XML 폼 구조를 기반으로 하는 리포트 생성기의 설계 및 구현 : MoonLight

황 기 태†

요 약

본 논문은 데이터베이스나 기타 데이타 소스로부터 출력할 리포트를 설계하고 프린트하는 리포트 생성기의 기능을 가진 MoonLight 시스템을 설계 구현한 내용을 논한다. 리포트 시스템을 모델링하고 MoonLight의 구성, 동작 과정을 정의하였으며 리포트를 구성하는 구성 요소와 리포트 폼을 작성하는 알고리즘 및 리포트를 프린트하는 알고리즘의 구현 내용을 기술하였다. 리포트의 템플릿 폼은 XML을 이용하여 정의하였으며 리포터 생성기는 플랫폼에 종속되지 않는 자바 언어로 작성되었다. 또한 본 논문은 페이지 객체 구성 시간과 프린트 이미지의 렌더링 시간을 측정하여 구현된 MoonLight의 실행 성능을 평가하였다.

Design and Implementation of Report Generator based on XML Form : MoonLight

Kitae Hwang†

ABSTRACT

This paper presents details of the design and implementation of a report generator, MoonLight, which supports to design the report form from DB or data sources and print it to the printer. Also this paper defines the detailed model of the report system, the architecture and behavior of MoonLight, and the components of the report form, and also shows algorithms to make the report template and print the report bound to the data source. MoonLight takes advantage of XML language for the report template form. And also it is implemented by Java language to take the advantage of platform independency. This paper also shows the results of run-time performance of MoonLight which is measured with the time to construct page objects of a report and the time to render the page objects to the print images, respectively,

키워드 : 리포트 생성기(Report Generator), 리포트 폼(Report Form), XML, 데이타베이스 응용(Database Application)

1. 서 론

리포트 생성기(Report Generator)란 리포트 폼을 정의하고 데이타베이스 등과 같은 데이타 소스로부터 사용자가 필요로 하는 데이타를 추출하여 리포트를 출력하는 소프트웨어이다[10, 11, 13, 14]. 예를 들어 보험 회사의 경우 60세 이상의 가입자들에서 A4 한 장 분량의 우편물을 발송하고자 하면 리포트 생성기를 이용할 수 있다. A4 크기의 화면에 출력하고자 하는 문장과 받는 사람의 이름 부분을 생성하고, 이름 부분에는 데이타베이스에서 가입자 테이블의 이름 필드를 연결하는 방식으로 리포트 폼을 정의한 뒤, 이름 부분에 60세 이상이라는 조건을 두면, 60세 이상의 가입자 수만큼 A4 용지가 출력되어 나오게 된다.

리포트 생성기는 기업에서 매우 필요한 소프트웨어로서 전혀 새로운 것이 아니며 오래 전부터 많은 연구 개발이 이루어져왔다. 최근에 와서 그래픽을 기반으로 하는 사용자 인터페이스에 대한 개발이 급속하게 진행되고 정보 전송의 기반으로 웹이 널리 사용되면서, 보다 화려하고 다양한 사용자 인터페이스를 가지며 웹 환경에서 사용 가능하고 표준화된 리포트 폼을 가지는 등 리포트 생성기에 대한 요구가 새롭게 대두되어 왔다. 현재 OZ[10], MS-ACCESS, Elixir[11], AI Report[13], RReport Visual Builder[14] 등과 같은 리포트 생성 소프트웨어들이 개발되어 있으며, 국내에서는 OZ가 많은 부분을 차지하고 있다. 그러나 지금까지 대부분의 리포트 생성기는 웹 프린팅 기능을 지원하지 않으며, 리포트를 구성하는 단위인 스트라이프(제 3장 참조)를 독립적으로 편집하는 기능이 없는 단점이 있었다. 또한 리포트의 폼 포맷도 리포트 생성기에 종속적인 단점이 있었다. 본 연구에서는 리포트 폼을 정의하는 도구로 XML[2, 3, 16]을 사용하고 자바[1, 4, 5, 9]의 풍부한 사용자 인터페이스를 이용하여 웹에서 사용할 수 있는 리포트 생성기인 MoonLight을 설계 구현한 내용을 논하고자 한다.

리포트의 템플릿 역할을 하는 리포트 폼은 기업의 리포

※ 본 연구는 2003년도 한성대학교 교내연구비 지원과제임.
† 정 회 원 : 한성대학교 컴퓨터시스템공학부 교수
논문접수 : 2002년 11월 25일, 심사완료 : 2003년 3월 18일

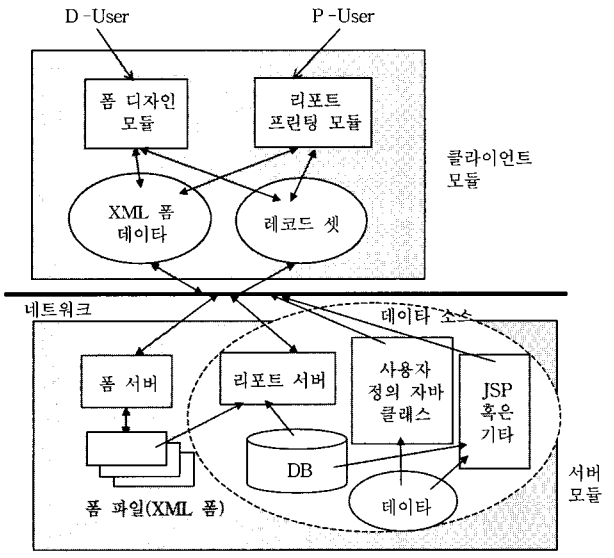
트 디자이너에 의해 한 번 생성되면 오랜 동안 사용되며 여러 부서에서 사용되어야 하는 특성이 있기 때문에 다루기 쉬운 편리성과 미래에 대한 확장성, 표준성 등의 성격을 지니는 것이 바람직하다. MoonLight은 XML을 리포트 폼의 포맷으로 이용한다[7, 8]. 한편 자바는 다양한 사용자 인터페이스를 지원하며, 플랫폼 독립적인 언어로서 한번 작성하여 어떤 플랫폼이나 하드웨어 환경에서도 사용할 수 있는 장점을 가지고 있다[4]. 자바는 또한 특별히 웹 기반의 소프트웨어를 작성하는데 적합한 프로그래밍 언어이다[9]. 이러한 이유로 본 연구에서는 자바를 개발 언어로 선택하였다. 자바의 가비지 켈렉션[1] 지연 시간은, 리포트 생성기가 실시간 응용이 아니기 때문에 문제가 없다고 판단된다. XML을 이용하는 응용들은 대부분 XML 폼을 생성하는 에디터나 XML 폼의 내용물을 출력하는 전용의 브라우저를 작성한다. MoonLight은 이러한 전형적인 XML 응용에 필수적인 기술을 거의 모두 담고 있으며, XML과 데이타베이스, 그리고 자바가 결합된 형태의 응용 프로그램을 개발하는 좋은 모델이 될 수 있다.

