컴포넌트 기반 웹 데이터베이스 응용의 자동 생성기

음 두 헌[†] · 고 민 정[††] · 강 이 지[††]

요 약

인터넷 기술의 급속한 발전과 함께 전자 상거래가 활성화 되고 있다. 이러한 전자 상거래 응용의 핵심은 웹 기반 데이터베이스 응용이다. 현재는 데이터베이스 응용에 필요한 모든 폼과 질의 처리코드를 수동 또는 반자동으로 작성하므로 웹 응용 개발에 많은 시간이 소요된다. 따라서 웹 기반 데이터베이스 응용의 생산성 향상이 요구되고 있다. 본 논문에서는 데이터베이스부터 새롭게 생성해야 하는 응용과 기존 데이터베이스를 사용하는 응용의 생성을 위한 사용자 인터페이스 폼들과 이 폼들을 통해 이뤄질 질의를 처리하는 EJB 및 JSP 컴포넌트들을 자동으로 생성하는 도구인 WebSiteGen2를 소개한다. WebSiteGen2는 컴포넌트 기술을 기반으로 3-계층(3-tier) 구조를 가지는 응용을 자동 생성함으로써 웹 응용의 생산성을 향상시키고 확장성, 재사용성 및 이식성을 증대시킨다. 또한, WebSiteGen2가 생성하는 사용자 인터페이스 폼들은 질의의 대상인 개체 뿐 아니라 이와 직·간접으로 연관된 모든 개체들에 대한 정보를 한 폼에 제공한다. 본 논문에서는 WebSiteGen2의 기능 및 구현 원리를 설명하고, 상용화된 타 웹 응용 생성기들과의 기능을 비교하여 WebSiteGen2의 장점을 설명한다.

Automatic Generator for Component-Based Web Database Applications

Doohun Eum[†] · Minjeung Ko[††] · Izzy Kang[††]

ABSTRACT

E-commerce is in wide use with the rapid advance of internet technology. The main component of an e-commerce application is a Web-based database application. Currently, it takes a lot of time in developing Web applications since developers should write codes manually or semi-automatically for user interface forms and query processing of an application. Therefore, the productivity increase of Web-based database applications has been demanded. In this paper, we introduce a software tool, which we call the WebSiteGen2, that automatically generates the forms that are used as user interfaces and the EJB/JSP components that process the query made through the forms for an application that needs a new database or uses an existing database. The WebSiteGen2 thus increases the productivity, reusability, expandibility, and portability of an application by automatically generating a 3-tier application based on component technology. Moreover, one user interface form that are generated by the WebSiteGen2 provides information on an interested entity as well as information on all the directly or indirectly related entities with the interested one. In this paper, we explain the functionality and implementation of the WebSiteGen2 and then show the merits by comparing the WebSiteGen2 to the other commercial Web application generators.

키워드 : 데이터베이스(Database), 컴포넌트(Component), 자동생성(Automatic Generation), EJB, JSP

1. 서 론

인터넷 및 네트워크 관련 서비스의 확산과 더불어 전자 상거래 응용의 수요가 급증하고 있다. 전자 상거래 응용의 핵심 요소는 웹 기반 데이터베이스 응용이다. 전자상거래에 사용되는 데이터베이스는 응용 환경이 다양해짐에 따라 복잡한 구조를 가지게 된다. 따라서 프로그래머가 수작업으로 데이터베이스 응용을 위한 사용자 인터페이스 폼 및 질의 처리코드를 설계하고 구현하는데는 많은 시간이 소요된다[13].

본 논문에서는 웹 기반 데이터베이스 응용의 생산성 향상을 위해, 사용자 인터페이스로 이용될 HTML 폼들과 이 폼들을 통해 이뤄질 질의를 처리하는 EJB(Enterprise Java Beans) 및 JSP(Java Servlet Pages) 컴포넌트들을 자동으로 생성하는 도구인 WebSiteGen2를 소개한다. 응용 데이터베이스는 WebSiteGen2를 이용해 새롭게 구축하거나[1] 기존의 데이터베이스를 사용할 수 있다.

웹 응용에서 데이터베이스의 사용은 주로 HTML 폼을 통해 이루어진다. 폼은 사용자 인터페이스를 제공하므로 데이터의 검색과 갱신을 쉽게 해준다. 사용하기 쉽고 연관된 정보들을 한번에 제공할 수 있는 폼의 설계 및 구현은 웹 응용의 핵심적인 구성요소이다. 잘 알려진 상용 웹 응용 생성기로는 ASP.NET[3], CodeCharge[4], WebDBGenerator[5] 등이 있다. ASP.NET은 Microsoft가 추진하는 .NET 기술의 일부로서 ASP 질의 처리코드가 포함된 폼을 반자동

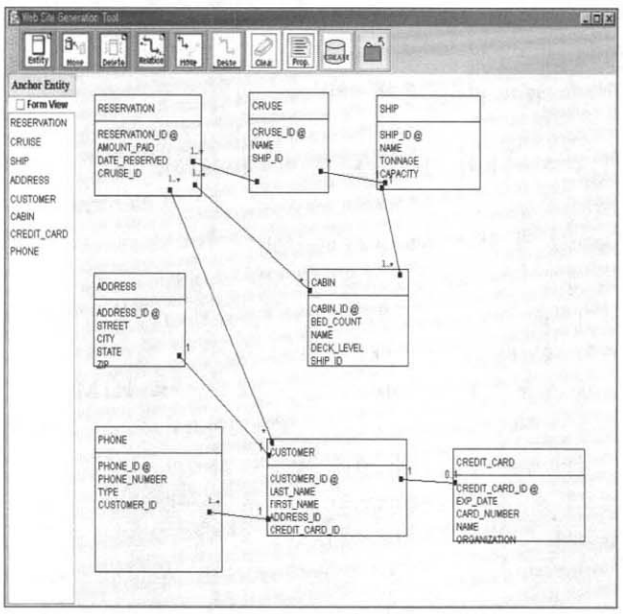
※ 본 논문은 2003년 덕성여대 자연과학연구소 연구비 지원으로 연구되었음.
† 정 회 원 : 덕성여자대학교 컴퓨터과학과 교수
†† 준 회 원 : 덕성여자대학교 대학원 전산·정보통신학과
논문접수 : 2003년 7월 24일, 심사완료 : 2003년 11월 17일

으로 생성한다. CodeCharge는 새로운 데이터베이스를 구축할 수는 없으나 질의 처리코드와 폼을 반자동으로 생성한다. WebDBGenerator는 텍스트 입력을 통하여 데이터베이스를 구축하고 폼을 생성한다. 그러나 이러한 상용 생성기들이 생성하는 폼들은 사용자가 필요로 하는 관련 정보를 한 폼에 모두 표현하지 못한다. 본 논문에서 소개하는 WebSiteGen2가 생성하는 폼은 현재 관심의 대상인 닻 개체(anchor entity)는 물론 닻 개체와 일대일, 다대일 관계 타입으로 연관되는 첨부 개체(appended entity)와 닻 개체 또는 첨부 개체와 일대다 또는 다대다 관계 타입으로 연관되는 확장 개체(expended entity)들 및 이 확장 개체들에 대한 첨부 개체들의 정보를 한 폼에 모두 표현할 수 있다. 또한, ASP.NET, CodeCharge, WebDBGenerator는 2-계층 구조를 가지는 응용을 생성하는 반면 WebSiteGen2는 EJB와 JSP의 컴포넌트 기술을 도입한 3-계층 구조를 가지는 컴포넌트 기반 데이터베이스 응용을 생성한다.

