EJB 컴포넌트 테스트를 위한 Deployment Descriptor 기반의 테스트케이스 자동 생성 방법

국승학° 김현수
충남대학교 전기정보통신공학부 컴퓨터 전공
{triple888°, hskim401}@cnu.ac.kr

An Automatic Generation Method of Test Cases from the Deployment Descriptor for Testing EJB Components

Seung-hak Kuk°, Hyeon Soo Kim
Dept. of Computer Science and Engineering, Chungnam National University

요 약

최근 산업계에서는 급변하는 사용자의 요구사항을 반영하면서, 경쟁력 있고, 경제적인 소프트웨어를 개발하기 위해서 EJB 컴포넌트 개발 방법을 채택하고 있다. 그러나 EJB 컴포넌트를 테스트하는 것이 기존의 자바 클래스의 테스트보다 더 많은 노력이 필요하고, 어려운 작업이기 때문에 많은 경우에 EJB 컴포넌트에 대한 테스트가 수행되지 않는다. 이에 본 논문에서 EJB 컴포넌트의 테스트 케이스 생성을 자동화 할 수 있는 방법을 제안한다. 이는 EJB 컴포넌트 개발 시 만들어지는 Deployment Descriptor로부터 컴포넌트에 대한 정보를 추출해내고, 이를 기반으로 테스트 케이스를 자동으로 생성하는 방법이다.

1. 서 론

EJB(Enterprise Java Beans)는 웹 응용 서버 스펙인 J2EE(Java2 Enterprise Edition)의 핵심으로서, 비즈니스 업무를 웹 환경에서 컴포넌트 형태로 작성하여 재사용성을 높이기 위한 서버 측 컴포넌트 프로그래밍 모델이다[1]. 최근 산업계는 급변하는 사용자의 요구사항을 반영하면서, 경제적이고, 경쟁력 있는 소프트웨어를 개발하기 위해 EJB 컴포넌트 개발 방법을 채택하고 있다. 이는 EJB의 스펙을 따르는 컴포넌트는 어느 곳에서나 동작한다는 것과, EJB 컨테이너에서 보안과 트랜잭션 처리와 같은 많은 서비스를 제공하기 때문에 개발자는 비즈니스 로직 개발에만 집중할 수 있기 때문이다[1]. 그러나 많은 경우에 EJB 컴포넌트 개발자들은 EJB 컨테이너에서 제공하는 많은 서비스를 이용할 뿐 정확히 동작하는지 테스트하지 않는다. 이는 EJB 컴포넌트에 대한 테스트를 수행하는 것이 EJB 컴포넌트를 개발하는 것보다 더 많은 노력을 요구하기 때문이다. EJB 컴포넌트의 테스트가 일반 자바 클래스의 테스트에 비해 더 복잡하고 어려운 작업이다[2][5][6].

본 논문에서는 EJB 컴포넌트의 테스트를 위한 테스트케이스 자동생성 방법 제안하고, 도구를 설계한다. 2장에서는 EJB 컴포넌트 테스트 전략과 기존의 EJB 컴포넌트의 테스트케이스 자동 생성 방법 및 문제점에 대해 언급하고, 3장에서 본 논문에서 제안하는 Deployment Descriptor 기반의 테스트케이스 자동 생성 기법에 대해서 설명한다. 또한 4장에서는 결론 및 향후 연구 과제에 대해 설명한다.

2. 관련연구

2.1 EJB 컴포넌트 테스트 전략

일반적인 EJB 컴포넌트 테스트 전략은 크게 3가지로 나뉘어진다[1].

- Testing in Isolation : 컨테이너나 다른 EJB의 Stub을 생성해 완벽한 독립적인 환경에서 테스트를 수행할 수 있는 전략이다. EJB는 컨테이너 서비스와 다른 EJB 또는 DB와 같은 자원들에 의존적이다. 이러한 의존성 때문에 완벽한 단위 테스트를 수행하는 것은 극히 어려운 일이다. 이러한 어려움을 극복하기 위해 가장 많이 사용되는 방법이 Mock Object를 이용하는 것이다. 현재 개발 중인 MockEJB 프레임워크는 컨테이너나 다른 EJB들의 Stub을 손쉽게 만들어 주는 기능을 제공함으로써 독립적인 환경에서 EJB의 단위 테스트를 쉽게 수행할 수 있다. 그러나 이 전략은 컨테이너나 EJB, DB와 같은 자원에 대한 많은 Mock Object를 생성해 주어야 하며, 독립적인 환경에서 테스트를 성공적으로 수행했다고 하더라도 실제 컨테이너에서 올바로 동작한다고 보장할 수 없다는 단점이 있다.