본 논문은 다음과 같이 구성된다. 2장에서는 MoonLight을 기준으로 리포팅 시스템의 아키텍쳐를 모델링하고 3장에서는 MoonLight를 구현한 내용을 보인다. 4장에서는 MoonLight의 성능 평가 결과를 보이며 5장에서 결론을 맺는다.

2. 리포팅 시스템 아키텍쳐

2.1 시스템 구성

리포팅 시스템은 (그림 1)과 같이 데이타 소스와 폼 파일을 포함하는 서버 모듈과, 리포트 폼을 디자인하고 데이타 소스로부터 프린트할 데이타를 받아 클라이언트 컴퓨터 상에서 프린트하는 클라이언트 모듈로 구성된다. 본 논문에서



(그림 1) 리포트 시스템 구성

리포트의 레코드 셋을 제공하는 모듈을 데이타 소스라고 명명한다. 데이타 소스는 데이타베이스를 액세스하는 리포트 서버, 사용자가 특별히 저장한 데이타 포맷을 액세스하는 사용자 정의 자바 클래스, 혹은 웹 서버에 연결되어 레코드 셋을 제공하는 JSP(Java Server Page)[6]나 혹은 그 외 리포트 프린팅 모듈에서 정의한 템플릿 포맷을 지원하는 기타 방법들을 포함한다. 논문의 전개상 단순화를 위해 데이타 소스 중에서 데이타베이스 만을 대상으로 설명한다.

리포트 사용자는 두 부류로 구분된다. 그 중 한 부류는 리포트 폼을 설계하는 디자이너(D-User)이며 다른 하나는 리포트 폼을 이용하여 리포트를 프린트하는 사용자(P-User)이다. 리포팅 시스템은 다음과 같이 동작한다.

2.1.1 폼 디자인

D-User는 폼 디자인 모듈을 이용하여 리포트 폼을 작성한다. 작성된 폼은 XML로 표현되며 리포트 서버에 의해 저장된다.

2.1.2 프린트할 리포트 선택

P-User는 웹 브라우저를 이용하여 리포터 서버에 접속하고 원하는 리포트(혹은 리포트 폼)를 선택하여 리포트 서버로 요청한다.

2.1.3 리포트 폼과 레코드 셋의 전송

리포트 폼에는 프린트할 데이타베이스와 레코드 셋에 관한 정보가 들어 있으므로 리포트 서버는 리포트 폼을 읽고 해석한 다음, 필요한 레코드 셋을 데이타베이스로부터 읽어서 리포트 폼과 함께 클라이언트로 전송한다.

2.1.4 프린트

클라이언트는 리포트 폼과 레코드 셋을 받고, 리포트 폼을 해석하여 프린트 이미지를 만들고 프린터로 출력한다.

2.2 리포트 시스템 설계

2.2.1 리포트 폼과 리포트 레코드 셋

리포트 시스템이 다루는 컨텐트는 두 가지이다. 하나는 리포트 폼이고 나머지는 레코드 셋이다. 리포트 폼은 단순한 형태에서 매우 복잡한 형태까지 다양하다. 이름과 전화번호로만 구성되는 전화번호부를 출력하기 위한 리포트 폼의 경우는 리포트 폼에 표시되는 내용이 단순하고 작기 때문에 크기는 몇 KB 미만이다. 그러나 세금 계산서와 같은 경우에는 리포트 폼이 매우 복잡하며 그 크기가 몇 백 KB 이상이 되기로 한다.

전자의 경우는 리포트 폼이 단순한 반면 연결되는 레코드 셋의 크기는 매우 크며 출력되는 페이지 수 또한 많은 것이 일반적이다. 그러나 후자의 경우 보통, 출력되는 리포트는 한 두 페이지 정도이며 따라서 레코드 데이타의 크기보다 리포트 폼의 크기가 상대적으로 매우 클 수 있다. 따

라서 전자의 경우는 데이타베이스에 레코드 셋을 액세스하고 이를 프린트하는 시간이 중요하며, 후자의 경우는 리포트 폼을 로드하는데 걸리는 시간이 중요하다.

2.2.2 폼 레이블(Form Label)

본 연구에서는 리포트에 출력되는 내용 요소들을 구성하는 텍스트, 이미지, 표 등을 폼 레이블 이라고 부른다. 폼 레이블는 리포트를 작성하는데 필요한 일종의 메뉴이며 폼 레이블이 다양하고 기능이 풍부할수록 다양한 모양과 내용의 리포트를 작성할 수 있다. MoonLight은 구체적으로 <표 1>과 같은 폼 레이블을 가진다.

폼 레이블들은 데이타베이스의 레코드 필드에 매핑되는 것과 그렇지 않은 것으로 구분된다. 예를 들어 DB 레이블은 레코드셋의 한 필드를 반영하므로, 리포트가 출력될 때 레코드 수 만큼 용지 상의 DB 레이블 위치에 반복적으로 해당 필드 값을 프린트하여야 한다. 차트 레이블은 레코드셋의 데이타를 이용하여 차트를 그리기 위한 레이블이다. 파라미터 레이블은 P-User에 의해 리포트가 프린트되는 시점에서 P-User로부터 입력받고자 하고 값을 반영하는 레이블이다. 이들 폼 레이블들이 실제 스트라이프에 배치되어 있는 예는 (그림 4)와 (그림 8)에서 볼 수 있다.