2절에서 WebSiteGen2의 개요를 설명하고 3절에서 타 웹 응용 생성기들과의 기능을 비교한 후, 4절에서는 WebSiteGen2의 구현을 간략히 설명한다. 마지막 절에서는 본 논문의 결론과 앞으로의 연구 방향을 정리한다.

2. WebSiteGen2의 개요

본 절에서는 웹 기반 데이터베이스 응용을 자동으로 생성하는 도구인 WebSiteGen2의 개요를 소개한다. WebSiteGen2의 기능을 설명하기 위해 본 논문에서는 간단한 전자상거래 응용(Titan 유람선 예약 시스템[6])을 예로 들어 WebSiteGen2의 기능을 설명한다.



(그림 1) Titan 응용의 클래스 다이어그램

(그림 1)은 WebSiteGen2로 작성한 Titan 데이터베이스 응용의 UML 클래스 다이어그램이다[2, 10-11, 16]. 데이터베이스와 연동될 엔티티 빈들의 방향성은 양방향을 가정한다.

Titan 응용은 다음을 가정한다.

① 고객(CUSTOMER)은 고객의 이름, 주소, 신용카드에 관한 정보로 관리된다. 각 고객에게는 아이디가 기본키로 주어지고, 고객은 신용카드(CREDIT_CARD)를 하나 가지거나 가지지 않을 수 있고, 하나 이상의 전화(PHONE)와 하나의 주소(ADDRESS)를 가지며 한 개 이상의 예약(RESERVATION)을 할 수 있다.

② 예약은 지불된 요금, 예약날짜, 그리고 다수의 객실(CABIN)을 가지며 하나의 선박여행(CRUISE)과 연계되고 다수의 고객을 포함할 수 있다.

③ 선박여행은 하나의 선박(SHIP)이 배정되고 한 개 이상의 예약을 가진다.

④ 선박은 선박의 이름, 무게, 용적과 함께 다수의 객실을 보유한다.

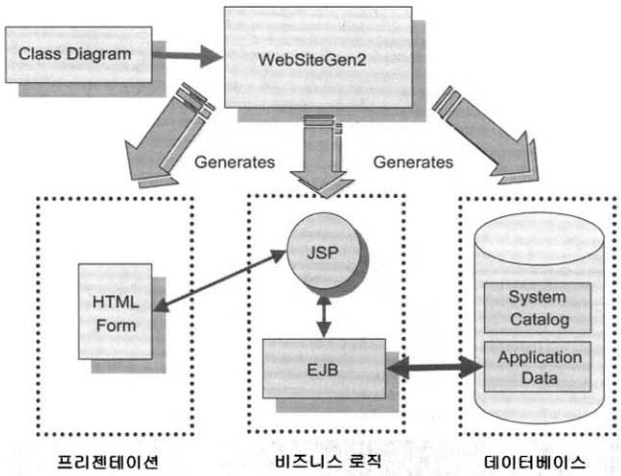
⑤ 신용카드는 카드번호, 유효기간, 발급기관, 이름, 그리고 카드를 소유한 한 명의 고객으로 구성된다.

⑥ 전화는 전화번호, 타입, 그리고 해당 번호를 사용하는 한 명의 고객으로 구성된다.

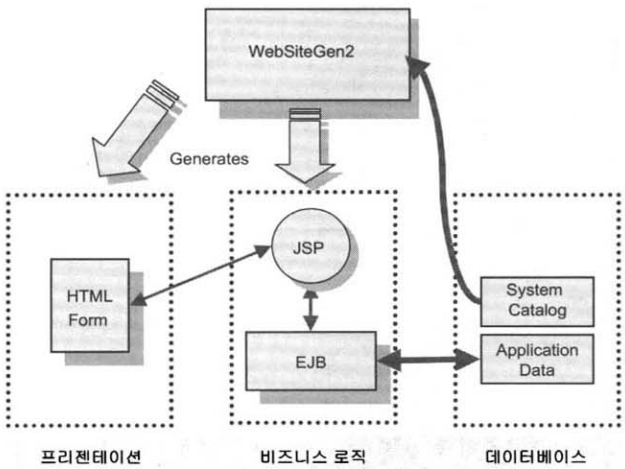
⑦ 주소는 거리, 도시, 주, 우편번호, 그리고 해당 주소에 거주하는 한 명의 고객으로 구성된다.

(그림 2)는 WebSiteGen2를 이용해서 3-계층 구조를 가지는 데이터베이스 응용을 생성하는 두 가지 방법을 보인 것이다. 첫 번째 방법을 보인 (그림 2)(a)에서 WebSiteGen2는 데이터베이스 응용을 위한 클래스 다이어그램으로부터 카디널리티와 참조 무결성 조건 등의 정보를 추출해 필요한 응용 데이터베이스를 create table 문으로 구축하고 사용자 인터페이스로 사용될 HTML 폼들을 생성하며 이 폼을 통해 이루어질 질의를 처리하는 EJB와 JSP 컴포넌트들을 생성할 수 있다. 두 번째 방법은 (그림 2)(b)와 같이, 기존 데이터베이스의 시스템 카탈로그로부터 스키마 정보를 추출해 폼들과 컴포넌트들을 자동 생성할 수 있다. WebSiteGen2는 현재 ORACLE8i 서버[7]와 MSSQL 서버 7.0[8]으로 구축된 기존 데이터베이스를 이용할 수 있다. 본 논문에서는 ORACLE8i 서버의 사용을 가정한다.

한 폼에서 현재 관심의 대상인 개체를 닻 개체라 한다. 닻 개체와 일대일 또는 다대일 관계 타입으로 연관되는 개체를 첨부 개체라 하고 닻 개체와 일대다 또는 다대다 관계 타입으로 연관되는 개체를 확장 개체라 한다. 첫 번째 방법인 클래스 다이어그램으로부터 응용을 생성할 경우는 각 개체의 카디널리티 정보로부터 첨부 및 확장 개체들을 결정 할 수 있다[1, 12]. 두 번째 방법인 기존의 데이터베이스를 이용하여 응용을 생성하는 경우는 다음과 같다.



(a) 클래스 다이어그램으로부터 응용 생성



(b) 기존 데이터베이스로부터 응용 생성

(그림 2) WebSiteGen2

(그림 3)의 구조를 가지는 ORACLE8i의 시스템 카탈로그 테이블인 ALL_CONSTRAINTS는 테이블들에 대한 관계 정보를 포함한다. CONSTRAINT_NAME은 각 테이블에 대한 제약사항 이름이고 R_CONSTRAINT_NAME은 참조되는 테이블의 고유 CONSTRAINT_NAME이다.

