웹 상에서 테스트 가능한 소프트웨어 컴포넌트 뱅크 서비스 모델

(주)비트컴퓨터 우형철 · 손무훈

1. 서 론

소프트웨어 컴포넌트 산업의 육성이라는 국가적 과제 성공을 위해서, 다양한 배경과 환경 속에 있는 도메인 전문 소프트웨어 컴포넌트 개발자들이 우후죽순격으로 생겨나야 한다. 또한, 이들을 통해서 모든 분야의 소프트웨어 컴포넌트들이 개발되어져야 하며, 개발된 컴포넌트들이 활발하게 유통될 수 있는 기본환경이 조성되어야 한다. 이를 성취하기 위해서 웹 상의 분산 컴포넌트 저장소, 컴포넌트 등록, 검색, 및 컴포넌트를 구독하고 청구하는 인터넷 기반의 컴포넌트 유통시스템에 관한 연구가 진행되고 있다.

본 고에서는 이와 같이 웹을 기반으로 한 소프트웨어 컴포넌트 등록, 저장, 검색, 및 유통을 가능하게 하는 시스템을 만들 때, 컴포넌트 개발자가 등록한 자신의 컴포넌트를 컴포넌트 뱅크에서 제공하는 서비스를 통하여 즉시 웹 상에서 테스트 할 수 있고, 컴포넌트 수요자가 구입을 결정하기 전에 웹 상에서 컴포넌트의 기능을 미리 테스트해 볼 수 있는 시스템 구축에 있어서 발생하는 문제점을 진단해 보고, 해결책을 모색하며, 이를 가능케 하는 모델을 제시하고, 제시된 모델을 기반으로 구현된 프로토타입 프로젝트의 결과 화면을 소개하도록 한다. 이때 등록될 컴포넌트는 Enterprise JavaBeans 스펙 1.1 기반으로 한 EJB 컴포넌트로 한정하겠다.

2. 웹 상에서 테스트 가능한 소프트웨어 컴포넌트 뱅크 구축 시 문제점과 그 해결 가능성

먼저 웹 상에서 테스트 가능한 소프트웨어 컴포넌트 뱅크를 구축할 때 문제가 될 수 있는 부분을 살펴보면 클라이언트 쪽으로 보여질 테스트 GUI에 관한 것과 데이터베이스와의 연결에 관한 것이다. 이러한 문제와 그 해결방향을 살펴보도록 하자.

2.1 테스트 GUI 문제와 그 해결책

컴포넌트 뱅크가 테스트 가능한 서비스를 제공하기 위해서 해결해야 하는 첫번째 문제는 웹으로 접근하는 클라이언트 쪽으로 보여줄 GUI라고 할 수 있을 것이다. 여기서 언급하고 있는 소프트웨어 컴포넌트들은 서버측에서 눈에 보이지 않는 비즈니스 로직만을 가지고 있기에 테스트 서비스를 제공하기 위해서는 웹 클라언트 쪽으로 보여주는 역할을 하는 GUI 부분들이 - HTML, Servlet, JSP(Java Server Page) - 함께 개발되어야 하며, 구동 시에 함께 연동이 되어야 한다는 것이다.

하지만 이러한 GUI 부분을 만드는 부가적인 작업은 컴포넌트 개발자들이 개발 당시에 자신이 개발하고 있는 컴포넌트를 테스트해 보기 위해서 함께 제작을 하고 있기에 이미 존재하고 있다고 가정을 해도 큰 무리는 없을 것이다. 문제는 이들 GUI 부분에서 EJB를 호출할 때 특정한 환경에 고정된 코딩이 들어가서는 안된다는 것이다. 개발자가 개발 당시의 특정 환경과 이들이 컴포넌트 뱅크에 업로드 되어 디플로이 될 환경과는 다를 수 밖에 없기 때문이다.