- In-Container Testing : 컨테이너 내부 혹은 어플리케이션 서버 내에서 테스트하는 방법은 어플리케이션 서버 내에서 실행하는 테스트를 작성하고 배치하는 것이다. 이것은 또한 로컬 인터페이스를 갖는 EJB를 테스트하는데 알맞은 방법이다. 기존의 Cactus나 JUnitEE와 같은 테스트 프레임워크가 널리 사용된다. 그러나 이 전략은 컨테이너 내에서 테스트를 수행할 수 있도록 도와주는 추가적인 테스트 프레임워크가 필요하며,

구현, 배치, 그리고 테스트를 위한 호출이 좀 더 복잡해진다.

- Out Side the Container Testing : Bean이 원격 인터페이스를 갖는 경우에 원격 클라이언트를 작성함으로써 테스트 할 수 있는 가장 간단한 방법이다. 이 전략은 테스트의 작성과 실행이 쉽고, JUnit과 같은 테스트 인프라를 쉽게 이용할 수 있다. 또한 이러한 방법을 통해 완전한 원격 인터페이스를 제공하는 것을 확인할 수 있으며, 분산 환경에서 원격 인터페이스를 통해 비즈니스 로직을 접근할 경우에 적합한 테스트 방법이라 할 수 있다. 그러나 이 전략은 컴포넌트가 로컬 인터페이스를 갖는 경우에 이를 테스트 할 수 없는 단점이 있다.

2.2 기존 연구 방법과 문제점

기존의 EJB 컴포넌트 테스트케이스의 자동생성 방법은 크게 스펙 기반의 방법과, 소스코드 기반의 방법으로 나눌 수 있다. 소스코드 기반의 테스트케이스 생성 방법은 테스트 대상 소스코드를 분석해 컴포넌트의 클래스 정보를 추출하고 이를 바탕으로 테스트케이스를 생성하는 방법이다. 이를 통해 블랙박스 테스트와 화이트박스 테스트를 수행할 수 있다. 논문[3]의 경우 이러한 소스코드 기반의 방법을 이용해 컴포넌트의 성능측정 방법을 제안하고 있다. 그러나 코드 기반의 테스트케이스 생성 방법은 소스코드를 분석할 수 있는 분석 도구의 작성이 필요할 뿐 아니라 실제 소스코드가 제공되지 않는 상용 컴포넌트를 테스트 할 수 없다. 반면에 스펙 기반의 테스트케이스 생성 방법은 컴포넌트의 스펙이 주어진다면, 어떤 컴포넌트든지 테스트케이스를 생성할 수 있다[4]. 대부분의 상용 컴포넌트에서도 컴포넌트의 스펙을 기술한 문서를 제공한다. 이 스펙에는 해당 컴포넌트의 인터페이스 정보를 담고 있고, 이를 분석해 테스트케이스를 생성할 수 있다. 논문[4]의 연구는 일반적인 컴포넌트에서 제공되는 스펙을 기반으로 테스트 랩퍼를 생성하고 이를 이용한 테스트 방법을 제안하고 있다. 그러나 이 방법은 컴포넌트의 스펙이 기술되어 있다는 가정 아래 테스트케이스를 생성하는 방법을 설명하고 있을 뿐, 스펙을 작성하는 방법 등에 대해서는 정확히 언급하지 않고 있고, 컴포넌트의 스펙이 표준화 되어있지 않은 상태에서 모든 컴포넌트에 적용하기 어렵다는 단점이 있다. 또한 컴포넌트의 외부로 드러난 인터페이스만을 테스트하는 Out Side the Container 방법을 취하고 있기 때문에 테스트에 제약이 있다. 이에 본 논문에서는 테스트 대상이 되는 컴포넌트를 EJB 컴포넌트로 한정하고, EJB의 컴포넌트를 생성할 때 자동으로 생성되는 스펙인 Deployment Descriptor를 기반으로 모든 인터페이스에 대한 테스트를 수행 할 수 있는 테스트케이스를 생성하는 구체적인 방법을 제안한다.