이 름	타 입	설 명
OWNER	VARCHAR2(30)	제약사항 정의의 소유자(Owner)
CONSTRAINT_NAME	VARCHAR2(30)	제약사항 정의 이름
TABLE_NAME	VARCHAR2(30)	제약사항 정의를 가진 테이블 이름
R_OWNER	VARCHAR2(30)	참조 제약사항의 테이블 소유자 (Owner)
R_CONSTRAINT_NAME	VARCHAR2(30)	참조 테이블을 위한 유일한 제약사항 정의의 이름
DELETE_RULE	VARCHAR2(9)	참조가 있는 제약사항을 위한 삭제 규칙

(그림 3) ALL_CONSTRAINTS 테이블

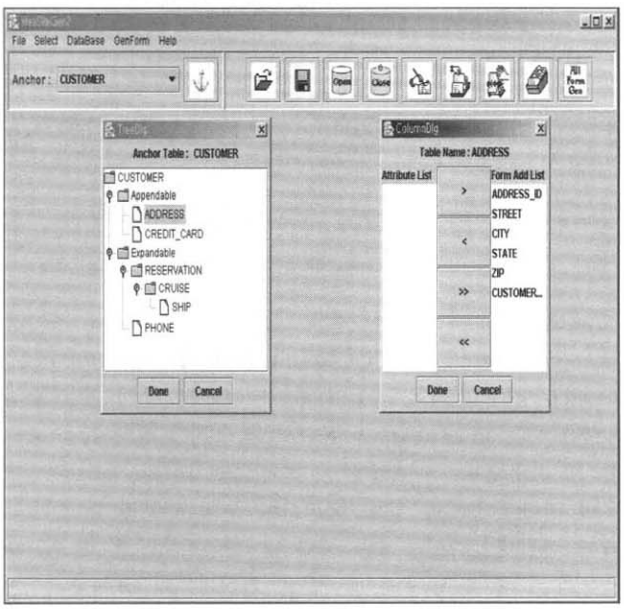
따라서 CONSTRAINT_NAME과 R_CONSTRAINT_NAME 필드를 통해 각 개체에 대한 첨부 개체와 확장 개체에 관한 정보를 추출할 수 있다.

아래의 SQL 문을 이용하면 닻 개체인 CUSTOMER 개체에 대한 첨부 개체들을 결정할 수 있다.

```
SELECT B.TABLE_NAME
FROM ALL_CONSTRAINTS A, ALL_CONSTRAINTS B
WHERE A.TABLE_NAME = UPPER('CUSTOMER')
AND B.CONSTRAINT_NAME = A.R_CONSTRAINT_NAME ;
```

또한, 다음의 SQL문을 이용하면 CUSTOMER 개체에 대한 확장 개체들을 결정할 수 있다.

```
SELECT B.TABLE_NAME
FROM ALL_CONSTRAINTS A, ALL_CONSTRAINTS B
WHERE A.TABLE_NAME = UPPER('CUSTOMER')
AND A.CONSTRAINT_NAME = B.R_CONSTRAINT_NAME ;
```



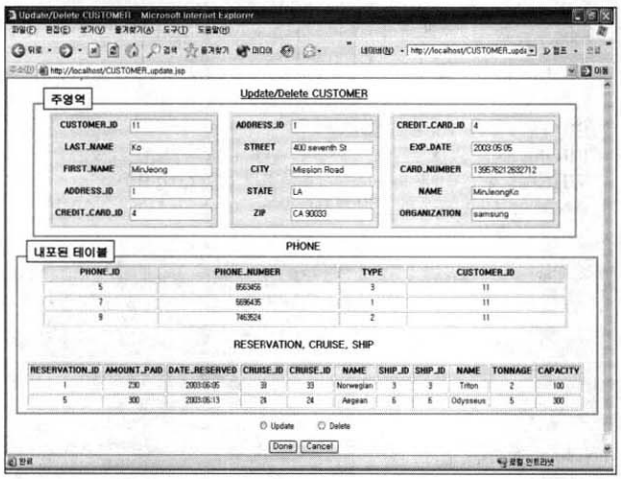
(그림 4) TreeDlg와 ColumnDlg 다이얼로그 박스

WebSiteGen2는 첫 번째와 두 번째 방법으로 추출된 첨부 및 확장 개체들에 대한 정보를 (그림 4)와 같이 트리 형식으로 TreeDlg 창에 나타내고 ColumnDlg 창에는 TreeDlg 창에서 선택된 개체의 속성들을 나열해 생성될 폼에 포함 여부를 선택할 수 있게 한다. TreeDlg 창에서 Appendable 노드 밑의 개체들은 첨부 개체들이고 Expandable 노드 밑의 개체들은 확장 개체들이다. 두 다이얼로그 박스를 통해 선택된 개체 및 속성들은 (그림 5)와 같이 한 폼에 표현된다.

(그림 4)에서 고객(CUSTOMER)이 닻 개체일 때, 고객은 신용카드(CREDIT_CARD)와 일대일 관계 타입으로 연관되고 주소(ADDRESS)와도 일대일 관계타입으로 연관되므로 신용카드와 주소는 첨부 개체가 되어 Appendable 노드 밑에 표시된다. 고객과 일대다 관계 타입인 전화(PHONE)와 예

약(RESERVATION)은 확장 개체가 된다. 첨부 개체에는 그 개체의 첨부 개체와 확장 개체가 다시 추가될 수 있고 확장 개체에는 그 개체의 첨부 개체가 추가될 수 있다. 따라서 예약의 첨부 개체인 선박여행(CRUISE)과 선박여행의 첨부 개체인 선박(SHIP)도 Expandable 노드 밑에 계속 추가된다.

2.1 폼의 생성



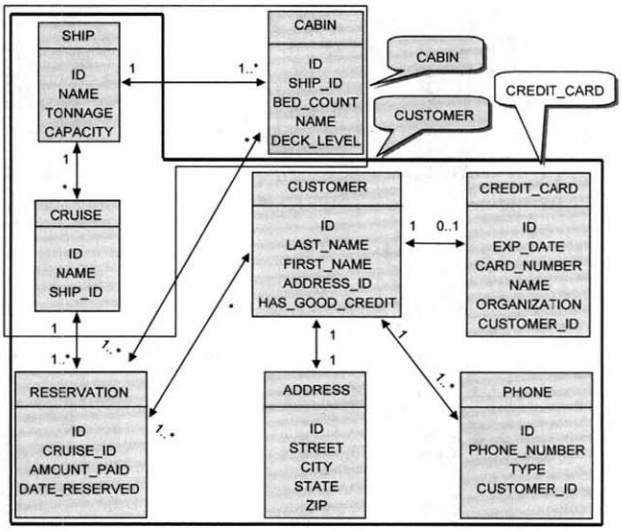
(그림 5) Titan 응용의 Update/Delete CUSTOMER 폼

WebSiteGen2가 생성하는 폼의 종류는 Insert/Search 폼, Select 폼, Update/Delete 폼, Result 폼의 네 가지이다. Insert/Search 폼은 데이터를 삽입하거나 검색할 때 사용된다[14]. Select 폼은 Insert/Search 폼으로 검색한 결과 목록을 제공할 때 사용된다. Select 폼의 목록 중 하나의 항목을 선택하면 Update/Delete 폼이 생성되어 선택된 항목(닻 개체)과 함께 연관된 정보들(첨부 개체 및 확장 개체)을 제공한다. 사용자는 이 폼을 통해 데이터를 갱신하거나 삭제할 수 있다. Result 폼은 수정, 삭제에 대한 결과를 보여준다.