그러나 다행히도 이를 인식한 SUN Microsystems에서는 J2EE(Java 2 Enterprise Edit-

ion)를 발표할 때, 그림 1에 나타난 바와 같이 EJB 컴포넌트, 웹 컴포넌트, Application Client 컴포넌트들을 J2EE Application 안에 하나로 묶어서 디플로이 하도록 하였다. 여기서 클라이언트 측에 보여질 GUI를 포함하고 있는 부분이 바로 웹 컴포넌트에 해당되는 것이다. 이 경우 웹 컴포넌트 안에 포함될 Servlet과 JSP에서 Java JNDI(Java Naming and Directory Service) 서비스를 이용하여 EJB의 이름을 논리적인 이름으로 맵핑하여 사용하기 때문에 특정 환경에 고정된 부분이 GUI 부분에 들어가지 않게 되어 컴포넌트 뱅크에 디플로이 될 때에 문제가 생기지 않게 된다. 또한 SUN이 제공하고 있는 J2EE 서버는, 자바언어를 사용하여 어플리케이션을 개발하는 개발자들이 JDK(Java Development Kit)를 기본으로 사용하듯이, EJB를 개발하는 개발자들이 비싼 CTM(Component Transaction Monitor)이 없더라도 기본적으로 사용할 수 있으므로 보편화 될 것으로 예상된다.

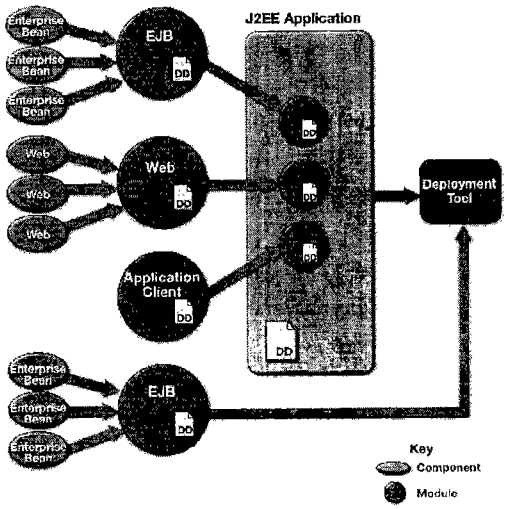


그림 1 J2EE Application 디플로이 개념도

2.2 데이터베이스 연결 문제와 해결책

둘째 문제로는 데이터베이스와의 연동 문제이다. 개발자가 등록할 컴포넌트 대부분은 데이터베이스와 연동을 하게 될 것이다. 이 경우에 데이터베이스 환경과 관련된 특정 부분이 컴포넌트에 포함되어 있을 경우에는 디플로이 될 때에 문제가 생기게 된다. 이 문제가 실제로 업로드와 동시에 자동적으로 컴포넌트를 컴포넌트 뱅크에 디플로이 하여 웹 상에서 테스트 환경을 제공하는데 있어서 꼭 해결해야 할 가장 큰 문제점이라 할 수 있다.

이러한 문제가 발생할 수 있는 부분들로는, GUI 부분에서는 Servlet과 JSP이며, EJB 부분에서는 세션빈에서 데이터베이스를 직접 접근 할 경우와 데이터베이스 테이블 레코드를 대표하는 빈 관리 엔터티빈과 컨테이너 관리 엔터티빈 모두가 이에 해당한다. 이들에게서 발생할 수 있는 문제는 크게 두 가지로 구별해 볼 수 있다. 첫째로, 이들 코드에 해당하는 데이터베이스 테이블이 컴포넌트 뱅크 쪽 데이터베이스에 없을 경우이며, 둘째로, 동일한 테이블이 존재하더라도 이들 코드에서 개발자 데이터베이스에 한정적인 데이터베이스 이름을 사용하여서 데이터베이스 커넥션을 얻고 접근하였을 경우이다.

두번째 문제 경우에는, 앞 절에서 Java JNDI 서비스를 사용하여 EJB의 이름을 논리적인 이름으로 맵핑하여 해결한 것과 같은 방법으로 논리적인 이름으로 데이터베이스를 접근하면 해결 할 수 있다. 즉, 대부분의 EJB 서버에서는 서버를 기동할 때 데이터베이스 커넥션 풀(Database Connection Pool)을 형성하고 이를 Java JNDI 서비를 사용하여 논리적인 이름으로 맵핑한다. 그리고 데이터베이스를 접근할 때에 이 논리적인 이름을 사용하여 접근하게 한다.

문제는 첫번째 경우라고 할 수 있다. 어떻게 개발자가 개발 당시에 사용한 데이터베이스 테이블과 동일한 테이블을 컴포넌트 뱅크에서 제공하면서 등록된 컴포넌트가 즉시 디플로이가 가능하여 테스트 가능한 환경을 제공할 수 있을까? 다음 절에서 이를 해결할 수 있는 세 가지 모델을 제시하고자 한다.