2.2.3 리포트의 구성과 리포트 폼

일반적으로 문서 편집기에 의해 편집되는 문서들은 페이지 헤더, 페이지 풋터, 본문 등으로 구성된다. 리포트 생성기들은 리포트 혹은 리포트 폼을 작성함에 있어 이들을 포함하여 보다 다양한 구성 요소를 가진다. 본 연구에서는 이 구성 요소를 스트라이프(stripe)라고 부른다. 스트라이프의 종류로는 페이지 헤더 스트라이프(page header stripe)와 페이지 풋터 스트라이프(page footer stripe), 카버 헤더 스트라이프(cover header stripe)와 카버 풋터 스트라이프(cover footer stripe), 그룹 헤더 스트라이프(group header stripe)와 그룹 풋터 스트라이프(group footer stripe), 데이타 스트라이프 셋, 즉 데이타 헤더(data header stripe), 데이타 풋터(data footer stripe), 바디 스트라이프(data body stripe) 등 9개를 정의하였다. 스트라이프에는 텍스트, 이미지, 라인, 원, 표, 날짜 및 시간, 페이지 번호 등 <표 1>에 있는 폼 레이블들이 삽입된다.

리포트 폼은 리포트를 구성하는 각 스트라이프에 대한 정보를 가진 일종의 템플릿이다. 하나의 리포트 폼에는 페이지 헤드 스트라이프, 페이지 풋터 스트라이프, 카버 헤더 스트라이프, 카버 풋터 스트라이프는 항상 한 개만 존재할 수 있으며, 프린트시 페이지 헤더와 페이지 풋터 스트라이프는 모든 페이지에 적용되며, 카버 헤더 스트라이프는 첫 페이지에 처음에 출력되며 카버 풋터 스트라이프는 마지막 페이지의 가장 끝에 출력된다. 스트라이프의 크기에 있어 폭은 출력할 용지에서 왼쪽과 오른쪽 여백을 제외한 크기를 의미하며, 높이는 용지에서 출력되는 높이를 의미한다. 그러므로 모든 스트라이프의 폭은 동일하며 높이는 서로 다르다. 모든 스트라이프 들이 폼에 선택적으로 존재할 수 있고 존재하지 않을 수도 있다. 데이타 스트라이프 셋과 그룹 헤더, 그룹 풋터 스트라이프는 프린트 하고자 하는 리포트에 따라 선택적으로 사용된다.

가. 단순 리포트

세금 계산서, 계약서, 편지 등과 같이 하나의 리포트가 거의 종이 한 장 정도 크기를 가지는 것으로 프린트되는 내용은 매우 복잡하지만 폼 구조는 단순하다. 이런 경우 대체로 페이지 헤더와 페이지 풋터 스프라이프에는 날짜 정도의 정보를 삽입하고, 카버 헤더 스트라이프에 텍스트, 표, 이미지, 라인 등 복잡한 내용들을 세금 계산서나 계약서 등에 표현하게 된다. (그림 8)(c)는 세금 계산서를 표현한 폼의 예이다.

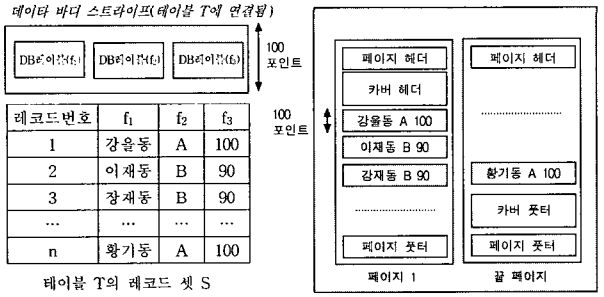
〈표 1〉 폼 레이블

폼 레이블 이름	기 능	레코드셋 연결여부
텍스트 레이블	한 줄짜리의 텍스트를 가지는 레이블	×
스타일텍스트 레이블	여러 줄로 구성되며 색깔 및 크기가 글자단위로 조절되는 레이블	×
페이지번호 레이블	전체 페이지 수, 현재 페이지의 번호를 가지는 레이블	×
날짜 레이블	프린트 날의 날짜와 시간 값을 가지는 레이블	×
이미지 레이블	이미지를 출력하는 사용되는 레이블	×
DB 레이블	데이타 소스의 레코드의 한 필드를 연결하는 레이블	○
수식 레이블	레코드 필드와 텍스트, 수식, 조건문 등의 표현을 입력받아 프린트 시에 계산하고 결과를 출력하는 레이블	○
파라미터 레이블	프린트할 때 사용자로부터 값을 입력받기 위한 레이블	
차트 레이블	레코드 셋으로부터 차트를 그리기 위한 레이블	○
테이블 레이블	레코드 셋으로부터 표를 그리기 위한 레이블	○
크로스탭 레이블	레코드 셋으로부터 크로스탭을 생성하기 위한 레이블	○
선 레이블	선을 긋기 위한 레이블	×
사각형 레이블	사각형을 출력하기 위한 레이블	×
둥근사각형 레이블	둥근 모서리를 가진 사각형을 출력하기 위한 레이블	×
원 레이블	원, 타원을 출력하기 위한 레이블	×

나. 데이타 스트라이프 셋을 가지는 리포트

사원이 1000여명이 되는 규모의 회사 경우, 사원 중에서 1960년 이후에 태어난 사람들의 이름과 주민등록번호를 출력하고자 한다면, 리포트는 아마도 몇 십장 혹은 몇 백장의 분량이 될 것이다. 이런 경우 데이타베이스로부터 1960년 이후의 생년월일을 가지는 사람의 이름과 주민등록번호를 프린트하기 위해 데이타 스트라이프 셋을 이용한다.

데이타베이스에 테이블 T가 있다고 가정하자. 그리고 만일 어떤 데이타 스트라이프 셋이 T와 연결되어 있고 T의 필드들 중에서 f_1(이름), f_2(학점), f_3(점수)의 세 필드로 이루어진 n개 레코드 셋을 S라고 가정하자. 이런 경우 리포트 생성기는 종이 위에 데이타 헤더 스트라이프를 출력하고, 데이타 바디 스트라이프의 높이 간격으로 n번 데이타 바디 스트라이프를 프린트한 후, 데이타 풋터 스트라이프를 출력하게 된다. 즉, i번째 프린트 시에는 n개 중에서 i번째의 레코드 R_i를 액세스하고 R_i의 세 필드를 데이타 바디 스트라이프에 정의된 DB 레이블에 매핑시켜서 프린트한다. (그림 2)는 데이타 헤더와 데이타 풋터 스트라이프가 존재하지 않는다고 가정할 때 데이타 바디 스트라이프와 레코드 셋 그리고 프린트되는 페이지의 관계를 보여 준다. 그림에서 데이타 바디 스트라이프의 높이는 100 포인트이다.