(그림 5)는 CUSTOMER를 닻 개체로 하는 WebSiteGen2가 생성한 Update/Delete 폼이다.

- 닻 개체와 일대일 또는 다대일 관계인 개체를 첨부 개체라 하고 일대다 또는 다대다 관계인 개체를 확장 개체라 한다.
- 첨부 연산은 모든 개체(닻 개체, 첨부 개체, 확장 개체)에 적용 가능하며 하나의 폼 내에서 첨부 연산의 회수에는 제한이 없다.
- 확장 연산은 닻 개체 또는 닻 개체에 직·간접적으로 첨부된 개체로부터 확장할 때 사용된다. 하나의 폼에는 한번의 확장을 통해 도달할 수 있는 개체들만이 포함될 수 있다.

Update/Delete 폼은 (그림 5)의 Update/Delete CUSTOMER 폼과 같이 주 영역(main area)과 내포된 테이블(nested table)들로 구성된다. 주 영역에는 닻 개체와 첨부 개체에 대한 정보를 보인다. 내포된 테이블에는 닻 개체와 첨부 개체에 대한 확장 개체 및 이 확장 개체에 대한 첨부 개체에 관한 정보를 보인다. Update/Delete 폼은 아래의 그룹화 규칙에 따라 구성된다. 첨부는 제한이 없으나 내포는 한번만 가능하다.



(그림 6) 각 폼이 커버하는 개체 그룹

Titan 응용에 이러한 그룹화 규칙을 적용하면 3개의 Update/Delete 폼들(CUSTOMER, CABIN, CREDIT_CARD)을 얻을 수 있으며 (그림 6)은 각 폼이 커버하는 개체 타입들을 보인 것이다. 데이터베이스와 연동될 엔티티 빈들은 방향성이 있으므로 엔티티 빈들의 생성을 위해 이 방향성은 화살표로 표기하며 WebSiteGen2에서는 모두 양방향을 가정한다.

(그림 5)에 보인 Update/Delete CUSTOMER 폼의 닻 개체 타입은 CUSTOMER이다. CUSTOMER는 개체 타입 CREDIT_CARD와 일대일 관계 타입으로 연관된다. 따라서 그룹화 규칙에 의해 CREDIT_CARD는 첨부 개체 타입이 될 수 있다. ADDRESS 개체 타입도 CUSTOMER와 일대일 관계 타입으로 연관되므로 첨부될 수 있다. PHONE 개체 타입은 CUSTOMER와 일대다 관계 타입으로 연관되므로 규칙에 의해 확장 개체 타입이 될 수 있고 RESERVATION 개체 타입은 CUSTOMER와 다대다 관계 타입으로 연관되므로 확장 관계 타입이 된다. CRUISE 개체 타입은 RESERVATION과 다대일 관계 타입으로 연관되므로 내포된 테이블에 첨부된다. 마지막으로 SHIP 개체 타입은 CRUISE와 다대일 관계 타입으로 연관되므로 규칙에 의해 CRUISE에 첨부될 수 있다. 비슷한 방법으로 CABIN과 CREDIT_ CARD를 닻 개체 타입으로 하는 Update/Delete CABIN 폼과 Update/Delete CREDIT_CARD 폼도 생성된다.

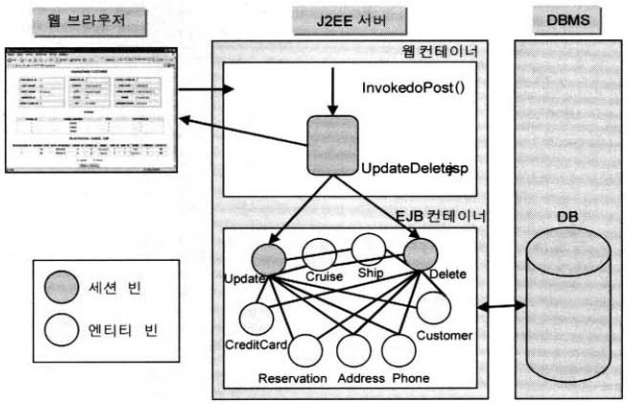
2.2 EJB 및 JSP 컴포넌트의 생성

EJB는 컴포넌트를 기반으로 하는 분산 비즈니스 응용의 개발과 배치를 위한 표준 아키텍처이다. EJB에서는 3-계층(3-tier) 구조를 가지는 응용의 비즈니스 로직을, 데이터 로직을 처리하는 엔티티 빈(entity bean)과 순수 비즈니스 로직만을 담당하는 세션 빈(session bean)의 두 가지로 나눈다[15]. 엔티티 빈은 데이터베이스의 엔티티 개체에 대한 맵핑으로 볼 수 있다. 세션 빈은 서로 다른 빈 간의 상호작용을 담당하는 비즈니스 메소드를 가지며 엔티티 빈과는 달리 빈 내부에서 사용자 정의 트랜잭션 처리도 가능하다. 웹 컴포넌트인 JSP는 동적 HTML 코드를 생성하는 과정을 간단하게 만드는 서블릿 컴포넌트 모델을 확장한 것이다. JSP는 JDBC(Java Database Connectivity)를 이용하거나 엔티티 빈을 이용해 데이터베이스에 접근할 수 있다. JSP가 동적으로 생성하는 HTML 폼은 자바 애플릿에 비해 용량이 매우 작고 기본 플랫폼에서도 실행되며 전자 상거래에서 보안을 위해 설치한 방화벽을 통과할 수 있기 때문에 JSP는 3-계층 응용의 프리젠테이션 로직을 담당하는 컴포넌트로 적합하다.

WebSiteGen2는 (그림 4)의 TreeDlg 창과 ColumnDlg 창을 통해 얻은 정보를 토대로 사용자 인터페이스 폼들을 생성 한 후, 이 폼들을 통해 이루어질 질의를 처리할 EJB 및 JSP 컴포넌트들을 생성한다. 생성될 EJB 컴포넌트들은 데이터베이스 내의 모든 개체 타입에 대한 데이터 로직을 담당하는 엔티티 빈들과 응용의 비즈니스 로직을 담당하는 세션 빈들이다. 세션 빈은 각 개체 타입 별로 비즈니스 로직에 맞게 InsertBean, SelectBean, UpdateBean, DeleteBean의 4가지씩 생성된다. 생성될 JSP 컴포넌트는 사용자 인터페이스인 HTML 폼과 함께 질의 결과를 웹 브라우저에 보내 사용자에게 보여주거나 폼으로부터 받은 질의를 처리하기 위해 적절한 세션 빈을 호출한다. JSP 컴포넌트는 각 개체 타입 별로 사용자 인터페이스 폼에 따라 Insert.jsp, Select.jsp, UpdateDelete.jsp, Result.jsp의 4가지씩 생성된다.