3. 웹 기반 소프트웨어 컴포넌트 뱅크 서비스 모델 제시

소프트웨어 컴포넌트 개발자는 다양한 환경에 처해 있을 수 있다. 이들을 CTM(Component Transaction Monitor)과 데이터베이스 유무에 따라서 분류해 보면 다음과 같은 세 그룹으로 구

분해 볼 수 있다.

첫째는, 상업용 CTM과 데이터베이스를 소유하고 있지 않은 개발자 그룹이다. 그러나 이들도 최소한 SUN에서 제공하는 J2EE를 개발 시에 사용할 수 있다. 이 그룹에 속하는 개발자들은 개발한 소프트웨어 컴포넌트를 전적으로 컴포넌트 뱅크가 관리해 줄 것을 희망할 것이다. 즉, 컴포넌트 뱅크의 데이터베이스를 사용하고 개발한 컴포넌트를 컴포넌트 뱅크의 CTM에 디플로이하기를 원할 것이다. 이들을 위해서는 데이터베이스를 개방해 놓아야 할 필요성이 있다. 이 그룹을 서비스 하기 위한 모델을 Full Management Model이라고 명한다.

둘째로, 상업용 CTM과 데이터베이스를 자체적으로 소유하고 있는 개발자 그룹이다. 이 그룹에 속하는 개발자들은 자신들이 개발한 컴포넌트를 자신의 CTM에 디플로이하고 자신의 데이터베이스를 이용하기를 원할 수도 있다. 이들을 위해서는 이들의 컴포넌트에 관한 정보들만 등록받아서 관리하고, 이를 컴포넌트 사용자들에게 제공, 연결해 주는 역할을 하면 된다. 이 그룹을 서비스하기 위한 모델을 Search & Link Model이라고 명한다.

셋째로, 상업용 CTM은 없지만 상업용 데이터베이스를 소유하고 있는 개발자 그룹이다. 이들도 역시 상업용 CTM은 없지만 J2EE 기반에서 컴포넌트를 개발할 수 있다. 이 그룹에 속하는 개발자들은 개발한 컴포넌트를 컴포넌트 뱅크 상의 CTM에 디플로이 하고 데이터베이스는 자신들의 것을 사용하기를 원할 수도 있다. 이들을 위해서는 컴포넌트 뱅크에서 이들의 데이터베이스와의 연결될 수 있도록 데이터베이스 연결 풀을 생성해 주어야 할 것이다. 이 그룹을 서비스하기 위한 모델을 Database Connection Pool Model이라고 명한다.