3. Deployment Descriptor 기반의 테스트케이스 자동생성 기법

EJB는 Bean을 정의하는 자바 클래스와 더불어 Deployment Descriptor라는 설정 파일을 가지고 있다. 이 Deployment Descriptor를 통해 컴포넌트 개발자는 Bean 클래스의 정보와 트랜잭션 정책, 접근제어 정책, 가용자원을 포함하는 Bean이 런타임에 수행하는 동작을 정의한다[1]. 본 논문에서 제안하는 방법은 이러한 Deployment Descriptor가 Bean에 대한 모든 정보를 담고 있다는 사실을 기반으로 한다. Deployment Descriptor는 XML로 기술되어 있고, 어플리케이션 내의 모든 Bean에 대한 정보를 담고 있다. 이를 XML Parser를 이용해 Bean의 정보와 테스트 대상이 되는 메소드의 정보를 추출해낸다. 추출해낸 정보를 기반으로 테스트케이스 생성기를 통해 테스트케이스를 생성한다. 테스트케이스 생성기에서 테스트케이스의 생성과정은 다음과 같다.

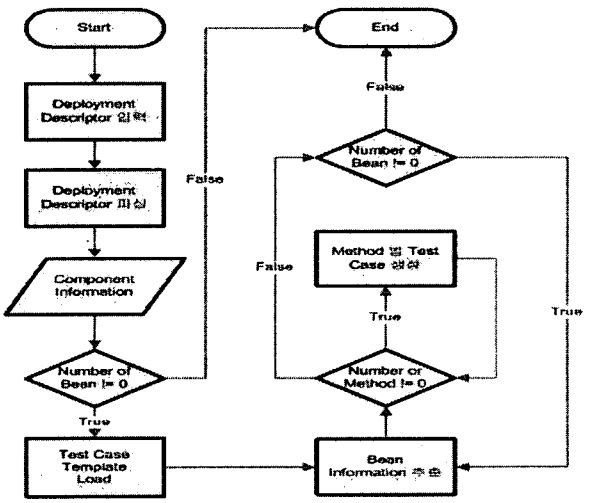

그림 1 테스트케이스 생성 흐름

테스트케이스 생성 시 입력물은 Deployment Descriptor이다. 이 Deployment Descriptor에는 컴포넌트에 들어있는 Bean의 정보를 가진 Component Information을 추출한다. 추출된 정보에서 Bean의 개수를 알아내고, 각 Bean에 대한 테스트케이스를 생성하게 된다. Bean의 테스트케이스를 생성하기 위해서 미리 정의된 테스트케이스 템플릿을 사용한다. 이 템플릿은 테스트 전략에 따라 결정된다. Testing in Isolation 전략을 사용할 경우 테스트케이스 템플릿은 MockObject 생성과 관련된 코드와 테스트케이스 생성에 관련된 코드의 생성에 관련된 정보를 담고 있을 것이다. Out Side the Container Testing 전략의 경우에는 테스트 클라이언트 생성에 관련된 정보를 담고 있을 것이다. 그러나 3가지 EJB 컴포넌트 테스트 전략 중에 가장 현실적이고 정확한 테스트를 수행할 수 있는 방법은 In-Container Testing이다. 본 논문에서 제안하는 방법도 이 전략을 기반으로 하고 있다. 따라서 테스트케이스 템플릿 또한 컨테이너 내부에서 동작할 수 있는 EJB형태로 제공된다. 그림 2는 테스트케이스 생성에 사용될 템플릿 중에 Bean 클래스 템플릿의 구조를 보여주고, 그림 3에서 실제로 생성된 테스트케이스의 예를 보여준다.

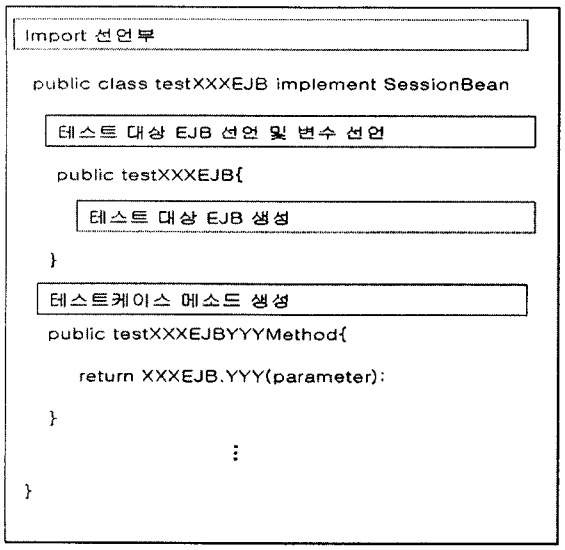

그림 2 테스트케이스 템플릿

테스트케이스 템플릿에서 테스트케이스 메소드 생성 부분은 Deployment Descriptor에서 추출한 Bean의 메소드 정보를 이용한다. 테스트 대상이 되는 EJB의 메소드를 호출하고 그 결과를 전달하는 형태를 가진다. 작성된 테스트케이스의 실행을 위해 EJB 형태의 테스트케이스를 활성화 하고, 호출할 수 있는 테스트 클라이언트를 생성해야한다. 테스트 클라이언트는 테스트케이스를 생성할 때 자동으로 생성된다. 이때 테스트케이스를 참조할 수 있도록 JNDI 컨텍스트 정보를 제공해야하고, 테스트케이스를 호출할 때 입력변수를 전달하고 그 결과를 확인할 수 있도록 작성된다.