WebSiteGen2가 생성하는 웹 응용 중 EJB 컴포넌트는 RMI-IIOP 프로토콜을 이용해 분산 객체간의 통신을 담당하고, JSP 컴포넌트는 방화벽을 통과하기에 적합한 HTML 코드를 동적으로 생성한다. 따라서 생성된 컴포넌트들은 분산과 방화벽 문제를 모두 해결할 수 있는 조합이 된다.

(그림 7)은 WebSiteGen2가 생성한 Titan 응용의 구조도이다. WebSiteGen2는 J2EE 서버 환경을 가정한다. (그림 7)에서 J2EE 서버의 웹 컨테이너엔 JSP 컴포넌트들이 위치하고 EJB 컨테이너엔 EJB 컴포넌트들이 위치한다. 사용자 인터페이스인 HTML 폼으로부터 질의를 받게 되면 해당 JSP 컴포넌트가 호출되고 이 JSP 컴포넌트는 질의 처리 기능을 가진 해당 세션 빈을 호출한다. 호출된 세션 빈은 데이터베이스 테이블들과 연동되는 엔티티 빈들을 조작하여 질의 결과를 얻게 되고 이는 JSP 컴포넌트를 통해 다시 웹 브라우저에 전송되어 폼에 보이게 된다.



(그림 7) Titan 응용의 구조

(그림 7)에서 (그림 5)의 CUSTOMER 폼과 같은 HTML 폼으로부터 데이터 갱신을 위한 질의를 입력받게 되면 갱신 작업을 수행하기 위한 JSP 컴포넌트인 UpdateDelete.jsp가 호출되고 이는 다시 갱신 작업을 위한 세션 빈인 UpdateBean을 호출한다. 호출된 UpdateBean은 닻 개체에 해당되는 엔티티 빈과 그 닻 개체와 연관된 첨부 개체와 확장 개체에 해당하는 엔티티 빈들의 데이터를 갱신하는 비즈니스 로직을 수행하고 이는 해당 데이터베이스 테이블에 바로 반영된다. 삭제 질의도 비슷한 과정으로 수행된다. 데이터 입력과 검색을 위한 질의를 처리할 때도 마찬가지로 Insert.jsp와 Select.jsp 등이 사용된다.

WebSiteGen2가 생성하는 컴포넌트들은 이식성과 확장성이 좋으며 그 중 엔티티 빈들은 다른 응용을 위한 재사용성이 매우 높다. 또한, 다양한 비즈니스 로직을 수행하는 세션 빈들과 조립된다면 생성된 데이터베이스 응용을 완전한 웹 응용으로 쉽게 확장할 수도 있다.

3. 상용 웹 응용 생성기와의 기능 비교

본 절에서는 WebSiteGen2와 상용 웹 응용 생성기들의 기능을 비교한다. <표 1>은 WebSiteGen2와 상용 웹 응용 생성 도구인 ASP.NET, CodeCharge, WebDBGenerator와의 기능을 비교한 것이다.

ASP.NET은 Microsoft가 추진하는 .NET 기술의 일부이다. Visual Studio.Net과 SQL Server 데이터베이스를 이용하여 웹 응용을 자동으로 생성한다. ASP.NET은 폼에 필요한 GUI 요소들을 선택적으로 구성하여 반자동으로 폼을 생성할 수 있다. 한 폼에 표현 가능한 개체는 닻 개체와 첨부 개체이다. ASP.NET에서 제공하는 Web Matrix 툴을 이용하면 텍스트 입력을 통해 새로운 테이블 생성 및 기존 데

〈표 1〉 상용 웹 응용 생성기와의 기능 비교

		WebSiteGen2	ASP.NET	CodeCharge	WebDB Generator
프리젠테이션	한 폼에 표현 가능한 개체	닻 개체 첨부 개체 확장 개체	닻 개체 첨부 개체	닻 개체 첨부 개체	닻 개체
	폼 생성	자동	반자동	반자동	반자동
비즈니스 로직	컴포넌트 사용유무	○	×	×	×
	생성코드	JSP와 EJB 컴포넌트	C#, Visual Basic, JScript	HTML, JavaScript, XML and XSL	ASP, JavaScript, VBScript, HTML
데이터베이스	데이터베이스 생성	클래스 다이어그램 이용	Web Matrix 툴 이용	×	텍스트 입력 폼 이용
	기존 데이터베이스 사용	○	○	○	○
전체	생성된 응용의 구조	3-계층	2-계층	2-계층	2-계층
	생성된 응용의 재사용성·확장성·이식성	+++	++	+	+

이터베이스 접근이 가능하다. 데이터베이스에 접근한 후에, 워크스페이스(Workspace) 창에 생성된 데이터베이스나 기존 데이터베이스의 이름을 부모 노드로 갖는 트리 뷰어를 보여준다. 부모 노드는 두 개의 자식 노드를 포함하는데, Tables라는 자식 노드와 Stored Procedures라는 자식 노드를 갖는다. 생성된 응용은 2-계층 구조를 가지기 때문에 클라이언트측이 프리젠테이션과 비즈니스 로직을 담당하게 된다. 따라서 클라이언트-서버간의 통신 부하가 가중되어 WebSiteGen2가 생성하는 응용에 비해 성능 면에서 비효율적이다.

CodeCharge는 HTML, JavaScript, XML과 XSL 등의 사용자 인터페이스로 사용될 폼과 그 폼을 통해 이뤄질 질의를 처리하는 코드를 자동으로 생성해주는 웹 응용 생성기이다. 폼은 사용자의 선택에 따라 반자동으로 생성되며 하나의 폼은 ASP.NET과 같이 닻 개체와 첨부 개체를 표현할 수 있다. CodeCharge는 새로 데이터베이스를 생성할 수는 없고 기존에 구축된 데이터베이스를 이용하는 것만 가능하다. CodeCharge는 WebSiteGen2와 달리 웹 응용을 반자동으로 생성한다. CodeCharge가 제공하는 Application Builder라는 툴을 이용하면 사용할 데이터베이스의 테이블 리스트에서 각 테이블에 대해 검색, 삽입, 수정 기능의 제공 여부를 선택하는 과정을 거쳐 반자동으로 웹 응용을 생성할 수 있다. 생성된 응용은 2-계층 구조를 가지므로 ASP.NET과 같이 WebSiteGen2가 생성하는 응용에 비해 성능 면에서 비효율적이다. 또한, 컴포넌트 기반이 아니므로 재사용성, 확장성, 이식성 등도 떨어진다.

WebDBGenerator는 Internet 환경에서 작업이 이루어지므로 언제 어디서나 Internet에 접속하여 운영할 수 있다. 폼과 질의 처리코드로 ASP, JavaScript, VBScript, HTML 코드 등을 생성한다. 텍스트 입력을 통해 새로운 데이터베이스를 생성하므로 WebSiteGen2의 클래스 다이어그램을 이용하는 방식에 비해 비 시각적이다. WebSiteGen2와 같이 기존의 데이터베이스에 대한 접근도 가능하다. 그러나 WebDBGenerator가 반자동으로 생성하는 웹 응용은 WebSiteGen2가 생성하는 웹 응용과 달리 닻 개체에 관한 정보만 표현 할 수 있다. WebDBGenerator는 ASP.NET이나 CodeCharge와 같이 2-계층 구조를 가지는 응용을 생성하므로 WebSiteGen2가 생성하는 응용에 비해 비효율적이다.