그림 2는 개발자가 개발한 컴포넌트를 컴포넌트 뱅크에 등록할 때의 모델별 시나리오를 모두 포함하고 있는 활동 다이아그램이다. 그림 3은

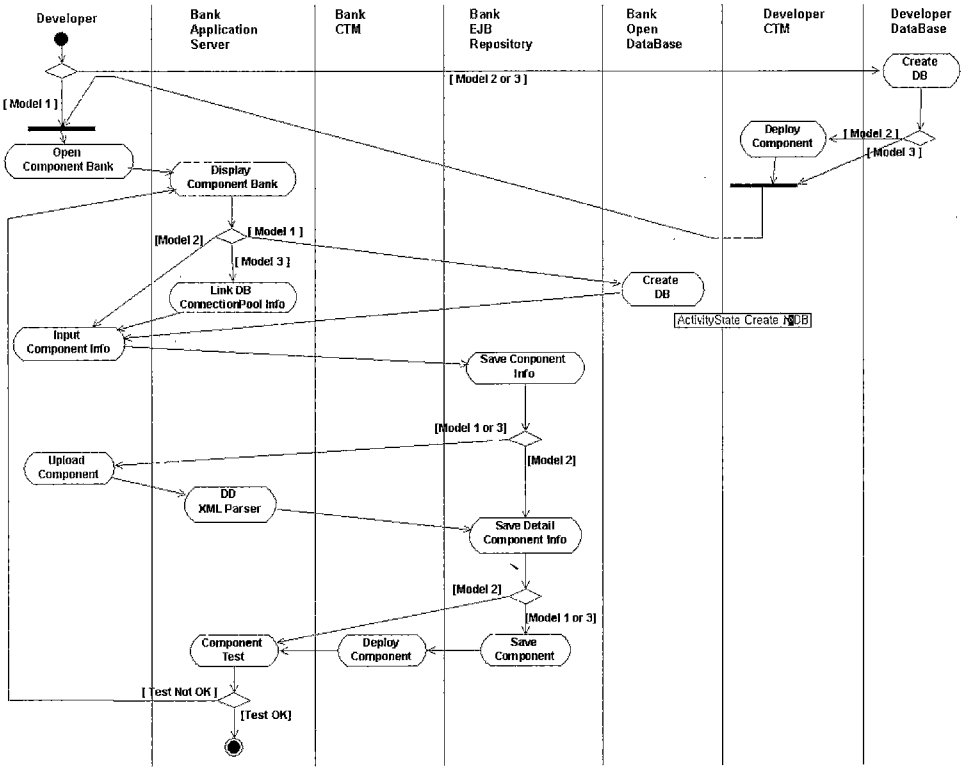


그림 2 developer 활동 다이아그램

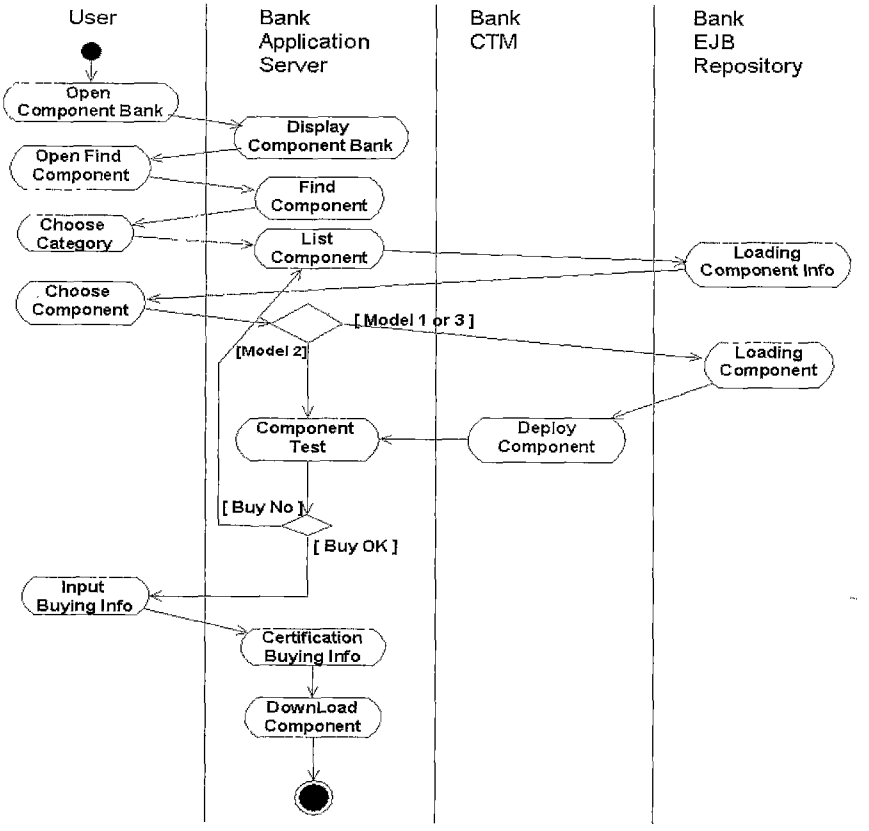


그림 3 User 활동 다이아그램

컴포넌트 사용자가 컴포넌트 뱅크에서 컴포넌트를 검색하고 테스트할 때의 모델별 시나리오를 모두 포함하고 있는 활동 다이아그램이다.

3.1 모델 1 : Full Management Model

3.1.1 컴포넌트 뱅크

1) CTM, 컴포넌트 저장소, 및 개발자를 위한 개방 데이터베이스를 모두 관리한다.

3.1.2 컴포넌트 개발자

1) 컴포넌트를 작성한다.
2) 컴포넌트 뱅크 사이트에 웹으로 접속한다.
3) 개발자로서 자신을 등록한다.
4) 개방 데이터베이스에 필요한 테이블을 생성한다.
5) 개발한 컴포넌트를 등록 및 업로드 한다.
6) 디플로이가 끝난 후에 자신이 업로드한 컴포넌트가 정상적으로 작동하는지 테스트 한다.

