JDBC 응용 개발을 위한 RAD TOOL 개발
(Development of RAD Tool for JDBC Application Development)

손 승 우 † 김 순 용 †† 김 창 갑 † 이 상 덕 ††
(Seung Woo Son) (Soon-Yong Kim) (Chang Kap Kim) (Sang Duck Lee)

요 약 클라이언트/서버 응용은 크게 데이타베이스 서버와 응용 및 프리젠테이션 로직을 포함한 클라이언트로 구성된다. 이러한 클라이언트/서버 응용은 최근의 웹의 성장과 함께 자바 언어를 많이 이용하는데 특히 클라이언트 쪽의 GUI 구현과 JDBC를 이용한 클라이언트/서버 응용에 많이 쓰인다. JDBC를 이용하여 클라이언트/서버 응용을 개발할 경우 먼저 응용의 바탕이 되는 데이타베이스의 설계, JDBC 접속을 위한 코딩, 데이타베이스로부터 선택된 데이타들을 보여주기 위한 리포트 양식 등의 설계가 필요하다. 본 논문에서는 이러한 JDBC를 이용한 클라이언트/서버 응용 개발에 필요한 컴포넌트들을 자바빈즈로 설계 및 구현하고 이를 이용한 개발 환경을 제시한다. 구현된 환경을 이용하면 클라이언트/서버 응용의 개발에 있어서 소스 코드 편집을 최소화하고 자바빈즈 컴포넌트들의 선택 및 이들의 속성 편집만으로 쉽게 구현할 수 있다.

Abstract The currently and most widely used client/server architecture is composed of database servers and clients that include both application and presentation logic. To date, due to the rapid growth in WWW, these client/server applications are implemented using Java language, especially in GUI design at client sides and client/server applications development using JDBC. To develop the applications using JDBC, developers should design database schema, code source code for JDBC connection, design report forms to display selected columns from server-side database. In this paper, we propose a development environment based on JavaBeans component for client/server applications development using JDBC. The proposed environment minimizes manual coding and enables developers to develop client/server applications easily with the designed JavaBeans components and their custom property editor.

1. 서 론

클라이언트/서버(Client/Server) 응용의 개발은 크게 윈도우 기반 GUI 환경의 프레젠테이션 로직과 입출력 데이타에 대한 연산을 포함하는 응용 논리로 구성된 클라이언트 프로그램의 개발과 서버의 데이타를 관리하는 서버 프로그램의 개발로 이루어진다 [1][2][3][4][5]. 이러한 클라이언트/서버 모델을 기반으로 응용들은 최근 웹과 많이 연동되면서 응용 또한 웹 또는 인트라넷/인터넷에 적합한 자바 언어를 이용하여 구현되고 있다 [6][7][8]. 전형적인 클라이언트/서버 환경은 그림 1과 같다.

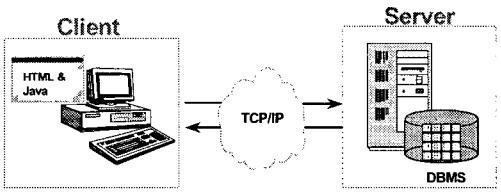


그림 1 전형적인 클라이언트/서버 환경

그림 1을 보면 응용을 위한 다량의 데이타가 서버의 관계형 데이타베이스에 저장되어 있다. 클라이언트 쪽에는 사용자가 서버의 데이타나 서비스를 접근하기 위한

† 정 회 원 : 한국전자통신연구원 S/W공학연구부 연구원
swson@etri.re.kr
changkap@etri.re.kr
†† 비 회 원 : 한국전자통신연구원 S/W공학연구부 연구원
syk@etri.re.kr
lsd@etri.re.kr
논문접수 : 1999년 3월 9일
심사완료 : 2000년 1월 24일

다양한 접근 방법을 제공하는 UI 코드와 응용 로직을 포함한 비즈니스 정책이 구현된 코드가 있다. 이들을 각각 프레젠테이션(또는 UI) 계층, 데이타베이스 계층, 응용 계층(또는 비즈니스 클래스)이라 구분하여 설명한다 [4][5].

최근에는 자바를 이용하여 클라이언트 쪽의 UI를 자바의 AWT(Abstract Window Toolkit)나 스윙(Swing)을 이용하여 자바 애플릿이나 애플리케이션으로 구현한다. 자바는 순수한 객체지향 언어이므로 응용 로직을 UI 클래스와 나누어 설계하고 구현하면 계층 간의 분리 배치도 가능하며 3-계층(tier) 응용으로도 잘 확장된다.

서버의 데이타베이스와 연동하는 클라이언트/서버 응용을 자바 언어로 개발할 경우 개발자는 먼저 응용에 필요한 데이타베이스 설계, 엔티티 관계 또는 객체 모델링, JDBC(Java DabaBase Connectivity)를 이용한 접속 및 SQL 문장 처리, 처리된 결과 값을 보여주기 위한 UI 설계 등의 작업이 필요하다 [5][7]. 자바 API는 클래스 라이브러리의 집합으로 자바에는 프로그램을 작성할 때 사용할 수 있는 방대한 클래스가 패키지로 정의되어 있다. 자바를 이용하여 기존의 클라이언트/서버 응용을 작성하려면 java.sql 패키지의 클래스들이 정의하고 있는 API들을 호출하여 구현해야 하므로 개발자는 이러한 자바 API들을 알고 이에 따라 코딩하여야 한다. 이것은 자바를 이용한 클라이언트/서버 응용의 개발을 불편하게 할 뿐 아니라 응용 개발의 생산성도 떨어지게 된다. 따라서 개발자가 응용의 핵심 비즈니스 문제에 초점을 맞추어서 개발하기 어렵다. 이뿐만 아니라 자바를 요즘의 클라이언트/서버 응용에 적용하려면 기존의 클라이언트/서버 환경에서처럼 시각적인 사용의 용이성과 응용 개발의 생산성을 가지면서 자바 응용의 개방성을 그대로 이용할 수 있는 RAD(Rapid Application Development) 개발 환경이 필요하다 [5].

본 논문에서는 이러한 JDBC를 이용한 클라이언트/서버 응용의 개발을 자바빈즈(JavaBeans) 컴포넌트를 이용하여 쉽게 개발할 수 있는 개발 환경을 제시한다. 제시한 환경은 JDBC를 이용한 데이타베이스의 접속 및 리포트 양식의 설계를 쉽게 하는 자바빈즈 컴포넌트들로 구성하여 쉽게 구현할 수 있다. 또한 이에 해당하는 개발자의 소스 코드 처리를 최소화하고, 이의 수정도 속성 편집기(property editor)를 이용한 마우스 클릭만으로 클라이언트/서버 응용을 쉽게 개발할 수 있다. 본 논문에서 제시한 개발 환경은 컴포넌트들로 설계 및 구현되었기 때문에 컴포넌트기반 소프트웨어 개발의 이점을 그대로 얻을 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 컴포넌트의 기본 개념과 현재 업계 표준 컴포넌트 모델인 DCOM(Distributed Component Object Model)과 자바빈즈와 이들을 이용한 개발 환경에 대해서 살펴본다. 3장에서는 JDBC를 이용한 클라이언트/서버 개발 과정을 살펴보고, 4장에서는 3장에서 JDBC 처리 과정에서의 자바 클래스들을 자바빈즈를 이용하여 JDBC 기반의 클라이언트/서버 응용을 위한 컴포넌트로 설계하고 이를 이용한 개발 환경의 기본 구조를 제시한다. 마지막으로 5장에서 결론과 향후 연구 방향을 기술한다.