<표 1>에서 보듯이 WebSiteGen2가 자동으로 생성한 폼은 닻 개체뿐만 아니라 첨부 개체와 확장 개체까지 한 폼에 표현이 가능하다. 따라서 닻 개체와 직·간접으로 일대일, 다대일, 일대다, 다대다로 연관된 개체들을 한 눈에 보며 조작할 수 있다는 장점이 있다. 또한, 생성된 응용은 성능면에서 우수한 3-계층 구조를 가지며 EJB 및 JSP 컴포넌트 기반이기 때문에 재사용성, 확장성, 이식성 등에도 타 상용 생성기들이 생성하는 응용에 비해 우수하다. 즉, 생성된 각 컴포넌트는 레고(Lego) 블럭과 같이 다른 컴포넌트들(레고블럭들)과 조립되어 비즈니스 응용을 신속하게 생성할 수 있게 하므로 응용의 생산성 향상을 지원한다. 또한, 이러한 컴포넌트들은 조립 및 분해가 용이할 뿐 아니라 다른 조합으로의 조립 및 수정이 용이하므로 재사용성, 확장성, 이식성 등을 지원한다[6].

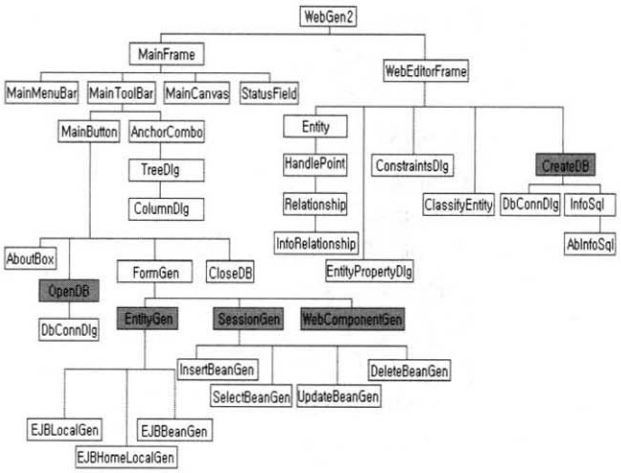
4. WebSiteGen2의 구현

본 절에서는 WebSiteGen2 프로토타입의 구현을 간략히 소개한다. WebSiteGen2는 Java로 구현되었으며 JSP 컴포넌트들과 함께 WebLogic 서버 6.0에 배치 가능한 EJB 2.0 컴포넌트들을 생성한다. 데이터베이스는 ORACLE8i 서버 및 MSSQL 서버 7.0을 사용할 수 있다. WebSiteGen2의 컴포넌트 계층구조 및 각 클래스에 대해 설명한 후, WebSiteGen2가 자동 생성하는 EJB 컴포넌트와 JSP 컴포넌트에 대해 설명한다.

4.1 WebSiteGen2의 컴포넌트 계층구조

(그림 8)은 WebSiteGen2의 컴포넌트 계층구조이다. 주요

클래스에 대한 설명은 다음과 같다.



(그림 8) WebSiteGen2의 컴포넌트 계층구조

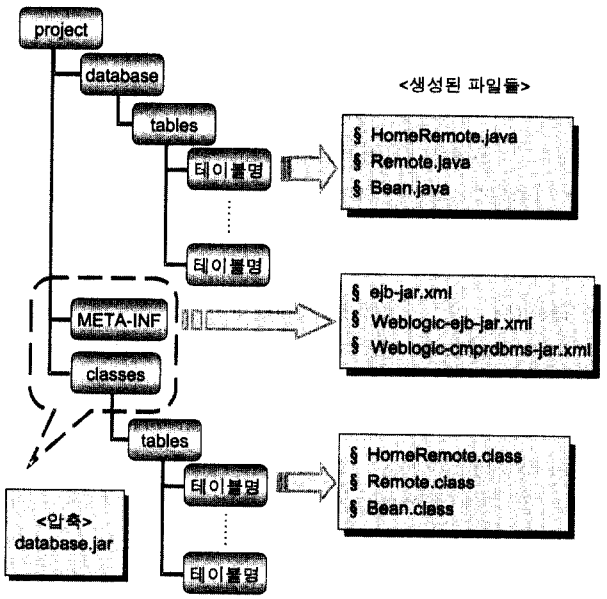
- **WebGen2** : 이 클래스는 WebSiteGen2의 MainFrame을 생성하는 메인 메소드를 포함한다. 데이터베이스부터 구축할 것인지 아니면 기존 데이터베이스를 사용할 것인지에 따라 WebEditorFrame 객체가 생성되거나 MainFrame 객체가 생성된다.
- **Entity** : 데이터베이스의 시스템 카탈로그 테이블에 있는 스키마 정보를 테이블 단위로 저장한다. entityName, primaryKey, foreignKey, foreignKey_table, atts, appendEntities, expandEntities 등을 속성으로 갖는다.
- **WebEditorFrame** : 이 클래스는 JFrame의 서브클래스이다. Entity, EntityPropertyDlg, ConstraintsDlg, CreateDB의 네 개의 컴포넌트들을 포함한다. 클래스 다이어그램을 그릴 수 있는 JPanel 클래스의 drawingPanel 객체를 생성한다. JPanel에 ActionListener를 이용하여 Entity와 Relation을 그리고 각 Entity의 Property 및 무결성 제약조건 등의 정보를 입력받기 위해 Entity, Relationship, EntityPropertyDlg, ConstraintsDlg 등의 객체를 사용한다.
- **CreateDB** : 이 클래스는 JFrame의 서브클래스이다. MS-SQL 7.0이나 ORACLE8i 데이터베이스 중 하나를 선택하게 하고, DbConnDlg 컴포넌트로부터 구축할 데이터베이스의 정보를 입력받은 후, 클래스다이어그램을 그릴 때 생성된 Entity, Relationship, ConstraintDlg 등의 객체를 이용하여 데이터베이스를 구축한다.
- **DbConnDlg** : 이 클래스는 JDialog의 서브클래스이다. 구축할 데이터베이스에 대한 IP 주소, JDBC 포트, 데이터베이스의 사용자 아이디, 패스워드 정보를 입력받는다.
- **MainFrame** : 이 클래스는 JFrame의 서브클래스이다. 기존의 데이터베이스를 이용하여 웹 응용을 생성하기 위한 MainMenuBar, MainToolBar, MainCanvas, Status Field 등의 네 개의 컴포넌트들을 포함한다.
- **MainMenuBar** : 이 클래스는 MenuBar의 서브클래스이다. 메뉴 항목으로는 File, Select, DataBase, GenForm, Help를 포함한다. File 메뉴는 Open, Save, Exit 명령을 갖는다. Select 메뉴는 Class Diagram과 Database 항목이 있으며 데이터베이스부터 구축할 것인지 또는 기존 데이터베이스를 사용할 것인지에 따라 사용자가 필요한 선택을 할 수 있도록 한다. DataBase 항목은 OpenDB와 Close DB 항목을 갖는다. GenForm 항목은 GenEJB, GenForm 항목을 갖는다.
- **MainToolBar** : 이 클래스는 JFrame의 서브클래스이다. AnchorCombo와 MainButtons의 두 컴포넌트들을 포함한다. 이 컴포넌트들은 MainToolBar의 JPanel에 의해 붙여지며 MainMenuBar와 같은 명령을 수행할 아이콘들로 구성된다.
- **OpenDB** : 이 클래스는 JFrame의 서브클래스이다. 데이터베이스를 선택하게 하고, DbConnDlg 컴포넌트에서 입력 받은 정보들을 이용하여 데이터베이스에 연결한다. 데이터베이스의 시스템 카탈로그 테이블에 있는 스키마 정보는 테이블 단위로 Entity 객체들에 저장된다. Entity 객체에는 테이블명, 각 테이블의 속성들, 그 속성의 타입들, 기본 키, 외래 키가 저장되며 또한 Statement 객체의 executeQuery() 메소드를 사용하여 ALL_CONSTRAINS 테이블로부터 검색된 각 테이블과 첨부 및 확장 관계에 있는 테이블명들이 저장된다.
- **AnchorCombo** : 데이터베이스와 연결 후, 콤보박스에 테이블명이 설정된다. OpenDB 객체에서 executeQuery() 메소드를 사용하여 USER_TABLES 테이블로부터 검색된 테이블명은 AnchorCombo 객체의 JComboBox에 첨부된다. 첨부된 테이블명의 리스트에서 사용자가 닻 개체를 선택하게 된다.
- **TreeDlg** : 이 클래스는 JDialog의 서브클래스이다. (그림 4)의 TreeDlg와 같은 Dialog 박스를 생성한다. TreeDlg는 닻 개체의 이름을 보여주는 테이블을 포함한다. OpenDB에서 생성된 Entity 객체들의 getAppendEntities ()와 getExpandEntities() 메소드들을 이용하여 닻 개체와 첨부 및 확장 관계에 있는 테이블들을 트리 구조로 나타낸다.
- **ColumnDlg** : 이 클래스는 JDialog의 서브클래스이다. (그림 4)의 ColumnDlg와 같은 Dialog 박스를 생성한다. ColumnDlg에는 TreeDlg에서 선택된 테이블의 속성들을 나타낸다. 사용자는 폼에 포함시킬 속성들을 선택할 수 있다.
- **EntityGen** : 사용자가 EJBGen 버튼을 클릭 함으로써 이 클래스의 인스턴스가 생성된다. EntityLocalGen, EntityHomeLocalGen, EntityBeanGen 컴포넌트 등의 해당 엔티