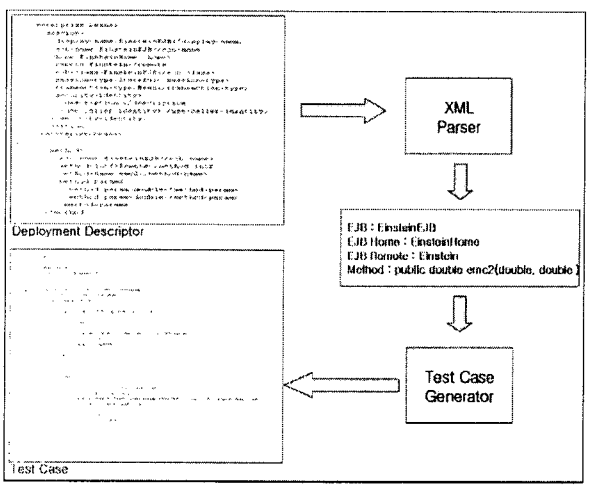

그림 3 테스트케이스 생성 예제

작성된 테스트케이스는 테스트 대상이 되는 EJB와 같은 컨테이너 내부에 배치되어야 한다. 테스트의 실행은 테스트케이스가 컨테이너에 배치된 후에 진행된다. 외부의 클라이언트로부터 테스트케이스를 호출하고 테스트케이스는 테스트 대상이 되는 EJB를 호출하고 그 결과를 테스트 클라이언트에 전달한다. 그림 4는 전체 시스템의 구조이다.

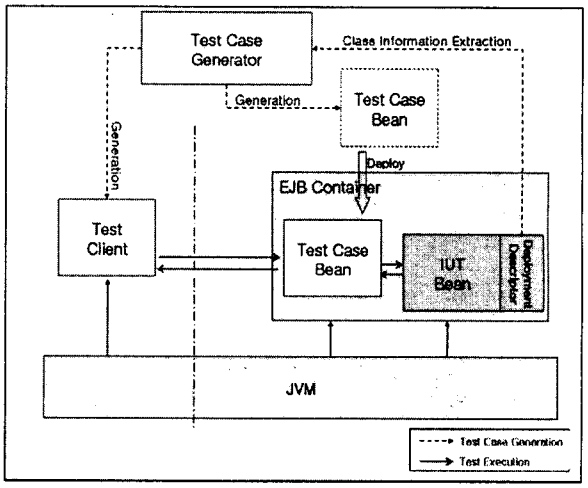

그림 4 시스템 구조

4. 결론 및 향후 연구 과제

본 논문에서는 EJB 컴포넌트 테스트를 위한 테스트케이스를 Deployment Descriptor를 기반으로 자동으로 생성하는 방법에 대하여 제안한다. 이 방법을 통해 EJB 컴포넌트의 테스트케이스를 따로 구현해야하는 문제점을 해결하고, 개발자 혹은 컴포넌트 사용자의 테스트에 관한 수고를 덜어 줄 수 있을 것이다. 향후에 본 논문에서 제안하는 방법을 실제로 구현하고, 생성되는 테스트케이스를 컨테이너에 자동으로 배치하는 방법에 대한 연구를 수행함으로써 최소한의 노력으로 테스트를 수행할 수 있는 방법을 제시하고자 한다.

5. 참고 문헌

[1] Rod Johnson, "Expert One-on-One J2EE Design and Development", Wrox, 2003

[2] Scott W. Ambler, "A J2EE Testing Primer", http://microsites.cmp.com , May 2001

[3]오창남 이긍해, "EJB 컴포넌트 성능 측정 방법", 한국정보과학회 논문지 2000

[4] 송호진 최은만, "컴포넌트 테스트를 위한 래퍼의 자동생성에 관한 연구", 정보과학회 논문지 Vol32, No.8, Aug 2005

[5] Scott Stirling, "Testing J2EE applications", Java World , August 2004

[6] JiRong Hu, "Automating EJB Unit Testing", http://onjava.com , May 2003

[7] Michael T. Nygard and Tracie Karsjens, "Test infect your Enterprise JavaBeans", Java World, May 2000