2. 관련 연구

본 장에서는 제안한 개발 환경에서 사용한 소프트웨어 컴포넌트의 개념 및 이를 이용한 소프트웨어 개발의 특성에 대하여 살펴본다. 또한 대표적인 컴포넌트 모델인 DCOM과 자바빈즈, 그리고 이들을 이용한 개발 환경을 살펴본다.

소프트웨어 컴포넌트는 소프트웨어 시스템을 구성하는데 쓰이는 재사용 가능한 블록이다. 컴포넌트는 내부의 기술적인 구현을 감추고 이를 이용할 수 있도록 외부로 인터페이스만을 내놓는다. 기존의 라이브러리나 서브루틴 등과 같은 형태의 재사용 가능한 소프트웨어 모듈들은 소스 코드를 수정해야만 하지만, 컴포넌트는 바이너리 실행모듈로서 수정 가능하다는 점에서 이들과 구별된다.

컴포넌트-기반 소프트웨어 개발(Component-Based Software Development: 이하 CBSD)은 기존의 소프트웨어 컴포넌트를 통합하여 대규모 소프트웨어 시스템을 구축하는데 중점을 둔다. CBSD는 시스템의 유연성과 유지보수성을 향상시킴으로써 소프트웨어 개발비용을 줄이고, 시스템을 빠르게 조합하고, 대형 시스템의 지원 및 업그레이드로 수반되는 유지보수 비용을 줄일 수 있다. CBSD의 근본에는 대형 소프트웨어 시스템의 일정 부분은 충분한 규칙성을 가지고 다시 나타나기 때문에 한번만 작성되어야 하며, 공통 시스템은 재작성하는 것보다 재사용을 통해 조립해야 된다는 가정이 있다 [9].

CBSD 성공의 핵심 요인 중 하나는 표준 하부구조인데 이 하부구조 중에서 특히 표준화된 인터페이스가 필요하다. 대표적인 것으로서 DCOM과 자바빈즈가 있으며, 이러한 컴포넌트 표준은 소프트웨어 컴포넌트를 어떻게 만들고 서로 연결하는지 기술하고 있다. 잘 만들어진 컴포넌트 표준은 다음을 보장한다 [9].

- 유사한 사양을 가진 컴포넌트는 상호 교환 가능하고 독립적으로 업그레이드 가능하다.

- 개발자는 컴포넌트의 외부 인터페이스와 행위를 미리 정해진 범위에 따라 사용자 정의할 수 있다.
- 컴포넌트는 그 자체로 하나의 완전한 응용일 뿐만 아니라 보다 큰 컴포넌트로 합쳐질 수 있다.

이와 같이 컴포넌트를 이용한 개발에서는 표준 인터페이스가 중요하며, 정의된 인터페이스에 따라 컴포넌트 표준의 특징이 결정된다. 컴포넌트 표준은 개발자가 재사용 가능한 컴포넌트로부터 예상되는 이점인 향상된 생산성, 일관성, 사용의 편리함, 빠른 시장 출시 등을 얻는데 중요한 역할을 한다.

2.1 컴포넌트 표준

현재 업계 표준 컴포넌트 모델에는 크게 마이크로소프트사의 DCOM과 썬(Sun)사의 자바빈즈가 있다. 현재 소멸된 CI Lab.의 OpenDoc 표준을 대체하는 자바빈즈는 마이크로소프트사의 DCOM과 여러 가지 면에서 비교가 되는 모델이다. 자바빈즈는 자바 1.1부터 소개된 재사용 가능한 소프트웨어 컴포넌트를 정의하는 프레임워크(framework)이다. 컴포넌트 소프트웨어는 원래의 데스크탑에 국한된 복합문서로부터 분산 서버 컴포넌트를 포함하는 엔터프라이즈 응용으로 이동하고 있다. 현재 CORBA의 최종 명세에는 두 컴포넌트 모델이 상호 연동되도록 할 예정이다 [10][11]. 본 절에서는 업계 표준인 DCOM과 자바빈즈의 기본 개념 및 장단점, 그리고 각각의 개발 환경에 대해서 설명한다.

2.1.1 DCOM 및 개발 환경

마이크로소프트사의 컴포넌트 모델은 DCOM에 기반하고 있으며, DCOM은 COM (Component Object Model) 바이너리 표준과 분산된 주소 공간에서의 컴포넌트 통신을 지원하기 위한 런타임(run-time) 하부 구조로 확대된 것이다. 예전의 비주얼 베이직(Visual Basic)으로 구현된 COM기반 컨트롤과 OLE(Object Linking and Embedding) 컨트롤은 각각 VBX와 OCX로 알려져 있다 [11]. DCOM은 인터페이스 컴포넌트가 반드시 구현해야 할 타입과 구조를 기술하고 있다. DCOM 컴포넌트들은 적어도 하나의 인터페이스, 즉 IUnknown을 구현해야 하는데 이것은 인터페이스 참조 캐스팅(casting)과 참조 카운팅(counting)을 위한 기본 메커니즘을 지원한다 [10].

DCOM 그 자체는 단순한 컴포넌트 표준이며, DCOM의 유용성은 복합문서(compound document), 드랙앤드랍(drag and drop), 지속적 스트리밍(persistent streaming) 및 저장과 같이 구체화된 인터페이스에 있다. DCOM의 이벤트 통지(notification) 메커니즘은 IConnectionPoint와 몇 개의 관련 인터페이스를 통하여 구현된다 [10][11].

DCOM 컴포넌트 인터페이스는 OSF(Open Software Foundation)의 IDL(Interface Definition Language)을 사용하여 기술된다. DCOM은 구현 언어와 구현에 독립적이다. 마이크로소프트사의 솔루션이 대부분 윈도우 플랫폼에 종속되기 때문에 최근까지의 DCOM도 윈도우 플랫폼에 제한되어있다 [2][10]. 플랫폼 종속적인 폐쇄형 접근 방식은 성능 측면의 이점이 있는 반면, 응용에 사용될 DBMS를 선택하거나 시스템의 유연성을 유지하고자 하는 사용자에게는 단점이 된다. 윈도우 환경 외에 Unix나 MacOS로의 이식이 DCOM의 플랫폼 종속성 단점을 보완해 줄 수 있다.