티 빈들을 생성한다.

- **SessionGen** : 사용자가 EJBGen 버튼을 클릭 함으로써 이 클래스의 인스턴스가 생성된다. InsertBeanGen, SelectBeanGen, UpdateBeanGen, DeleteBeanGen 컴포넌트 등의 해당 세션 빈들을 생성한다.
- **WebComponentGen** : 사용자가 WebGen 버튼을 클릭 함으로써 이 클래스의 인스턴스가 생성된다. Insert/Search 폼, Select 폼, Update/Delete 폼, Result 폼 등의 네 개를 자동 생성한다. WebSiteGen2가 생성한 EJB 컴포넌트들은 이 HTML 폼들을 통해 이루어질 질의를 처리한다.

4.2 EJB 컴포넌트의 자동 생성

(그림 9)는 WebSiteGen2가 EJB 컴포넌트들을 위해 생성하는 파일들과 이 파일들이 저장될 폴더의 구조이다. 생성되는 EJB 컴포넌트들은 버전 2.0인 엔티티와 세션 빈들이다. EJB 컴포넌트들을 상용 제품인 WebLogic 서버 6.0에 배치하기 위한 디플로이먼트 디스크립터(deployment descriptor)도 생성한다. 하나의 EJB 컴포넌트인 엔티티 빈 또는 세션 빈에 대해 두개의 인터페이스와 하나의 빈 클래스를 생성한다. 각 개체 타입의 개수만큼 엔티티 빈이 생성되며 세션 빈으로는 데이터베이스에 대한 질의처리를 수행하기 위한 InsertBean, SelectBean, UpdateBean, DeleteBean을 생성한다.



(그림 9) WebSiteGen2가 생성하는 EJB 파일들

WebSiteGen2가 EJB 컴포넌트들을 위해 생성하는 파일들은 project/database/beans 폴더 밑에 저장된다. Titan 응용인 경우, beans 폴더 밑에 customer, credit_card, address, phone, ship, cruise, reservation, cabin 등의 서브폴더가 생성되고 각 폴더에 해당 엔티티 빈에 대한 두개의 인터페이스와 하나의 빈 클래스인 HomeRemote.java, Remote.java, Bean.java 파일들이 생성되어 저장된다. 생성된 엔티티 빈들은 Titan 응용 데이터베이스 내의 모든 개체 타입에 대한 데이터 로직을 담당한다. Customer를 닻 개체로 선택하였을 때, 세션 빈들에 대해서는 beans 폴더 밑에 Customer Insert, CustomerSelect, CustomerUpdate, CustomerDelete 등의 서브폴더가 생성되고 각 폴더에 Bean.java, Home Remote.java, Remote.java 파일들이 생성되어 저장된다. 세션 빈은 데이터베이스에 대한 질의 처리를 수행하기 위한 비즈니스 로직을 담당한다.

이렇게 생성된 EJB 컴포넌트들을 상용 제품인 Web Logic 서버 6.0에 배치하기 위한 디플로이먼트 디스크립터도 project 폴더 밑의 META-INF 서브폴더에 생성된다. 생성된 빈들은 컴파일 되어 디플로이먼트 디스크립터에 의해 디플로이 되면서 분산 객체 간의 통신을 위한 스텁(stub)과 스켈레톤(skeleton)이 생성된다. 컴파일 된 클래스 파일, 디플로이먼트 디스크립터, 스텁, 그리고 스켈레톤은 "database.jar"라는 파일로 압축된다. 압축된 파일은 project 폴더 밑에 저장되며 바로 WebLogic 서버 6.0에서 사용이 가능하다.

4.3 JSP 컴포넌트의 자동 생성

WebSiteGen2는 HTML 폼으로부터의 질의 처리를 위해, 생성되는 폼의 구조에 맞게 Insert.jsp, Select.jsp, Update Delete.jsp, Result.jsp의 4가지의 JSP 웹 컴포넌트들을 생성한다. 이들은 세션 빈들과 엔티티 빈들을 이용해 질의를 처리한다. 예를 들어, 데이터베이스에 데이터를 삽입하기 위한 질의를 처리하는 Insert.jsp는 세션 빈인 InsertBean을 호출하여 질의를 처리하고 처리된 결과를 HTML 폼으로 전송하여 화면에 보일 수 있도록 한다.

