템이 되고 있다. 웹 어플리케이션이 대형화되고 복잡해지면서 품질의 중요성은 더욱 높아지게 되었으며 신뢰도 높은 웹 어플리케이션을 개발하기 위해 웹 프레임워크를 사용하고 있다. 이러한 개발 환경 하에서는 이질적인 프레임워크가 서로 올바르게 연동하는지를 평가하는 통합 테스트가 필수적이며, 이를 지원하기 위해 적절한 통합 테스트 항목을 식별하는 것이 중요하다.

따라서 본 논문에서는 통합 테스트 단계에서 테스트 항목이 되는 인터페이스를 분석하였다. 웹 어플리케이션의 인터페이스를 분석하기 위해, 우선 웹 어플리케이션의 특징을 살펴보고, 웹 어플리케이션의 인터페이스를 식별하기 위한 기반이 되는 웹 어플리케이션의 아키텍처를 프레임워크를 기반으로 분석하였다. 마지막으로 웹 어플리케이션에서 서로 다른 계층 간 통합이 일어나는 부분을 정의하고 이를 인터페이스로 식별하였다.

향후에는 본 논문에서 제안한 인터페이스 분석을 실제 웹 어플리케이션의 통합 테스트 단계에 적용하여 수행하고, 이 때 계층 간 존재하는 인터페이스가 전체 통합 테스트 항목 중에서 차지하는 비율 및 커버리지 측정 연구를 수행할 예정이다.

5. 참고 문헌

[1] Chien-Hung Liu et al., "*Object-Based Data Flow Testing of Web Applications*", Proceedings. First Asia-Pacific Conference on, 2000, 1(1):7-16

[2] Ezra Ebner, Weiguang Shao and Wei-Tek Tsai, "*The five-module framework for Internet application development*", ACM Computing Surveys (CSUR), 2000, 32(1es)

[3] Ahyoung Sung, Byoungju Choi, Seokkoo Shin, "*An Interface Test Model for Hardware-dependent Software and Embedded OS API of the Embedded System*", the Computer Standard & Interface journal, 2007, 29(4):430-443

[4] Samsung SDS, "Anyframe JAVA", 2008

[5] Kallepalli, C., Tian, J., "*Measuring and modeling usage and reliability for statistical Web testing*", Software Engineering, IEEE Transactions on, 2001, 27(11):1023-1036

[6] David Chenho Kung, "*An Object-Oriented Web Test Model for Testing Web Applications*", 24 th International Computer Software and Applications Conference, 2000, 1(1): 537-542

[7] Arabi Keshk, Amal Ibrahim, "*Ensuring the Quality Testing of Web Using a New Methodology*", IEEE International Symposium on Signal Processing and Information Technology, 2007, pp.1071-1076

[8] Ezra Ebner, Weiguang Shao, Wei-Tek Tsai , "*The five-module framework for Internet application development*", ACM Computing Surveys(CSUR), 2000, 32(1es)

[9] Samsung Elec. Co., Ltd.,, "*Technical Report of Emulation Test Tool for Linux and ARM based Embedded Software Interface Coverage*", 2007