마이크로소프트사의 비주얼 베이직 3.0은 간단하지만 생산적인 VBX 컴포넌트를 합성하는 환경을 제공했으며, VBX 컨트롤은 DCOM 표준에 근간을 둔 컴포넌트로 대체되었다. 게다가 마이크로소프트사는 자사의 통합 개발환경에 점진적으로 컴포넌트-지향의 프로그래밍 특성들을 주입해왔다 [10]. 예를 들어, 비주얼베이직 5.0은 기존의 컴포넌트뿐만 아니라 개발자가 새로운 컴포넌트를 만들 수도 있게 해준다. 더욱이 비주얼 베이직 IDE(Integrated Development Environment)는 컴포넌트가 실행 시에 필요한 코드의 상당 부분을 생성해준다. 예를 들어, 비주얼 베이직 5.0은 컴포넌트 인터페이스와 메타데이타를 자동으로 생성하고 등록해준다 [15].

마이크로소프트사의 IDE는 현재 비주얼 베이직, 비주얼 C++, J++ 3가지 언어로써 컴포넌트 개발을 지원한다. IDE 자체는 네이티브 DCOM을 이용하여 만들었으며, 표준 DCOM 메커니즘을 통하여 확장 또는 사용자 정의할 수 있게 한다. 개발자는 이러한 IDE를 이용하여 컴포넌트의 룩앤필(look and feel)을 바꾸거나 원하는 개발 패턴을 가진 컴포넌트로 사용자 정의할 수 있다.

2.1.2 자바빈즈 및 개발 환경

DCOM이 구현 언어에 독립적이지만 플랫폼에 종속적인 반면, 자바빈즈는 플랫폼에 독립적이지만 구현 언어에 종속적이다. 즉, 자바빈즈의 구현 언어는 자바 언어만이 가능하다. DCOM 컴포넌트의 개발은 보다 간단한 바이너리 컴포넌트들을 합성해서 만들어진다. 예를 들어, IPersist 인터페이스는 IStorage와 IStream으로부터 합성되어진다 [10]. 반면 자바빈즈 컴포넌트는 표준 자바 클래스 라이브러리의 언어 확장 집합으로 구현되어진다. 따라서 자바빈즈는 기능별로 나누어진 자바 API의 집합이다. 자바빈즈의 플랫폼 이식성은 자바 가상 머신(Java Virtual Machine)을 통하여 이루어진다 [

2][6][10][12].

DCOM과 마찬가지로 자바빈즈 인터페이스도 속성(property), 메쏘드(method), 이벤트(event)를 외부 인터페이스를 통하여 보여준다. 자바빈즈의 속성 모델은 DCOM보다 풍부하다. 자바빈즈는 단일/다중치(single/multi value) 속성뿐만 아니라 한계(bound) 속성과 제한된(constrained) 속성 타입도 정의하고 있다. 한계 속성은 자바 이벤트를 사용하여 다른 컴포넌트에게 속성 값의 변화를 알려준다. 제한된 속성을 가진 컴포넌트는 속성의 변경을 거부하는 특성이 있으며, 이 특성을 이용하면 언어기반의 일관적인 검증 논리를 제공할 수 있다. 한계 및 제한된 속성은 대부분의 객체기반 시스템에서 없던 것들이다. 이들 속성을 이용하면 개발자는 응용 로직을 모듈별로 분해할 수 있고, 비즈니스 데이타의 일관성을 유지할 수 있으며, 전체 시스템은 유지보수가 쉬워진다 [10][13].

자바빈즈의 이벤트 통지 메커니즘은 3개의 자바 클래스 인터페이스인 이벤트(Event), 이벤트소스(EventSource), 이벤트리스너(EventListener)로 되어있다. 이벤트소스는 등록된 모든 이벤트리스너에게 통보되며, 관심사에 있는 이벤트가 발생하면 각 이벤트 객체를 전달한다. 이 3개의 클래스들은 그림 2와 같은 위임(delegate) 모델로 표현되어진다. 이벤트 모델은 사용의 용이성을 위하여 두 가지 유형을 제공한다. 이벤트소스는 멀티캐스트(Multicast)가 기본이나 유니캐스트(Unicast)로도 설정할 수 있다. 유니캐스트 이벤트는 단 하나의 리스너와 연결될 수 있는 클래스이다. 멀티캐스트 이벤트는 여러 리스너에 의해 모니터될 수 있다. 이벤트어댑터(EventAdapters)를 추가함으로써 개발자가 하나의 리스너 인터페이스에 정의된 모든 이벤트들에 대한 핸들러를 코딩해야 하는 부담을 덜어준다 [10][13].

그림 2 자바의 위임 이벤트 모델

자바빈즈는 컴포넌트 인터페이스 구성 개념뿐만 아니라, DCOM과 유사한 컴포넌트 내포와 메타 데이타를 위한 충분한 메커니즘을 갖고 있다.

자바빈즈 API는 명시적으로 속성 시트(property sheets) - 내장된 속성 편집기와 편집기 집합 - 를 통하여 빈 컴포넌트의 시각적 개발을 지원한다. 이러한 IDE 또는 응용 생성기는 DCOM의 IPropertyPage 인터페이스와 유사하다.

시멘텍사의 비주얼 카페(Visual Cafe), IBM사의 비주얼에이지 for 자바(VisualAge for Java), 인프라이즈사의 J빌더, 썬사의 자바 워크샵(Java WorkShop) 등을 포함하여 다수의 IDE들이 속성 시트, 팔레트, 설계시 드래앤드랍 방식을 가진 시각적 개발을 지원한다. 이러한 도구들은 마이크로소프트사의 IDE와 비교할 수 있을 만큼 생산적이다. 게다가 컴포넌트를 상호 연결하는 이러한 도구들의 시각적 모델은 현재까지의 마이크로소프트사 IDE보다 다양하고 직관적이다. 이와 같이 자바빈즈는 DCOM보다는 성능측면만을 제외하고는 컴포넌트기반의 개발을 용이하도록 하는 특징들을 언어 자체에 내포하고 있다.

지금까지의 DCOM과 자바빈즈의 특징들을 정리하면 표 1과 같다.

표 1 DCOM과 JavaBeans의 특징 비교

특징	DCOM	JavaBeans
소유 및 지원	마이크로소프트사	썬사
표준 상태	폐쇄적	개방적
플랫폼	윈도우즈/매킨토시	플랫폼에 독립
객체지향 상속	× (위임과 집단화를 통한 재사용 제공)	○
인터페이스 정의	IDL	get(), set() 디자인 패턴
지속적 인식	GUID(Globally Unique ID) : 전역적으로 유일	문자열 이름 (전역적으로 유일하지 않을 수 있음)
외부 참조 방식	ImportLib	Package import conventions
컴포넌트 정보	ITypeLib.GetLibAttr() ITypeInfo.GetDLLEntry()	BeanDescriptor, BeanInfo
시각적 개발지원 방법	IPropertyPage 인터페이스	Property Sheets

2.2 기존의 클라이언트/서버 개발환경과 비교 분석

비주얼 베이직은 다른 클라이언트/서버 도구들과 비교될 만큼 UI 설계 기능이 강조된 도구이다. 비주얼 베이직은 ODBC(Open DataBase Connectivity)를 통하여 다수의 데이타베이스로의 접근을 제공하지만, 자사 제품

인 MS 엑세스(Accss)를 사용했을 때 보다 많은 기능을 사용할 수 있으므로 특정 DBMS에 성능이 종속된다. 비주얼 베이직으로 생성된 응용 역시 개발 환경인 윈도우 플랫폼에 종속된다.