3.1.3 컴포넌트 사용자

1) 컴포넌트 뱅크 사이트에 웹으로 접속한다.
2) 사용자로 자신을 등록한다.
3) 검색엔진에 찾고자 하는 컴포넌트에 대한 검색어를 입력한다.
4) 관심 있는 컴포넌트를 웹 상에서 테스트해 본다.
5) 테스트를 마친 후 필요한 컴포넌트를 컴포넌트 뱅크에서 구입한다.
6) 다운로드한 컴포넌트에 필요한 다른 컴포넌트를 추가하여 구하고자 하는 솔루션을 완성한다.

Full Management Model의 장점은 첫째로, 컴포넌트 뱅크가 모든 자원을 직접 관리하기에 신뢰성 있는 서비스를 컴포넌트 사용자에게 지속적으로 제공할 수 있다는 것과, 둘째로, 컴포넌트 개발자들은 값 비싼 CTM이나 데이터베이스를

직접 관리할 필요가 없기에 폭 넓은 개발자들을 수용할 수 있으며, 셋째로, 컴포넌트 개발, 등록, 관리, 유통에 필요한 통일된 규격을 쉽게 이룰 수 있을 뿐 아니라, 마지막으로, 컴포넌트에 대한 통일된 가격책정을 가능하게 한다는 것이다.

3.2 모델 2 : Search and Link Model

3.2.1 컴포넌트 뱅크

1) CTM과, 컴포넌트 뱅크 시스템에 필요한 자원만을 관리한다.

3.2.2 컴포넌트 개발자

1) CTM, 개발자 자신의 데이터베이스를 모두 개발자가 관리한다.
2) 개발한 EJB를 포함한 컴포넌트를 작성하여 자신의 환경에 디플로이 한다.
3) 컴포넌트 뱅크 사이트에 웹으로 접속한다.
4) 개발자로서 자신을 등록한다.
5) 자신의 환경에 디플로이 해 놓은 컴포넌트의 연결정보를 등록한다.
6) 연결정보가 바르게 등록되었는지 테스트한다.

3.2.3 컴포넌트 사용자

1) 컴포넌트 뱅크 사이트에 웹으로 접속한다.
2) 사용자로 자신을 등록한다.
3) 검색엔진에 찾고자 하는 컴포넌트에 대한 검색어를 입력한다.
4) 관심 있는 컴포넌트를 웹 상에서 테스트해 본다. 이때 테스트는 개발자의 사이트에서 이루어진다.
5) 테스트를 마친 후 필요한 컴포넌트를 개발자의 사이트에서 구입한다.
6) 다운로드한 컴포넌트에 필요한 다른 컴포넌트를 추가하여 구하고자 하는 솔루션을 완성한다.

Search & Link Model의 장점은 첫째로, 컴포넌트 뱅크가 모든 자원을 직접 관리할 필요가 없기에 부하가 현저하게 감소한다는 것이며, 둘째로, 개발자가 스스로 자신의 컴포넌트를 관리하기에 신속한 업그레이드가 가능하며, 셋째로 소비자가 애프터서비스를 요구할 때 신속하게 대처할 수 있다는 것이다.

3.3 모델 3 : Developer Connection Pool Model

3.3.1 컴포넌트 뱅크

1) CTM과, 컴포넌트 뱅크 시스템에 필요한 자원만을 관리한다.
2) CTM이 기동 될 때 컴포넌트 개발자의 데이터베이스와 연결될 커넥션 풀이 미리 생성되어진다.

3.3.2 컴포넌트 개발자

1) 자신의 데이터베이스만을 관리한다.
2) 컴포넌트를 작성한다.
3) 컴포넌트 뱅크 사이트에 웹으로 접속한다.
4) 개발한 컴포넌트를 등록 및 업로드 한다.
5) 디플로이가 끝난 후에 자신이 업로드한 컴포넌트가 정상적으로 작동하는지 테스트 한다.

3.3.3 컴포넌트 사용자

1) 컴포넌트 뱅크 사이트에 웹으로 접속한다.
2) 사용자로 자신을 등록한다.
3) 검색엔진에 찾고자 하는 컴포넌트에 대한 검색어를 입력한다.
4) 관심 있는 컴포넌트를 웹 상에서 테스트해 본다.
5) 테스트를 마친 후 필요한 컴포넌트를 컴포넌트 뱅크에서 구입한다.
6) 다운로드한 컴포넌트에 필요한 다른 컴포넌트를 추가하여 구하고자 하는 솔루션을 완성한다.