(그림 2) 데이타 바디 스트라이프와 레코드 셋, 그리고 프린트 페이지의 관계

다. 그룹을 가지는 리포트

레코드 셋의 그룹핑이란 어떤 필드들의 값을 기준으로 레코드들을 여러 그룹으로 분할함을 의미한다. (그림 2)의 레코드 셋 S 중에서 f_2를 기준으로 그룹핑 하면 레코드 번호 1, n에 해당하는 두 레코드가 하나의 그룹이 되고, 레코드 2, 3이 또 다른 하나의 그룹이 된다. 의미를 부여하면 같은 학점을 가진 레코드들을 하나의 그룹으로 생성하겠다는 뜻이다. 그룹핑에 사용되는 필드는 한 개 이상 사용가능하다. 만일 f_1, f_2 두 필드를 기준으로 그룹핑 하면, f_1과 f_2의 두 값이 모두 동일한 레코드들이 하나의 그룹을 형성한다.

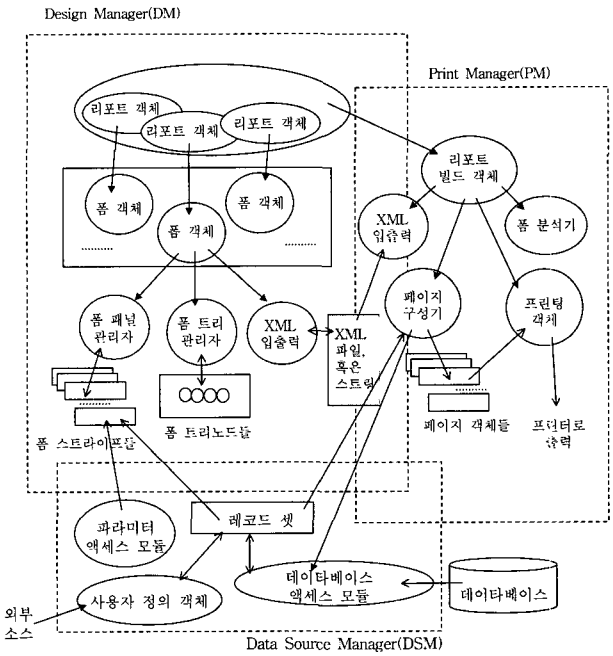
그룹핑 기능을 이용하여 레코드 셋을 여러 그룹으로 분할하여 그룹 별로 용지에 출력할 수 있다. 이를 위해 MoonLight에서는 그룹 헤더 스트라이프에 존재하는 필드들을 그룹핑 기준 필드로 정의한다. 예를 들어 그룹 헤더 스트라이프에 f_2 필드를 가진 DB레이블이 존재하고 데이타 바디 스트라이프에 f_1, f_2, f_3 필드를 각각 가진 세 개의 DB 레이블이 존재하면, S를 f_2를 기준으로 그룹핑하여 여러 그룹의 레코드 셋을 만든다. 그룹의 개수가 g개 라고 하고(현재 (그림 2)의 예에서는 g >= 2), 각 그룹 i의 레코드 셋을 S_i라고 하고 S_i의 개수를 g_i라고 하면, 다음과 같은 순서로 g번 반복하여 그룹 헤더와 풋터 스트라이프들과 데이타 스트라이프 셋을 리포트에 출력하게 된다. 즉 한 번 반복 시마다, 하나의 그룹에 해당하는 레코드 셋을 출력한다. 그룹이 설정되었을 때 프린트하는 알고리즘을 C 언어와 유사하게 기술하면 아래와 같다.

```
for (i = 0 ; i < g ; i++) {
        print group header stripe with the record R0 of gi ;
        print data header stripe with the record R0 of gi ;
        for (j = 0 ; j < gi ; j++)
                print data body stripe with the record Rj of gi ;
        print data header stripe with the record R(gi 1) of gi ;
}
```

3. MoonLight의 구현

3.1 MoonLight의 구성

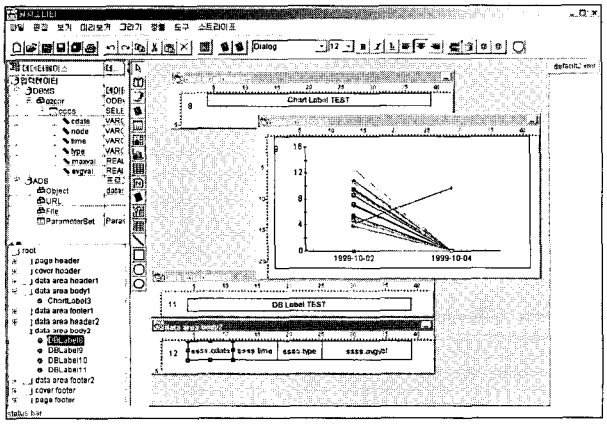
MoonLight은 객체지향언어인 Java로 작성되었으며 JDK 1.3.1 패키지를 이용하였으며 모든 모듈은 객체화 되어 있다. MoonLight은 크게 폼 디자인 관리자(Design Manager, DM)와 프린팅 관리자(Print Manager, PM), 그리고 데이타 소스 관리자(Data Source Manager, DSM)의 세 부분으로 크게 구성되며 (그림 3)과 같다.



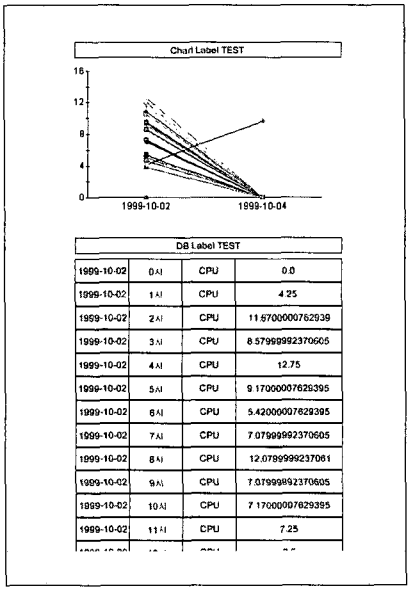
(그림 3) MoonLight의 객체 구성

(그림 4)는 MoonLight을 실행한 모습을 보여준다. 그림에서 볼 수 있듯이 MoonLight은 크게 3개의 윈도우로 구성되며, 각 윈도우는 데이타소스 윈도우, 폼 트리 윈도우, 폼 디자인 윈도우로 불리며 데이타 소스 윈도우는 DSM에 의해 출력된 윈도우이며, 폼 트리 윈도우는 (그림 3)의 폼 트리 관리자에 의해 출력된 윈도우이다. 그리고 폼 디자인 윈도우에는 폼 스트라이프들이 화면에 출력된다.