파워빌더(PowerBuilder)는 현재 상용의 클라이언트/서버 개발도구 중 가장 강력한 데이타베이스 응용 개발을 가능하게 하는 DataWindow를 제공한다. 파워빌더는 대부분의 주요 데이타베이스를 자체적으로 지원하며 ODBC를 통한 접근도 가능하다. 파워빌더는 작성된 응용에 대한 소스 코드를 생성하지 않으며 응용의 로직 부분을 작성하기 위해서 자체적인 스크립트 언어인 파워스크립트를 제공한다. 이것은 응용의 개발시 또 다른 언어를 습득해야 하는 부담이 있으며 개발후 배치에도 문제가 있다.

비주얼 카페는 작성된 응용에 대한 자바 소스 코드를 생성해 주며, 자바를 이용하여 3-계층 데이타베이스 응용을 개발할 수 있는 도구이다. 비주얼 카페는 3-계층 데이타베이스 응용을 위하여 dbANYWHERE라는 미들웨어 프로그램을 제공하는데 작성된 데이타베이스 응용은 이 미들웨어에 종속된다.

J빌더는 도구 자체의 많은 부분이 자바로 작성된 개발도구로서 시각적으로 자바 응용을 생성해줄 뿐만 아니라 CORBA를 이용한 분산 응용까지 개발할 수 있는 도구이다. J빌더 역시 데이타베이스를 이용한 응용의 생산성을 높이기 위해서 DataGateWay 서버라는 미들웨어를 사용한다. 현재 DataGateWay 서버는 윈도우 플랫폼에만 동작하므로 작성된 응용 역시 윈도우 플랫폼에 종속적이다.

비주얼 카페와 J빌더의 최근 버전들은 자바2로 구현되었는데, 이들을 살펴보면 이전의 폐쇄적 환경(dbANYWHERE, DataGateWay)을 배제하고 100% 순수 자바를 이용한 JDBC 응용 환경을 지원한다. 이는 자바 언어가 최초 1.0 이래 자바2로 계속 발전하면서 흡수된 다양한 클래스 패키지(예를 들어, 자바2의 CORBA 패키지)들을 개발자가 자바 언어를 써서 쉽게 개발할 수 있게 제공하기 위한 것이다. 즉, 자바 응용 개발 도구들의 환경이 점점 더 개방적인 환경으로 변하고 있다.

이러한 RAD 도구를 통하여 얻을 수 있는 장점으로 첫째는 C/C++와 같은 범용의 프로그래밍 언어를 사용하는 것보다 사용하기 쉽다는 것이다. 둘째로는 개발하는데 드는 시간이 범용 언어를 사용했을 때보다 단축된다는 것이다. 셋째로는 RAD 도구의 특성상 설계 단계에 쉽게 포함할 수 있다는 것이다. 단점으로는 C/C++와 같은 3세대 언어와 비교했을 때의 성능 문제와 RAD 도구 자체의 독자 모델 사용으로 인한 도구의 확장성, 폐쇄성 문제 등을 들 수 있다.

이와 같은 기존의 클라이언트/서버 도구들의 공통 특징은 일관성이 부족하다는 것이다. 도구마다 컴포넌트 모델을 채택하여 CBSD의 이점을 얻고는 있지만, 비주얼베이직이나 파워빌더처럼 특정 플랫폼에 제한적인 응용을 생성하는 경우가 많다. 비주얼카페나 J빌더는 플랫폼에 독립적인 자바빈 컴포넌트에 기반을 두지만, 데이타베이스를 이용한 응용에서는 dbANYWHERE와 DataGateWay와 같은 미들웨어를 이용한 경우에만 생산성을 높일 수 있으므로 도구의 개발 환경인 윈도우 플랫폼에 종속된다. 즉, 데이타베이스를 이용한 응용 작성시 도구마다 독자적인 클라이언트/서버 개발 접근 방법과 컴포넌트 모델을 따르므로 이들 모두 폐쇄적 개발 환경이다. 데이타베이스를 이용한 클라이언트/서버 응용의 개발시에는 본 논문에서 제안한 개발환경인 ShakeHands는 자바빈즈 컴포넌트를 이용하여 개발 및 실행 환경에 독립적으로 동작할 수 있는 데이타베이스 응용을 개발할 수 있다는 장점이 있다. 개발자가 작성한 화면에 대하여 자바 언어로 소스 코드를 생성하고 여기에 응용 논리도 자바로 코딩할 수 있으므로 추가의 개발 언어 습득에 어려움이 적다. 자바 언어의 특성상 개발 환경과 생성된 응용의 실행 속도가 문제이다. 그러나 핫스팟(HotSpot) 기술로 인하여 자바가상머신의 속도 향상을 얻을 수 있으며, EJB(Enterprise JavaBeans)와 같은 기술은 현재 클라이언트/서버 환경이 엔터프라이즈 환경으로 이동하는 것에도 쉽게 적용될 수 있다.

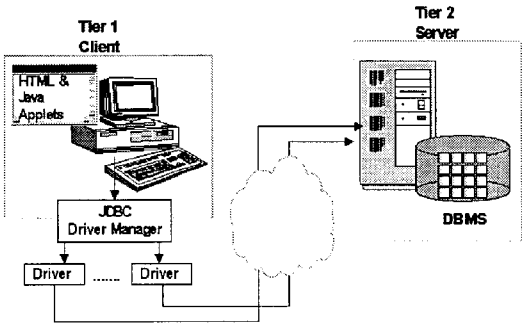


그림 3 JDBC 2-계층 구조

3. JDBC와 클라이언트/서버 응용

본 절에서는 JDBC를 이용하여 클라이언트/서버 응

용 개발 과정을 살펴본다. 그림 1의 클라이언트/서버 구조를 JDBC를 이용하여 구성하면 그림 3과 같다. 그림 3에서 보면 클라이언트 쪽에 JDBC 접속을 위한 드라이버들이 클래스 파일 형태로 존재하며, 클라이언트의 자바 애플릿 또는 애플리케이션은 드라이버 관리자가 관리하는 드라이버 파일들을 통하여 해당 데이타베이스로 접속하게 된다 [2][6][12][14]. 본 논문에서는 이러한 2-계층 클라이언트/서버 구조를 기반으로 하는데, 여기서 서버는 단지 데이타 서버로서 개발자가 직접 코딩해야 할 프로그램은 없다.

그림 4는 일반적인 JDBC 객체와 메쏘드, 그리고 이들을 이용하여 응용을 개발하는 순서를 나타낸 것이다. 자바 프로그램이 데이타베이스의 데이타를 처리하려면 다음과 같은 일련의 과정을 따라야 한다 [14].