Developer Connection Pool Model은 이미 어느 정도 신뢰성과 업적이 있는 개발자를 위한 데이터베이스 커넥션 풀을 미리 설정해 주는 것이 핵심이라고 할 수 있다. 이 모델의 장점은 첫째로, 컴포넌트 뱅크가 개발자가 필요로 하는 데이터베이스를 직접 관리할 필요가 없기에 그만큼 부하가 감소한다는 것이며, 둘째로, 개발자 입장에서는 값 비싼 상업용 CTM를 가지고 있을 필요가 없기에 그만큼 부담을 덜 수 있다는 것이다. 모델 1과 모델 2의 절충 형태라고 할 수 있겠다.

4. 프로토타입 컴포넌트 뱅크 시스템 구현

4.1 유즈케이스 다이아그램

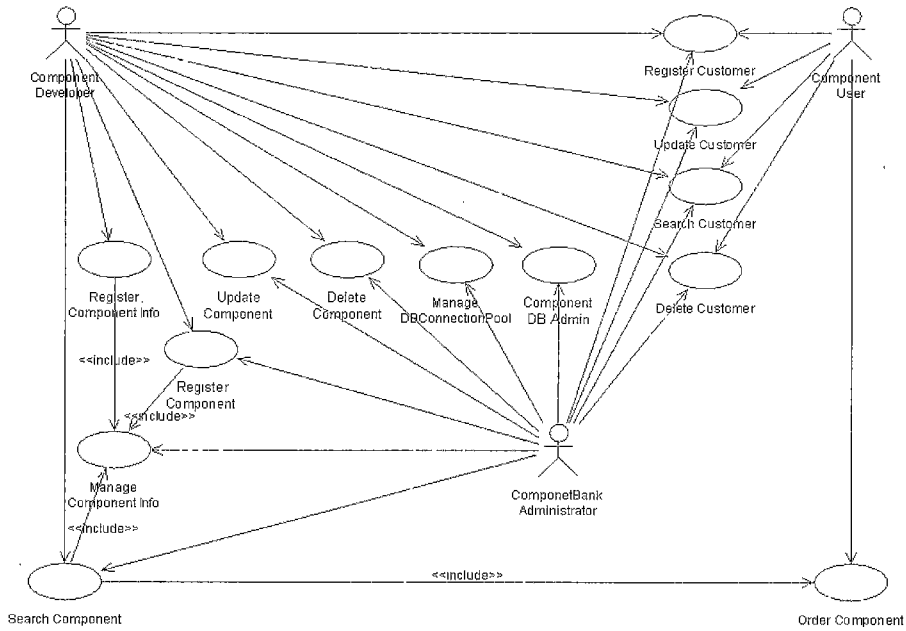


그림 4 유즈케이스 다이아그램

4.2 시스템 아키텍쳐

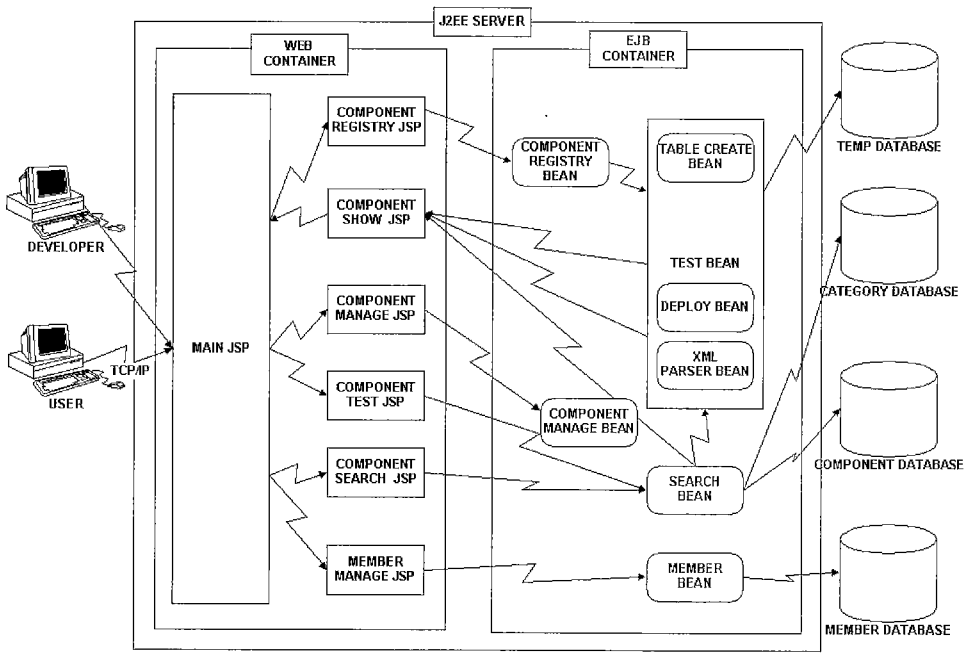
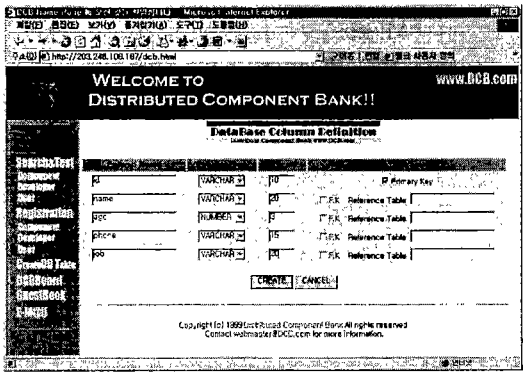