(그림 10)은 Titan 응용에서 Customer를 닻 개체로 선택하였을 때, 닻 개체와 첨부 및 확장 관계에 있는 개체 타입들을 세션 빈인 CustomerSelect를 이용해 검색하고 그 결과를 Update/Delete 폼에 보여주기 위해 생성된 JSP 컴포넌트의 일부분이다. 먼저, JNDI를 이용해 CustomerSelect 세션 빈이 위치한 주소를 찾아 홈 오브젝트의 참조를 획득한 후, CustomerSelect 세션 빈의 새로운 홈 인터페이스 객체인 home을 생성한다. 생성된 홈 인터페이스 객체를 이용하여 JSP 컴포넌트가 비즈니스 메소드인 select()를 호출할 수 있도록 하기 위해 EJB 객체인 Select를 생성하고 이 객체에 대한 스텁을 획득한다. JSP 컴포넌트는 닻 개체에 대한 검색을 수행하기 위해 스텁을 통해 CustomerSelect 세션 빈의 select() 메소드를 호출한다. select() 메소드는 닻 개체의 기본 키를 파라미터로 갖는다. 메소드 수행 결과는 벡터 타입으로 받아 HTML 폼을 동적으로 작성 한 후, 사용자 측에 전송되어 보여지게 된다.

```
<%@ page contentType = "text/html ; charset = euc-kr" %>
<%@ page import = "java.sql.*, java.util.*, javax.transaction.*,
                   javax.rmi.*" %>
<%@ page import = "java.text.*" %>
<%@ page import = "database.beans.CustomerSelect.*" %>
<%@ page import = "javax.naming.*, javax.ejb.EJBException,
                   javax.rmi.PortableRemoteObject, java.rmi.
                   RemoteException" %>

<%--Select Data Relation Query--%>
<%
    String select = request.getParameter("select") ; //anchor
    try {
        Context jndiContext = getInitialContext() ;
        Object ref = (CustomerSelectHomeRemote)
        jndiContext.lookup("CustomerSelectHomeRemote") ;
        CustomerSelectHomeRemote home = (CustomerSelectHome
                                          Remote)
                PortableRemoteObject.narrow(ref, CustomerSelect-
                                          HomeRemote.class) ;
        CustomerSelectRemote select = home.create() ;

    Vector selectV = (Vector)select.select(select) ;
        Vector anchorV = (Vector)selectV.elementAt(0) ;
    } catch (Exception e) {
        e.printStackTrace();
    }
%>

    :
    :
```

(그림 10) CUSTOMER_UpdateDelete.jsp

5. 결론 및 향후 연구

본 논문에서는 클래스 다이어그램으로부터 새로 구축한 데이터베이스나 기존 데이터베이스를 이용하여 웹 기반의 데이터베이스 응용을 자동으로 생성하는 방법을 제안하고 이 작업을 자동 수행하는 프로토타입인 WebSiteGen2를 소개하였다. WebSiteGen2는 데이터베이스부터 구축해야 하는 응용과 기존 데이터베이스를 사용하는 응용을 위해 사용자 인터페이스로 이용될 HTML 폼들과 이 폼을 통해 이루어질 질의처리를 위한 EJB 및 JSP 컴포넌트들을 자동으로 생성한다. 따라서 성능 면에서 효율적인 3-계층 구조를 가지는 응용의 구현을 자동화함으로써 웹 응용의 생산성을 향상시킨다. 또한, 컴포넌트 기술을 사용함으로써 생성되는 응용의 재사용성, 확장성 및 이식성을 높일 수 있다. 무엇보다도, WebSiteGen2가 생성하는 폼들은 질의의 대상인 개체뿐 아니라 이와 연관된 모든 개체들을 한 폼에 제공하여 조작할 수 있게 한다. <표 1>과 같이 상용 제품들과의 기능 비교를 통해 WebSiteGen2의 장점을 보였다.

향후, WebSiteGen2가 무선 인터넷 응용도 자동 생성하도록 하여 지리정보 시스템에 적용할 예정이다.

참 고 문 헌

[1] Doohun Eum and Toshimi Minoura, "Web-Based Database Application Generator," IEICE Trans. on Information & Systems, Vol.E86-D, No.6, pp.1001-1010, Jun., 2003.

[2] Yongzhen Ou, "On Mapping Between UML and Entity-Relationship Model," In Proceedings of the UML Workshop, pp.45-57, 1997.

[3] Pope and Mike, "Microsoft ASP.NET Web Matrix Starter Kit," Microsoft, 2003.

[4] http://www.codecharge.com/products/product.php?product_id = 1.

[5] WEB DB GENERATOR : http://www.solpa.com.

[6] Richard Monson-Haefel, Enterprise JavaBeans, O'REILLY, 2001.

[7] 차명훈 외 1, "모든 레코드 삭제 연산을 위한 효율적 처리기법의 설계와 구현", SIGDBKDBC-KISS, Vol.18, No.2, 2002.

[8] MSSQL : MS SQL 2000 Books Online, http://ddart.net/mssql/sql2000/html/.

[9] C. J. Date, "An Introduction to Database Systems," 8th Ed., Addison-Wesley, 2003.

[10] Grady Booch, "James Rumbaugh, and Ivar Jacobson," The Unified Modeling Language User Guide, Addison-Wesley, 1998.

[11] Rational Group, UML Notation Guide, Version 1.1, 1997.

[12] Ramez Elmasri and Shamkanth B. Navathe, "Fundamentals of Database Systems," 2nd Ed., Benjamin/Cummings, 1994.

[13] Angel R. Puerta, Henrik Eriksson, John H. Gennar and Mark A. Musen, "Model-Based Automated Generation of User Interfaces," In Proceedings of the 12th National Conference on Artificial Intelligence, Vol.1, pp.471-477, 1994.

[14] Michel E. Adiba and Christine Collect, "Management of Complex Objects as Dynamic Forms," In Proceeding of the 14th International Conference on Very Large Data Bases, pp.134-147, 1988.

[15] 정화영 외 1, "Implementation and Development of Seat Reservation System based on EJB for E-Business", 정보처리학회 2002년 추계학술대회, Vol.9, No.2, 2002.

[16] Grady Booch, James Rumbaugh and Lvar Jacobson, "The Unified Modeling Language User Guide," Addision-Wesley, 1998.

음 두 헌

e-mail : dheum@duksung.ac.kr

1984년 서강대학교 전자공학과(학사)

1987년 오레곤 주립대학교 컴퓨터 공학과(석사)

1990년 오레곤 주립대학교 컴퓨터 공학과(박사)

1991년 전자통신연구원 인공지능연구실

1999년~2000년 오레곤 주립대학교 전산학과 파견교수

1992년~현재 덕성여자대학교 컴퓨터과학부 교수

관심분야 : 객체지향 시스템, 컴포넌트 기반 시스템

고 민 정

e-mail : mjko@duksung.ac.kr
2002년 덕성여자대학교 전산학과(학사)
2002년~현재 덕성여자대학교 대학원
전산·정보통신학과 재학 중
관심분야 : 객체지향 시스템, 컴포넌트 기반 시스템, 모바일 시스템

강 이 지

e-mail : izzy@duksung.ac.kr
2003년 덕성여자대학교 전산학과(학사)
2003년~현재 덕성여자대학교 대학원
전산·정보통신학과 재학 중
관심분야 : 객체지향 시스템, 컴포넌트 기반 시스템, 모바일 시스템