- 초기화 및 할당(initialization and allocation) : Connection 객체를 얻음
- 문장 수행(statement handling) : SQL 문장 수행
- 결과 처리(process results) : 반환된 SQL 문장의 결과값 처리

먼저 프로그램에서 getConnection() 메쏘드를 호출하여 Connection 객체를 얻은 다음, Statement 객체를 생성하고 수행할 SQL문을 준비한다. SQL문은 즉시 실행가능(Statement 객체)하거나, 컴파일된 문장(Prepared-Statement 객체)이거나, 저장 프로시저의 호출(Call-ableStatement 객체)이 될 수도 있다. 만약 executeQuery() 메쏘드가 실행되면 ResultSet 객체가 반환된다. 갱신이나 삭제와 같은 SQL문은 ResultSet을 반환하지 않는다. 이러한 문장들에는 executeUpdate() 메쏘드가 쓰인다. executeUpdate() 메쏘드는 SQL문에 의해 영향을 받은 열의 숫자를 나타내는 정수를 반환한다 [14].

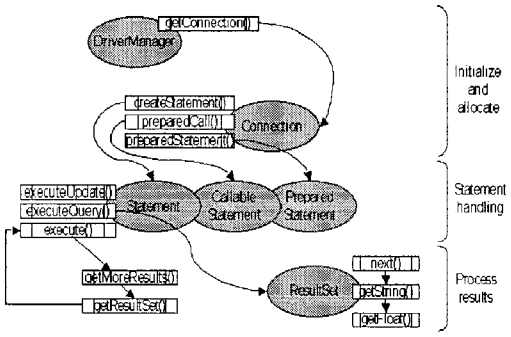


그림 4 JDBC 개발과정

ResultSet은 next() 메쏘드를 이용하여 파싱된 데이타의 열들을 갖고 있다. 트랜잭션 처리 응용의 경우 rollback(), commit()과 같은 메쏘드를 이용한다 [14].

이러한 일련의 개발 과정을 자바 응용 개발자는 적절히 해당 자바 클래스(java.sql 패키지)들의 API를 이용하여 코딩하여야 한다. 예를 들어, 그림 4에서 Driver-Manager 클래스의 getConnection() 메쏘드를 이용하는 경우 실제 개발자는 그림 5와 같은 소스 코드를 편집하게 된다. 그림 5는 JDBC를 통한 데이타베이스 접속을 위한 소스 코드를 보여주고 있다. 그림 5의 밑줄친 부분에서 알 수 있듯이 사용자는 매번 데이타베이스 접속시 JDBC 드라이버 클래스 파일(그림 5에서 oracle.jdbc.driver.OracleDriver)의 설정, 접속을 지정하는 JDBC URL(그림 5에서 jdbc:oracle:thin:@150.183.106.141:1521:ORA), 사용자(그림 5에서 scott) 및 패스워드(그림 5에서 tiger) 등의 정보를 코딩해야 한다. 이러한 과정을 드라이버, URL, 사용자, 패스워드 4개의 속성으로 정의하고 이들의 값을 읽어오고 수정하는 내부 구현을 가진 컴포넌트로 구현할 수 있다.

```
try {
    // Load the Oracle JDBC driver
    Class.forName ("oracle.jdbc.driver.OracleDriver");

    // Connect to the database
    Connection conn = DriverManager.getConnection
     ("jdbc:oracle:thin:@150.183.106.141:1521:ORA", "scott", "tiger");
}
catch (SQLException e) {
    System.out.println ("SQLException");
}
catch (ClassNotFoundException e) {
    System.out.println ("Exception");
}
```

그림 5 JDBC 접속을 위한 소스 코드

그림 4의 개발 과정을 살펴보면 처음 초기화 및 할당 과정에서는 주로 데이타베이스 접속을 위한 환경 설정을 하며, 문장 처리 부분에서는 실제로 개발자가 원하는 데이타를 접속 과정에서 얻은 접속 객체로부터 각종 질의문을 처리하며, 마지막의 결과 처리는 질의문 등을 통하여 얻어진 결과를 어떻게 화면상에 보여줄 것인가를

처리하는 것이다. 따라서 각 과정을 컴포넌트로 모델링하여 응용에서 사용하면 JDBC를 이용한 클라이언트/서버 응용의 개발을 용이하게 할 수 있다.

본 논문에서는 자바빈 컴포넌트를 이용하여 이러한 JDBC 관련 객체들을 자바빈즈 컴포넌트로 모델링하여 구현한다. 제안된 컴포넌트들을 이용하면 개발자는 이러한 개발 과정을 단지 설계시에만 나타나는 시각화 컴포넌트의 드래앤드랍의 과정을 통하여 쉽게 구현할 수 있다.

4. 개발 환경 설계 및 구현

4.1 시스템 구현

컴포넌트의 개념은 컴포넌트 기반 개발 환경이나 빌더와 같은 개념과 밀접한 관계에 있다. CBSD의 가치는 IDE의 효율성과 사용성에 전적으로 의존한다. IDE는 텍스트 기반의 프로그래밍에서 시각적인 컴포넌트 조작 쪽으로 빠르게 변하고 있다. 비주얼 베이직 [15], 파워빌더 [16], 비주얼에이지 for 자바, 비주얼 카페 [17]와 같은 상용의 IDE들은 보통 다음의 기능 요소들을 가지거나 또 가지려고 하고 있으며, 응용의 논리를 기술하기 위해 VBScript나 파워스크립트(PowerScript)와 같은 스크립트 언어를 가지고 있다 [10].

① 사용 가능한 컴포넌트를 보여주는 하나 또는 그 이상의 팔레트(아이콘으로 표현됨)
② 컴포넌트가 위치하고 드래앤드랍이나 팝업 메뉴 등을 통하여 상호 연결될 캔버스 컨테이너(폼 디자이너로 표현됨)
③ 개발자가 컨테이너 내의 컴포넌트를 사용자 정의할 수 있게 하는 속성 및 스크립트 편집기
④ 새로운 컴포넌트를 만들거나 컴포넌트 응용을 시험하고 정제하기 위한 편집기, 브라우저, 인터프리터(또는 컴파일러), 소스 수준 디버거
⑤ 사용자의 검색 원칙에 따라 컴포넌트를 찾고 컴포넌트의 메타 데이타를 조사하기 위한 컴포넌트 저장소 및 연관된 설계시간 탐색 기능

이와 같은 원칙에 따라 본 논문에서는 자바빈즈를 이용하여 응용을 개발할 수 있는 통합 개발 환경인 ShakeHands를 개발하였다. ShakeHands는 크게 메인 화면과 속성 및 이벤트 편집기, 폼 디자이너, 소스 코드 편집기 및 생성기, 응용 관리기 등으로 구성되어 있다. 위에서 열거한 기능 요소 중 ①~③은 UI로 표현되는 부분으로 스윙으로 구현하였다. 컴파일러와 디버거는 JDK에서 제공하는 javac와 jdb를 이용하였다. 컴포넌트 저장소 및 탐색 기능은 자바의 코어 리플렉션(Core Reflection)과 인트로스펙션(Introspection) 기능들을 이용하여 구현하였다. ShakeHands 구성 요소들간의 연동 관계는 그림 6과 같으며 이들간의 상호 연동원리는 자바의 감시자(Observer) 패턴을 이용하여 구현하였다 [18].