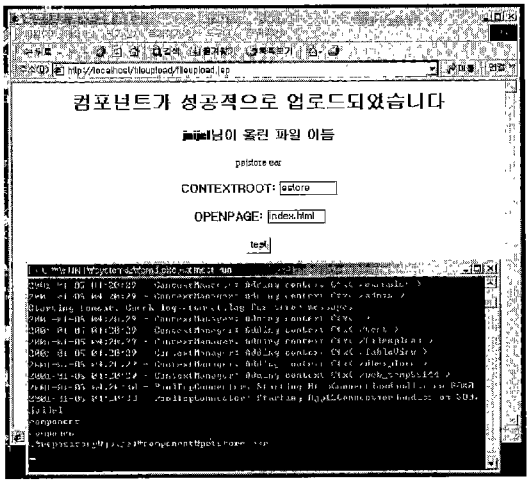
그림 5 프로토타입 컴포넌트 뱅크 시스템 아키텍쳐

4.3 데모 화면

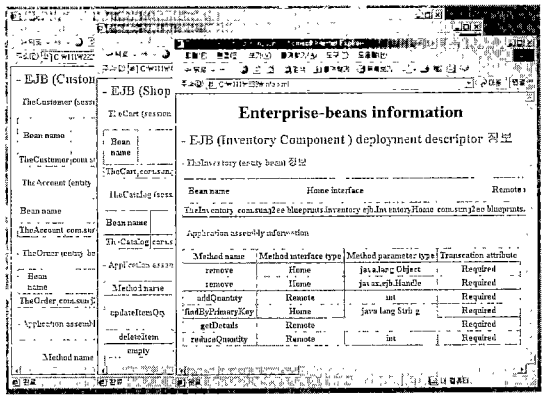


화면 1 개발자를 위한 Open DataBase 테이블 생성화면

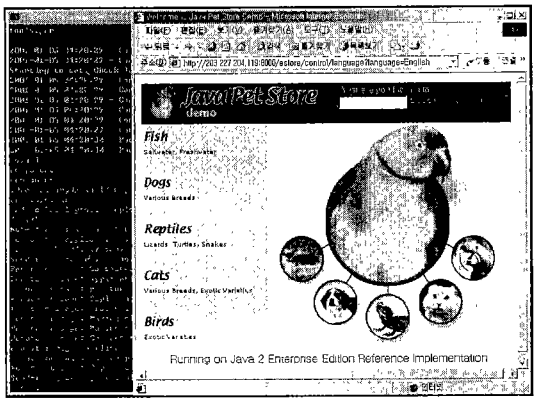
화면 2 컴포넌트 등록 화면



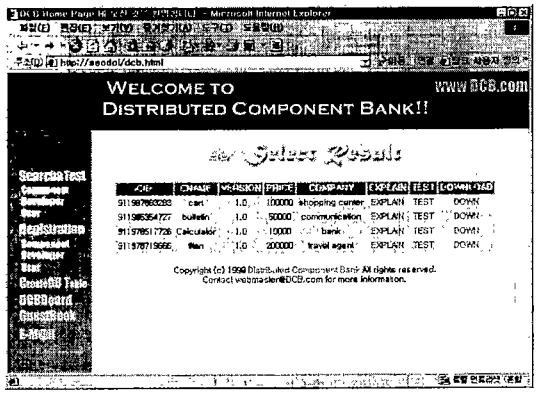
화면 3 컴포넌트 등록의 완료된 화면



화면 4 등록된 컴포넌트의 XML 파싱 결과물



화면 5 등록된 컴포넌트의 테스트 화면



화면 6 등록된 컴포넌트 검색결과 화면

5. 결 론

국내외적으로 거대한 시장을 형성하며 눈 앞에 다가온 소프트웨어 컴포넌트 산업을 육성하는 데

있어서 소프웨어 컴포넌트 뱅크 시스템 구축은 필수적인 기반환경에 해당한다고 할 수 있다. 이에 본 고에서는 웹 상에서 개발자들이 자신이 개발한 컴포넌트를 등록하고 나서 즉시 검색, 테스트하여 볼 수 있을 뿐 아니라, 컴포넌트 수요자가 웹 상으로 접근하여 원하는 컴포넌트를 검색하고, 구입을 결정하기 전에 테스트해 볼 수 있는 서비스를 제공하는 소프트웨어 컴포넌트 뱅크 시스템을 구축하는 데 있어서 예측되는 문제들과 그 해결방안, 그리고 다양한 환경에 있는 컴포넌트 개발자를 위한 세 가지 서비스 모델을 제시하였다. 또한 제시한 모델을 기반으로 구현한 프로토타입의 소프트웨어 컴포넌트 뱅크 시스템을 소개하였다.

제안한 모델을 기반으로 한 시스템은 다양한 환경에 있는 개발자들의 소프트웨어 컴포넌트를 수용하여 관리, 유통함을 통해서 개발자들에게 신속하고 적절한 이익을 가져다 줄 수 있기에 컴포넌트 개발에 대한 의욕을 고무할 것이며, 이에 따라, 산업 전 분야의 전문 지식을 가지고 있는 개발자들을 불러 일으킬 수 있을 것이다. 이에 더불어, 컴포넌트 뱅크 시스템을 중심으로 컴포넌트 개발자들의 풀(Pool)이 쉽게 형성될 수 있도록 해 줄 것이다. 또한 SI 업체들과 컴포넌트 개발자 사이에 원활하고 신속한 상호협력체제를 구축하는데 도움을 줄 것이며, 구입한 컴포넌트를 기반으로 한 신속한 엔터프라이즈 솔루션 구축과 시스템 업그레이드는 기업으로 하여금 급변하는 환경에 신속하게 대처하게 하여 기업 경쟁력을 높여 줄 것이다.