DM는 사용자가 폼 설계를 위해 D-User에게 그래픽 인터페이스와 폼 레이블을 지원한다. DM는 레이블이 연결하는 데이타베이스와 테이블을 확인하고 액세스 하는 기능과 쿼리를 만드는 기능 등을 수행한다. 임의의 리포트를 프린트할 때 리포트 폼에 정의된 레코드 셋을 읽어와서 프린트될 정보를 생성하게 지원한다. PM는 XML 폼을 읽고 폼에 정의된 레이블들을 이미지화하여 프린트할 페이지 이미지를 만든 다음 프린트로 출력하는 기능을 수행한다.



(a) 리포트를 디자인하는 화면



(b) 리포트의 미리보기

(그림 4) MoonLight의 실행 화면 예

3.2 MoonLight의 동작

3.2.1 리포트 폼 편집

D-User가 리포트를 생성하고자 할 경우 가장 먼저할 일은 리포트 폼을 작성하는 일이다. D-User가 리포트를 작성하기 시작하면 내부적으로 DM는 리포트 객체를 하나 생성한다. 리포트 객체는 다시 폼 객체를 하나 생성하고 폼 객체는 다시 폼 패널 관리자와 폼 트리 관리자를 생성한다. 폼 패널 관리자는 다시 4개의 디폴트 스트라이프들(페이지 헤더, 풋터 스트라이프, 그리고 카버 헤더, 풋터 스트라이프)을 생성한다. 4개의 스트라이프들은 현재 폼 레이블을 전혀 가지고 있지 않은 상태이다. 폼 트리 관리자는 현재 생성된 폼 스트라이프들을 제어하기 위한 폼 트리 노드들을 생성한다. D-User는 (그림 4)(a)에서 중간에 세로로 보이는 레이블 아이콘를 이용하여 레이블을 삽입하여 폼을 편집한다.

3.2.2 리포트 프린트

리포트의 프린트는 폼을 저장하고 있는 XML 스트링으로부터 이를 해석하여 프린트할 페이지 객체들을 생성하는 과정과 페이지 객체들을 렌더링하여 프린트 이미지를 생성하고 이를 프린터로 보내 프린트하는 과정으로 이루어진다.

가. 페이지 객체 생성

리포트를 프린트하기 위해서 우선 PM는 폼을 XML 스트링이나 XML 파일 형태로 입력 받는다. PM는 리포트 빌드 객체를 생성하고 XML 정보를 넘겨주고 프린트를 지시한다. PM가 독립적으로 웹 페이지에 내장되면 웹 서버로부터 XML 파일을 받게 되지만 (그림 3), (그림 4)와 같이 리포트 생성기에 통합된 상태이면 DM로부터 XML 스트링을 넘겨 받는다. 리포트 빌드 객체는 XML 입출력 객체를 생성하여 XML 스트링을 읽고 파싱하여 폼 구조체를 생성하도록 한다. 다시 페이지 구성기를 생성하고 XML 데이타에 들어있는 스트라이프에 관한 정보와 각 스트라이프에 존재하는 폼 레이블들을 해석하여 출력될 페이지 객체들을 생성한다. 페이지 객체들은 페이지의 크기 및 프린트될 폼 레이블들을 가지고 있다.

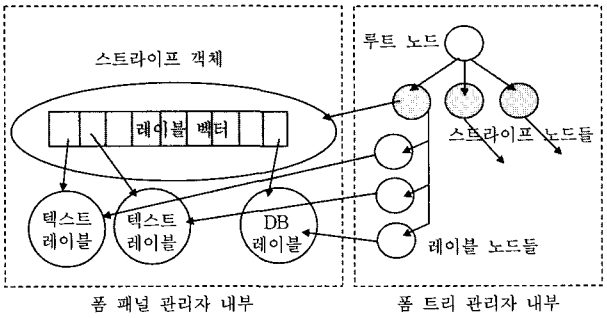
나. 페이지 렌더링 및 프린트

출력될 페이지 객체들이 완료되면, 프린팅 객체가 리포트 빌드 객체에 의해 생성되고 페이지 객체들을 렌더링하여 각 페이지 이미지들을 생성하고 이 이미지를 프린터로 전송하여 프린트한다.

3.3 디자인 관리자(DM)

DM란 폼을 디자인하기 위해 지원되는 소프트웨어 모듈이다. D-User는 여러 개의 리포트 폼를 동시에 편집할 수 있다. (그림 3)에서 볼 수 있듯이 편집되는 리포트 하나 당 하나의 리포트 객체와 폼 객체, 폼 패널 관리자, 폼 트리 관리자, XML 입출력 객체들이 생성된다.

폼은 앞서 설명한 것처럼 스트라이프라는 편집 단위의 집합이라고 볼 수 있다. 스트라이프는 객체 형태로 구현되며 (그림 5)와 같이 레이블들의 리스트를 가진다. 본 논문에서는 이 리스트를 레이블 벡터라고 부른다. 레이블 벡터에 존재하는 각 레이블 객체들은 자신의 위치 및 크기 등의 많은 속성값을 가지고 있다. 스트라이프 당 하나의 스트라이프 노드가 뒤에서 다룰 폼 트리에 존재하며 레이블 객체 하나 당 레이블 노드가 존재한다. 이는 폼 트리를 이용하여 스트라이프 및 레이블을 삭제, 삽입, 속성 변경, 찾기 등을 제어하기 위한 목적이며 지금까지의 다른 리포트 생성기에서는 볼 수 없는 기능이다. 레이블 벡터의 가장 앞에 존재하는 레이블이 z-order 상에서 가장 위쪽에 존재한다. 폼 디자이너가 레이블 들의 z-order 위치를 변경하면 내부적으로 벡터 내의 위치를 변경한다.



(그림 5) 스트라이프 객체와 폼 트리

폼 디자인 패널은 스트라이프를 편집할 수 있는 윈도우를 다중으로 유지하고 관리하는 GUI 객체이다. 기존의 리포트 생성기와 차별화되는 기능으로 각 스트라이프을 독립적으로 편집하고, 위치 변경 및 숨기기 자유롭게 사용할 수 있는 장점을 제공한다.