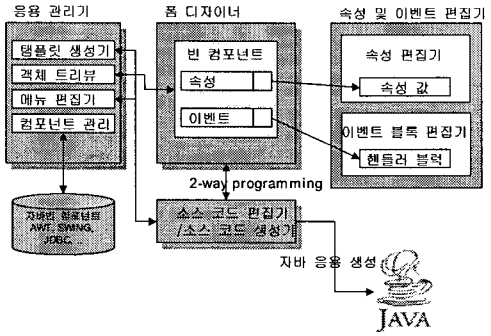


그림 6 ShakeHands의 구성 요소 및 연동관계

4.2 컴포넌트 및 사용자 속성 편집기의 설계 및 구현

본 절에서는 3장에서 기술한 JDBC를 이용한 클라이언트/서버 응용 개발을 위한 컴포넌트를 자바빈으로 설계/구현하고 이를 위한 사용자 속성 편집기에 대해 설명한다.

4.2.1 컴포넌트 설계

그림 3에서 응용 설계를 위해 필요한 JDBC 클래스들을 크게 다음과 같이 3가지로 기능을 분류하여 각각에 해당하는 자바빈즈 컴포넌트를 설계하였다.

- 데이타베이스 접속 컴포넌트 (JdbcConnection)
- 테이블 정의 컴포넌트 (TableDefinition)
- 데이타 컴포넌트 (DBAccess)

접속 컴포넌트는 JDBC를 통하여 데이타베이스에 접근할 때 필요한 정보를 가지고 있는 빈이다. 테이블 정의 컴포넌트는 접속 컴포넌트에 의해 설정된 데이타베이스에 접속하여 개발자가 보여주고자 하는 테이블을 설정할 수 있도록 하는 컴포넌트이다. 접속 및 테이블 정의 컴포넌트는 나중에 실행시에 전혀 화면 UI를 가지지 않으므로, 설계시에만 화면 디자이너 상에 나타나고 실행 시에는 보이지 않는 빈(invisible beans)이다.

데이타베이스와의 접속 설정을 위해 정의된 JdbcConnection 컴포넌트의 속성은 주로 JDBC의 접속을 위한 것들로서 driverName, url, userName, password 등의 속성을 포함해야 한다. 따라서 Jdbc-

Connection 컴포넌트는 이러한 속성들을 get(), set() 메쏘드를 통하여 속성 값을 조회하고 수정할 수 있다.

접속 및 테이블 정의 컴포넌트에 의해 설정된 데이타베이스 정보를 실제로 화면에 보여주는 것은 데이타 컴포넌트 (DBAccess) 이다. 정의한 DBAccess 컴포넌트들로는 TextArea, TextField, Label, List 등으로 모두 JDK(Java Development Kit)의 AWT에 있는 빈과 같은 기능을 하지만, 속성 값으로 데이타베이스와의 바인딩 정보를 갖고 있어서 실제 접속된 데이타베이스의 테이블 값을 보여줄 수 있다. 이외에도 리포트 양식 설계시 많이 사용되는 컴포넌트인 그리드(Grid) 컴포넌트를 설계하였다. 이들과 기존의 자바 클래스들간의 관계는 그림 7의 클래스 다이어그램에 나타나 있다.

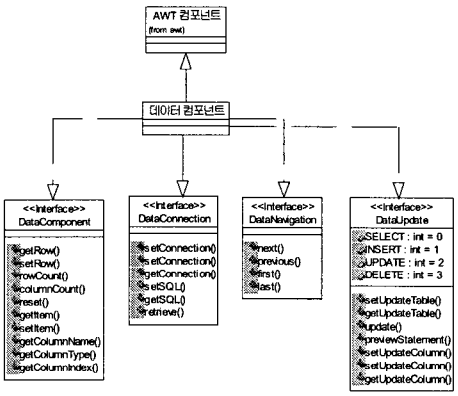


그림 7 데이타 컴포넌트의 클래스 다이어그램

DBAccess 컴포넌트를 개발하기 위해서는 자바에서 제공하는 AWT 컴포넌트를 상속받을 필요가 있다. 또한, 컴포넌트 사이에서 공통적인 기능, 예를 들면 변경된 자료의 수정이나 갱신 등의 기능을 제공해야 한다. 자바는 다중 상속을 지원하지 않기 때문에 컴포넌트 사이에서 원하는 공통적인 함수를 인터페이스를 통하여 구현해야 한다. 그림 7을 보면 DBAccess 컴포넌트, 즉 데이타 컴포넌트들은 각각 DataComponent, DataConnection, DataNavigation, DataUpdate 4개의 자바 인터페이스를 구현하고 있으며, 화면 UI 요소는 원래 GUI를 담당하는 리스트나 텍스트영역과 같은 AWT 컴포넌트를 상속받아 구현하고 있다. 여기서 DataComponent 인터페이스는 데이타 값에 접근하고 DBAccess 컴포넌트를 이루는 칼럼에 대한 정보를 얻기 위한 함수를 제공한다. DataConnection 인터페이스는 JdbcConection 컴포넌트와 DBAccess 컴포넌트에게 연결을 바꾸는 기능을 제공한다. DataConnection 인터페이스의 접속 환경은 JdbcConnection 객체로부터 얻을 수 있으며 SQL 속성 역시 TableDefinition 객체로부터 얻을 수 있다. DataUpdate 인터페이스는 DBAccess 컴포넌트가 사용자에 의해서 변경된 어떤 데이타를 갱신하는 DBAccess 컴포넌트에 기능을 제공한다. 그리고, 현재 데이타베이스에 넘겨진 SQL 문을 보는 함수도 제공한다. DataNavigation 인터페이스는 DBAccess 컴포넌트의 위치를 옮기기 위한 4가지 함수를 제공한다. DBAccess 컴포넌트의 위치가 움직일 때, 새 데이타 값이 컴포넌트에 표시되며, 위치를 움직이면 DBAccess 컴포넌트가 표시하고 있는 현재 레코드를 바꾸게 된다.

4.2.2 사용자 속성 편집기

개발자는 그림 8에 나타난바와 같이 폼 디자이너와 속성 및 이벤트 편집기를 이용하여 속성 값을 설정하고 빈의 이벤트에 대한 새로운 핸들러를 제공함으로써 빈 컴포넌트를 사용자 정의할 수 있다. 빈은 IDE 메뉴나 편집기와 같이 명시적으로 또는 폼 상에 빈을 배치하는 것과 같은 시각적 조작의 효과로써 암시적으로 수정된다. 두 방법 모두 IDE는 빈의 메타데이타를 접근해서 메타데이타를 어떻게 보여줄 것인지 해당 속성이나 드래앤드랍 이벤트를 처리할 편집기를 활성화시키는지 알고 있어야 한다 [10][13].