끝으로, 본 고에서 제안한 시스템이 현실화 되기 위해서는 다량의 컴포넌트들이 동시에 테스트 서비스를 하기 위한 모델 연구, 컴포넌트의 가격 결정 조건, 라이센스 등과 같은 컴포넌트 구매에 따른 비즈니스 과정에 있어서 중요한 요소들에 대한 연구 등이 이루어져야 할 것이다.

참고문헌

[1] Doo-Kwon Baik, MetaData Registry (MDR) for Exchange of Software Components, 제1회 한일 공동 컴포넌트 컴퍼런스, 2000.09.

[2] Sun Microsystems, Enterprise JavaBeans Specification , Version 2.0, Public Draft 2, 2000.09.

[3] Sun Microsystems, The Java 2 Enterprise Edition Developers Guide , Version 1.2.1, 2000.05.

[4] Richard Monson-Haefel, Enterprise Java Beans, 2nd Edition , ISBN 89-7914-084-3, OREILLY, 2000.3.

[5] 우형철 외 3인, 분산 컴포넌트 뱅크 , 비트 프로젝트 50호, 비아이티 출판사, 2000.2.

우 형 철

1985 한양대학교 전자공학과 학사
1988 한양대학원 전자공학과 석사
1991~1999 태국 Assumption대학 CEM, CIS, CS 석사과정 수료
1989 금성사 C&C 연구개발본
1991~1999 태국 Assumption대학 공과대학 Full-Time Lecturer
2000 현재 ㈜비트컴퓨터 기술연구소 차장, 컴포넌트기반 프로젝트 부책임자
관심분야: 객체지향, S/W 컴포넌트, 프레임워크, CBD, 컴포넌트 저장소
E-mail:isaacwoo@bit.co.kr

손 무 훈

1999 경남대학교 전자계산학과 학사
2000~현재 ㈜ 비트컴퓨터 기술연구소, 컴포넌트기반 프로젝트팀 개발자
관심분야: UML, OOAD, CBD, Java Server Programming
E-mail:hoonyy@bit.co.kr