MoonLight에서 모든 레이블 객체들은 JDK1.3의 swing 패키지의 JComponent를 실질적인 GUI 컴포넌트로 이용한다. 이들은 레이블의 위치, 크기, 보더(border), 텍스트의 폰트 정보, 그외 속성들과 이들의 위치 및 크기를 제어하고 속성을 변경하는 메소드들을 가진다.

3.4 폼 파일의 구조

XML은 다양한 응용에서 컴퓨터 상의 데이터의 교환이나 조직된 데이터의 표현을 위해 사용된다. MathML(Mathmatical Markup Langauge)[18]는 두 컴퓨터 사이의 전송할 수학적 표현을 위해 설계된 경우이며, XForms[17]는 HTML 문서의 폼을 개선한 경우이며. 오디오 및 음성 처리(합성, 변조, 인식 등)를 위한 Voice XML[20], 2차원 그래픽스를 묘사하기 위한 SVG(Scalabel Vector Graphics)[19] 등 많은 XML을 사용한 많은 표준들이 World Wide Consortium(W3C)[15]에 존재한다. 또한 많은 응용 프로그램들이 자신의 데이터 구조를 표현하기 위해 XML을 사용하고 있다. 그러나 리포트 생성기에서 사용되는 리포트 폼을 위해 제안된 XML 표준은 아직 없으며, OZ[10], Elixir[7], JClass[12] 등의 응용 소프트웨어들은 스스로 정의한 XML 태그 집합을 이용하여 자신의 데이터를 구조화하고 있다.

본 논문에서는 리포트 폼을 위해 XML 태그 집합을 정의하였다. <표 2>는 본 논문에서 정의된 리포트 폼을 위한 주요 XML 태그들의 리스트이다. 지면의 한계상 단순 폼의 XML 구성을 예로 하여 본 논문에서 구현한 폼 파일을 구조를 설명한다. (그림 6)은 전형적인 단순 폼 파일의 XML 텍스트이다. XML 파일은 루트 태그로 정의된 <MoonLight> 태그로 시작된다. <DataSet> 태그는 D-User가 정의한 테이블을 지정하는 것으로 (그림 6)의 폼의 경우 2개의 데이타 소스를 가지며, 그 중 하나는 파라미터로서 리포트가 프린트되기 직전에 P-User로부터 파라미터 값들을 얻고 이를 리포트 속에 전달하기 위한 것이다. 다른 하나는 ODBC 데이타

<표 2> 폼 파일을 구성하는 주요 XML 태그

XML 태그	기 능	내포하는 태그
MoonLight	XML 폼의 루트	ReportInput, ReportOutput
ReportInput	리포트에 주어지는 데이타 소스를 정의	DataSource, DataSet
ReportOutput	리포트의 모양, 리포트에 출력될 스트라이프를 정의	PageLayout, ReportForm
DataSource	리포트에 공급되는 데이타의 소스 정의	없음
DataSet	D-User가 정의한 테이블	Column
PageLayout	프린트 용지의 크기 및 여백, 용지방향	없음
ReportForm	D-User가 작성한 스트라이프들 포함	Page
Page	페이지 템플릿 정보	Header, Cover, Footer,
Cover	페이지의 헤더와 풋터를 제외한 페이지 템플릿 정보	Header, DataArea, Footer, Group
Header	헤더 스트라이프 정의	레이블들
Body	바디 스트라이프 정의	레이블들
Footer	풋터 스트라이프 정의	레이블들
DataArea	데이타 스프라이프 셋 정의	Header, Footer, Body
Group	그룹 스트라이프 셋 정의	Header, Body, DataArea, Group

소스로서 ODBC 드라이버를 가진 어떤 데이타베이스로부터 리포트에 레코드 셋을 공급하는 것을 의미하며, 테이블 이름은 mycar이고 이 테이블은 CustomerID, CustomerName, Cellular 라는 세 필드를 가지고 있음을 알 수 있다.

```
<?xml version = "1.0" encoding = "EUC-KR" ?>
<MoonLight>
<ReportInput>
<DataSource dbms = "Parameter">
<DataSource dbms = "ODBC" dsn = "car">
<DataSet name = "mycar">
<![CDATA[
SELECT Customer.*
FROM Customer
]]>
<Column name = "CustomerID" type = "INTEGER"/>
<Column name = "CustomerName" type = "VARCHAR"/>
<Column name = "Cellular" type = "VARCHAR"/>
</DataSet>
</DataSource>
</ReportInput>
<ReportOutput>
<PageLayout papersize = "A4" orientation = "portrait"/>
<ReportForm stripes = "6">
<Page>
<Header name = "page header" height = "72">
<PageNumberLabel name = "pagenum1" type = "Arabia">
<TextLabel name = "text1" fontsize = "12" text = "페이지 헤더입
        니다" x = "80" y = "30" width = "170" height = "30">
</Header>
<Cover>
<Header name = "cover header"/>
<DataArea datasourcename = "car.mycar">
<Header name = "data area header" height = "10"/>
<Body name = "data area body" height = "100"/>
<DBLabel name = "DBLabel1" fieldname = "mycar.CustomerID"
          fontsize = "12" x = "80" y = "10" width = "170"
          height = "30">
<DBLabel name = "DBLabel2" fieldname = "mycar.Cellular"
          fontsize = "12" x = "260" y = "10" width = "170"
          height = "30">
</Body>
<Footer name = "data area footer" height = "10"/>
</DataArea>
<Footer name = "cover footer">
</Footer>
</Cover>
<Footer name = "page footer"/>
</Page>
</ReportForm>
</ReportOutput>
</MoonLight>
```