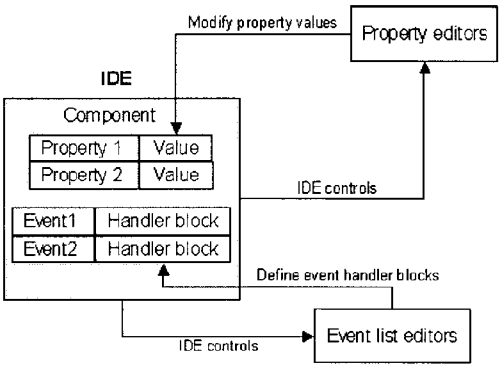


그림 8 ShakeHands에서의 컴포넌트 사용자정의

사용자 정의될 컴포넌트는 자신이 실행시에 실행되는지, 설계시에 실행되는지 알아야 한다. 컴포넌트는 각각의 시간에 서로 다른 인터베이스를 외부에 내놓고 다른 행위를 보여준다. 설계시에 컴포넌트는 자신의 속성과 이벤트 편집기를 외부로 나타내는 메타데이타를 통하여 IDE와 협동한다. 그러나, 실행시에는 설계시에 구체화시

킨 행위를 명시한다. 예를 들어, 설계시에 버튼 컴포넌트를 클릭하면 보통 이벤트 편집기를 보여주게 되고, 실행시에는 버튼의 윈도우 컨테이너에게 자신을 닫으라는 것과 같은 어떤 행위를 수행하게 된다. java.beans 패키지내의 자바빈즈 컴포넌트 API는 이렇게 구성되어 있어서 빈의 설계시 코드와 실행시 코드가 다른 클래스에 요약될 수 있다 [10][13].

본 논문에서는 사례 연구로서 이벤트 편집기는 제외하고 속성 편집기만을 구현하여 설계 및 구현한 자바빈즈 컴포넌트를 시험하였다. 4.2.1절에서 구현된 컴포넌트들은 모두 자바빈으로 만들어져 그 속성 값을 편집하고 수정할 수 있다. 본 논문에서는 이러한 컴포넌트의 속성을 개발자의 코딩 없이 처리하기 위하여 다음과 같이 4개의 사용자 속성 편집기를 구현하였다.

- UrlEditor : JdbcConnection빈의 URL 속성 편집용
- RecordEditor : 그리드 빈의 SQL 속성 편집용
- TableEditor : TableDefinition 빈의 table 속성 편집용
- FieldEditor : 그리드를 제외한 DBAccess 빈의 SQL 속성 편집용

4.2.3 구현 결과 및 분석

본 논문에서 제안한 개발 환경은 JDK1.1.7과 스윙 1.0.3을 기반으로 구현하였다. JDBC 드라이버는 썬사가 분류한 4가지 드라이버 타입 중 2-계층 구조에서 사용할 수 있으며 속도가 가장 빠른 타입 4 드라이버를 사용하였다 [19]. 서버의 데이타베이스는 오라클7을 사용하였으며 JDBC 드라이버는 오라클사에서 제공하는 JDBC 씬(thin) 드라이버를 사용하였다.

제안된 개발 환경의 구현된 전체 화면은 그림 9와 같다. 그림의 상단에 메인 화면과 메뉴 및 툴바, 그리고 자바빈 컴포넌트들이 팔레트 형태로 나타나있다. 그림 9의 중앙에 나타난 것이 폼 디자이너이며, 상단의 팔레트에서 빈 컴포넌트를 선택하여 폼 상에 클릭하면 빈이 클릭한 위치에 배치된다. 폼 상에 배치된 빈들의 계층 구조가 폼 디자이너 좌측의 응용 관리기에 트리 형태로 표시되고, 개발자가 선택한 빈의 속성 및 이벤트가 폼 디자이너 우측의 속성/이벤트 편집기에 표시된다. 개발자가 폼 상에 빈을 배치하고 배치한 빈을 응용 관리기와 속성/이벤트 편집기를 통하여 수정하는 등의 모든 행위는 즉시 그림 아래의 소스 코드 편집기로 반영된다. JDBC 접속 및 화면 설계용 데이타베이스 컴포넌트를 모두 자바빈으로 구현하였다. 개발자는 단지 컴포넌트 팔레트 상의 컴포넌트를 드랙앤드랍만으로 쉽게 화면을 설계할 수 있다. 또한 보이지 않는 빈 형태로 제공되는 접속 및 테이블 설정 컴포넌트를 이용하여 개발자의 코딩과정을 속성 편집기를 이용하여 시각화하고 이에 대응하는 소스 코드 생성을 자동화하였다.

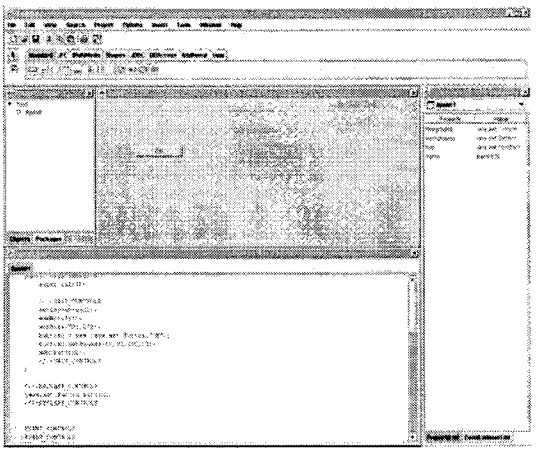


그림 9 제안된 개발 환경의 전체화면

개발자는 제안된 개발 환경을 이용하여 데이타베이스에 접근하는 클라이언트/서버 응용을 접속 컴포넌트를 사용하면 그림 4와 같은 소스 코드의 코딩 없이 시각적으로 프로그램할 수 있다. 그림 10은 제안된 자바빈즈를 이용하여 만든 자바 애플릿의 예이다. JDBC를 이용하여 데이타베이스에 접근하는 자바 애플릿을 개발하는 과정은 그림 11에서 그림 13까지 나타나 있는 것과 같이 시각적으로 편집 및 수정할 수 있다. 먼저 JdbcConnection 빈을 폼 상에 위치시키고 접속을 원하는 데이타베이스의 URL과 JDBC 드라이버 명, 사용자 및 패스워드를 설정한다. 그림 11은 접속 컴포넌트의 속성과 이중에서 URL을 편집하는 화면이다. 그 다음 TableDefinition 빈을 폼 상에 위치시킨 후 이전에 접속한 정보에서 원하는 질의문을 시각적으로 선택하여 레코드를 정의한다. 이 과정은 그림 12에 나타나 있다. 이렇게 접속 정보를 설정하고 원하는 질의문을 처리하는 과정이 단지 해당 자바빈의 배치와 이들의 사용자 속성 편집기만을 통하여 이루어지므로 개발자는 마우스의 클릭만으로 응용의 개발을 쉽게 할 수 있다. 마지막으로 설정한 레코드 정보를 그리드로 보여주기 위하여 그리드 빈을 폼 상에 위치시키고 그리드의 속성인 레코드 필드를 설정해주면 된다.

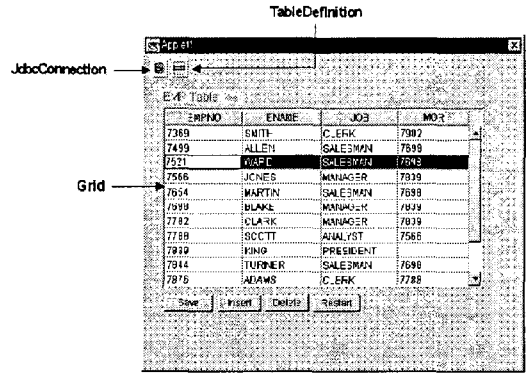


그림 10 구현 예

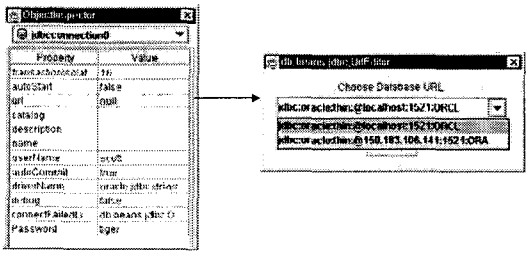


그림 11 접속 컴포넌트의 속성 및 사용자정의

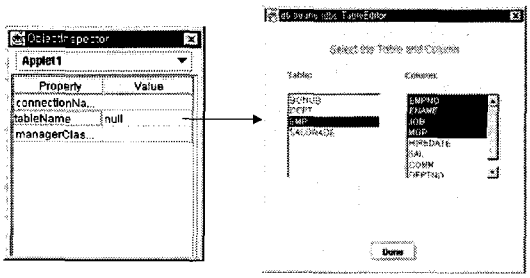


그림 12 TableDefinition 컴포넌트의 속성 및 사용자정의

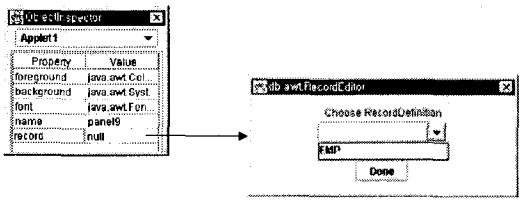


그림 13 그리드 컴포넌트의 속성 및 사용자정의

5. 결론 및 향후 과제

본 논문에서는 JDBC를 이용하여 클라이언트/서버 응용의 개발을 시각적으로 용이하게 하는 자바빈즈 컴포넌트들을 개발하고 이들의 속성을 편집할 수 있는 사용자 속성 편집기를 설계하고 구현하였다. 제안된 개발 환경은 JDBC를 이용한 데이타베이스 접속 및 화면 설계를 소스 코딩 없이 단지 관련 자바빈즈 컴포넌트들을 선택하고 이들의 속성을 편집함으로써 쉽게 개발할 수 있다. 또한 화면상의 설계 정보가 실시간으로 소스 코드로 생성되므로 클라이언트/서버 응용의 개발 및 수정을 보다 용이하게 하며 생산성을 높일 수 있다.

개발된 환경은 순수한 자바 언어로 구현되어 아직 성능 면에서 문제가 될 수 있다. 현재 제안한 개발 환경은 JDK1.1.7과 스윙 1.0.3을 기반으로 하고 있는데, JDK1.2로 변환되고 핫스팟 등 향후 JDK의 지속적인 향상을 이용한다면 속도 면에서는 많이 개선될 것으로 보인다. 이외에도 데이타베이스에서의 한글처리와 다양한 리포트 양식의 지원, 각종 위저드 기능이 추가로 필요하며, 현재 많이 연구되고 있는 ORB(Object Request Broker)를 이용한 3-계층 클라이언트/서버 응용을 위해 확장될 필요가 있다.

참 고 문 헌

[1] G. Schussel, Client/Server Past, Present, and Future, Available WWW <URL:http://www.dciexpo.com/goes/>, 1995.

[2] D. S Linthicum, *David Lithicum's Guide to Client/Server and Intranet Development*, John Wiley & Sons, Inc., 1997.

[3] John C. Zubeck, "Implementing Reuse with RAD Tools' Native Objects," *IEEE Computer*, Vol. 30, No. 10, pp. 60-65, Oct. 1997.

[4] Rosemary Rock-Evans, *Ovum Evaluates GUI Builders*, Ovum Limited, 1994.

[5] E. Wegscheider, "Toward Code-Free Business Application Development," *IEEE Computer*, Vol. 30, No. 3, pp. 35-43, Mar. 1997.

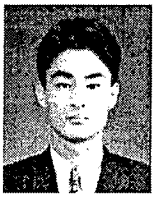
[6] R. Orfali, D. Harkey, *Client/Server Programming with JAVA and CORBA*, John Wiley & Sons, Inc., 1997

[7] E. Yourdan, "Java, the Web, and Software Development," *IEEE Computer*, Vol. 29, No. 8, pp. 25-30, Aug, 1996.

[8] M. A. Hamilton, "Java and the shift to Net-Centric Computing," *IEEE Computer*, Vol. 29, No. 8, pp. 31-39, Aug, 1996.

[9] Software Engineering Institute, *Software Technology Reference Guide*, CMU/SEI-97-HB-001, Jan. 1997.

[10] David Krieger, Richard M. Alder, "The Emergence of Distributed Component Platforms," *IEEE Computer*, Vol. 31, No. 3, pp. 43-53, Mar. 1998.
[11] Richard M. Alder, "Emerging Standards for Component Software," *IEEE Computer*, Vol. 28, No. 3, pp. 68-77, Mar. 1995.
[12] James Gosling et al., *The Java™ Application Programming Interface*, Vol. 1&2, Addison Wesley, 1996.
[13] Sun Microsystems, Inc., JavaBeans Specification 1.01, July 1997, http://www.javasoft.com/beans/docs/spec.html.
[14] J. Weber, *Special Edition Using JAVA 1.1, 3rd Ed.*, Que, 1997.
[15] Visual Basic 5.0 Users Guide, Microsoft, 1996.
[16] PowerBuilder 5.0 Users Guide, Sybase/PowerSoft, 1996.
[17] Visual Café 2.0 dbDE Users Guide, Symantec, 1997.
[18] 유철중, "Java 통합 개발 환경에서 기능 컴포넌트들의 상호연동 기법", 정보처리학회논문지, 제5권 제11호, pp. 2862-2873, 1998.
[19] M. Sood, Examining JDBC Drivers, *Dr. Dobbs Journal*, pp. 82-87, Jan 1998.



손 승 우
1995년 영남대학교 공대 컴퓨터공학과 졸업(학사). 1997년 동 대학원 컴퓨터공학과 졸업(석사). 1996년 12월 ~ 현재 한국전산통신연구원 연구원. 관심분야는 소프트웨어 공학, 실시간 시스템, 실시간 운영체제

김 순 용
1986년 서울산업대학교 전자계산학과(학사). 1995년 대전대학교 컴퓨터공학과(석사). 1981년 ~ 현재 한국전자통신연구원 선임연구원. 관심분야는 소프트웨어공학, 실시간 운영체제, 정보처리

김 창 갑
1984년 2월 경기대 경영학과 졸업. 1983년 12월 KAIST 전산개발센터 연구원. 1992년 3월 시스템공학연구소 선임연구원. 1998년 5월 한국전자통신연구원 선임연구원.

이 상 덕
1980년 부산대학교 대학원 경영학과 졸업(석사). 1978년 ~ 현재 한국전자통신연구원 책임연구원