(그림 6) 전형적인 리포트 폼의 XML 예

<DataArea>에는 datasourcename이라는 속성(attribute)을 있으며 이 속성은 데이타 스트라이프 셋과 연결된 데이타 소스를 지정하는데 이용된다. 본 예에서는 mycar라는 데이타 소스를 지정하고 있기 때문에 데이타 스트라이프에 존재하는 레이블들에서 언급되는 레코드 필드는 모두 mycar의 필드이다. <Header>와 <Footer> 태그는 페이지, 카버, 데이타, 그룹에 대해 동일하게 사용되는 태그이며 각각 헤더와 풋터 스트라이프를 표현한다. <Body> 스트라이프 태그는 오직 데이타 바디 스트라이프를 위한 태그이다. 이제 각 스트라이프는 스트라이프 내에 포함된 레이블을 표현하는 XML 태그들을 서브 태그로 포함한다. 레이블 태그들에는 <TextLabel>, <StyledTextLabel>, <PageNumberLabel>, <DateLabel>, <ImageLabel>, <DBLabel>, <ExpressionLabel>, <ParameterLabel>, <ChartLabel>, <TableLabel>, <CrosstabLabel>, <LineLabel>, <RectLabel>, <RoundRectLabel>, <CircleLabel>가 있다. 이들 태그의 속성(attribute)들은 레이블마다 조금씩 차이가 있지만 레이블이 프린트될 위치와 크기에 관한 정보는 x, y, width, height에 공통적으로 표현된다. <DBLabel>에서 fieldname 속성은 테이블 이름과 필드 이름을 지정하며, 프린트될 때 지정된 필드의 값이 프린트 된다.

3.5 프린팅 관리자(PM)

3.5.1 페이지 객체

MoonLight에서는 페이지 객체는 하나의 페이지에 그려질 레이블 객체들을 가지고 있는 객체이다. 페이지 객체는 프레임이라는 작은 단위로 구성되며[12], 따라서 실제 레이블 객체들은 프레임에 삽입된다. MoonLight에서는 페이지 객체는 페이지 헤더 프레임, 페이지 풋터 프레임, 페이지 바디 프레임의 3프레임을 구성된다. 페이지 헤더 프레임은 페이지 헤더 스트라이프의 내용을 그대로 반영하며 페이지 풋터 프레임은 페이지 풋터 스트라이프의 설계 내용을 그대로 반영한다. 페이지 바디 프레임은 페이지 헤더와 페이지 풋터를 제외한 페이지 내의 영역을 반영하는 것으로 다른 스트라이프들이 출력되는 공간이다. 페이지 객체는 두 개의 타입이 존재한다. 프린트될 한 페이지마다 그 페이지에 출력될 레이블 객체들을 포함하는 실제 페이지 객체와 모든 페이지의 템플릿 역할을 하는 템플릿 페이지 객체로 구분한다. PM는 XML 스트링으로부터 우선적으로 페이지 헤더 스트라이프와 페이지 풋터 스프라이프 태그를 분석하여 페이지 헤더 프레임과 페이지 풋터 프레임을 구성하고, 빈 바디 프레임을 생성하고 이들을 조합하여 템플릿 페이지 객체를 생성한다. 그리고나서 실제 프린트할 페이지를 만들 때마다 템플릿 페이지를 복사하여 실제 페이지 객체를 생성하고, 이 실제 페이지 객체의 바디 프레임에 프린트할 내용을 삽입한다.

3.5.2 프린트 알고리즘

PM의 핵심 객체는 (그림 3)에서 보여진 페이지 구성기와 프린팅 객체이다. XML 파일로부터 리포트 폼에 대한 정보를 읽고 이를 최종적인 프린트 이미지로 만들어 출력하는 기능을 수행한다. 프린트 알고리즘은 C 언어와 유사한 표현으로 (그림 7)과 같이 묘사된다. makePageObjects 알고리

즘은 폼 XML 스트링을 입력으로 받아 이를 해석하고 프린트할 페이지 객체(page object)들을 생성한다. printPages 알고리즘은 프린터를 입력으로 받고 각 객체들을 하나의 프린트 이미지로 렌더링하고 이를 프린터로 출력한다. 각 레이블의 렌더링은 기본적으로 JDK 1.3의 Jcomponent를 렌더링하는 루틴을 호출하는 방식이다.

```
(Algorithm) printReport(XMLString, TargetPrinter)
XMLString : XML string of the form file to be printed ;
TargetPrinter : real printer ;
{
1    makePageObjects (XMLString) ;
2    printPages (TargetPrinter) ;
}

(Algorithm) printPages (TargetPrinter)
TargetPrinter : real printer ;
{
1    for(each page object) {
2        make an image of the paper size ;
3        for (each frame) {
4            render the label objects of the frame into the
             image ;
5        }
6        print the image into TargetPrinter ;
7    }
}

(Algorithm) makePageObjects(XMLString)
XMLString : XML string of the form file to be printed ;
{
1    read XML String and parse it ;
2    open Data Source from <ReportInput> tag and read the
      record set using an extra thread ;
3    create a template page object from <PageLayout> tag ;
4    make a page header frame from <Header> within <Page> ;
5    make a page footer frame from <Footer> within <Page> ;
6    insert the page header frame and page footer frame into
      the template page object ;
7    create a new copy of template page object and set the
      copy page object as the current page ;
8    for (each stripe within XMLString) run the step 9 to
       step 36
9    if (the body frame of current page is too short to contain
       the next stripe) {
10       create a new copy of template page object and set
         the copy page object as the current page ;
11   }
12   switch (the current stripe tag) {
13   case cover header stripe
14   case cover footer stripe
15       create label objects from the label tags and insert
         them into the body frame of the current page
         object ;
16       break ;
17   case data header stripe
18       get the data source name from the <DataArea> tag ;
19       command DSM to construct RS, the record set ;
20       get R0, the first record from DSM ;
21       create label objects from the label tags and insert
         them into the body frame of the current page
         object. If there are some labels related with the
         record fields, the values of the labels are set by the
         field values of the record R0 ; break ;
22   case data body stripe
23       set the number of the records within the RS to
         variable n ;
24       variable i = 0 ;  // record index
25       while (i < n) {
26           if (the available room of the body frame of
                the current page object >
                the height of the current stripe) {
27               get Ri, the ith record from DSM;
28               create the label objects from the label
                 tags and insert them into the body
                 frame of the current page object. If there
                 are some labels related with the record
                 fields, the values of the labels are set by
                 the field values of the record Ri.
29               i = i + 1 ;
30           }
31           else {
32               create a new copy of template page
                 object and set the copy page object as
                 the current page ;
33           }
         } ; break ;
34   case data footer stripe
35       get R(n-1), the last record from DSM ;
36       create label objects from the label tags and insert
         them into the body frame of the current page
         object. If there are some labels related with the
         record fields, the values of the labels are set by the
         field values of the record R(n-1) ;
}
```

(그림 7) 리포트 프린트 알고리즘

4. MoonLight의 실행 및 성능 측정

본 연구에서 개발한 리포트 생성기를 기능적인 면과 성능적인 면에서 평가한다. 기능적인 면에서의 객관적인 평가는 사실상 쉽지 않다. 그러나 지금까지 대부분의 리포트 생성기는 스트라이프들을 독립적으로 편집할 수 없는 불편함이 있었으나 MoonLight에서는 각 스트라이프들이 하나의 독립된 윈도우로 구성되어 독립적으로 편집 가능하기 때문에 매우 편리하다. 또한 MoonLight는 리포트 생성기가 갖추어야 하는 거의 모든 폼 레이블를 가지고 있으므로 다양한 리포트의 작성을 지원하며 폼을 XML로 구현하는 장점을 가진다.

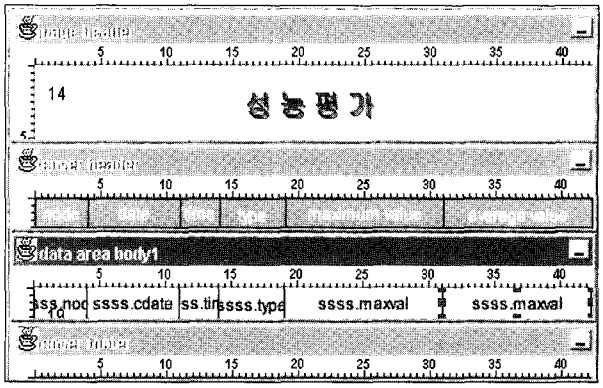
리포트 시스템의 성능은 폼 로드 시간과 리포트의 프린트 시간으로 설정할 수 있다. 폼 로드 시간이란 폼을 읽기 시작한 시간으로부터 화면에 출력하기까지의 시간을 의미하며 이는 폼 구성의 단순도 및 폼 해석의 효율성을 표현한다고 볼 수 있다. 폼 디자이너 혹은 프린팅 사용자는 폼 로드 시간에 대해 매우 민감한 반응을 보인다. 두 번째 지표인 리포트의 프린트 시간은 실질적인 프린트 시간으로 많은 페이지를 프린트하거나 매우 복잡한 폼 구조를 가진 리포트를 프린트하는 경우에 폼 프린트는 예상보다 많은 시간을 차지한다. 특히 웹에서 리포트를 프린트하는 경우

리포트 시스템의 성능은 전적으로 리포트의 프린트 시간으로 평가된다.

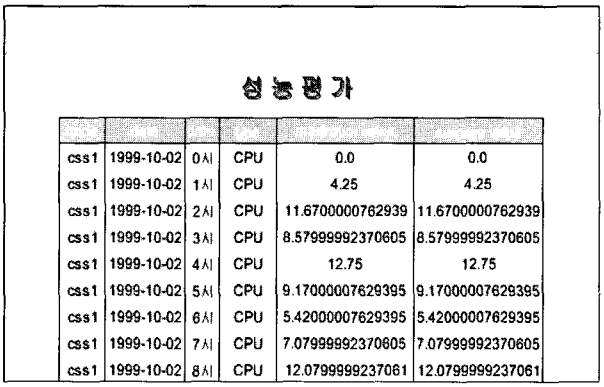
펜티움4, 1.5GHz CPU, 512MB 메모리를 가진 고성능 PC에서 두 가지 리포트 폼에 대해서 성능을 평가하였다. <폼 타입 1>은 4개의 스트라이프를 사용하며 데이타 바디 스트라이프에 6개의 필드를 가진 테이블을 연결하고 6개의 DB 레이블을 삽입하고 각 필드 값을 출력하도록 하였다. <폼 타입 2>는 데이타 소스와 연결되지 않으며 총 330개의 레이블로 이루어진 매우 복잡한 폼이다. 프린트 시간은 실제 프린터에서 소모하는 시간을 배제하고 프린트 명령을 내린 시점에서 프린트 이미지가 화면에 미리 보기로 출력되는 시간으로 측정되었다. (그림 8)(a)와 (그림 8)(b)는 각각 <폼 타입 1>로 사용된 폼 디자인 화면과 폼을 미리보기로 프린트하였을 때의 모습을 보여준다. (그림 8)(c)는 <폼 타입 2>로 사용된 세금계산서의 폼 디자인 화면을 보여 준다. (그림 8)(c)의 미리보기 화면은 거의 비슷하므로 생략하였다. <표 3>은 각 폼 타입에 사용된 XML 파일의 크기와 측정된 폼 로드 시간을 보여 준다. (그림 9)는 <폼 타입 1>의 프린트 시간을 측정한 결과이다. 사용된 레코드의 수를 변화시키면서 페이지 객체들을 생성하는 시간(알고리즘 make PageObject)과 페이지들의 프린트 이미지들을 생성(알고리즘 printPages)하는 시간을 각각 측정하였다.이 평가의 결과가 보여주듯이 당연히 레코드의 수에 비례하여 프린트 시간이 증가하고 있지만, 특이할 점은 페이지 객체의 프린트 이미지를 만드는 렌더링 시간이 대부분의 시간을 차지하고 있다는 점이다. 레이블(JDK에서 JComponent 객체)을 렌더링하는 방법은 완전히 자바의 JDK1.3 패키지에 의존하고 있다. 그러므로 레이블을 렌더링하는 새로운 소스를 작성하지 않는한 이 시간을 궁극적으로 줄이기는 힘들지만, 렌더링에 소요되는 시간을 세분화하여 가장 시간이 많이 걸리는 부분의 로드를 줄일 수 있는 페이지 객체 구조를 연구하여야 할 것이다. 현재 MoonLight은 성능 튜닝이 완전히 끝난 상태가 아니므로 이 문제를 해결하여야 할 것이다. 이 문제를 해결하는 한 가지 방법은 (그림 8)(b)에서 힌트를 얻을 수 있듯이, 각 열은 모양이 모두 같고 출력되는 텍스트만 다르므로, 처음 하나의 레이블에서 텍스트를 제외하고 프린트 이미지를 생성한 다음 모든 레이블에 대해서는 텍스트 이미지를 생성하고 두 이미지를 합친다면 렌더링 시간은 매우 줄어질 수 있을 것이다. 이러한 주제는 본 논문의 범위를 넘어서는 것으로 추후 새로운 연구를 통해 계속되어야 할 것으로 본다.

〈표 3〉 폼 로드 시간

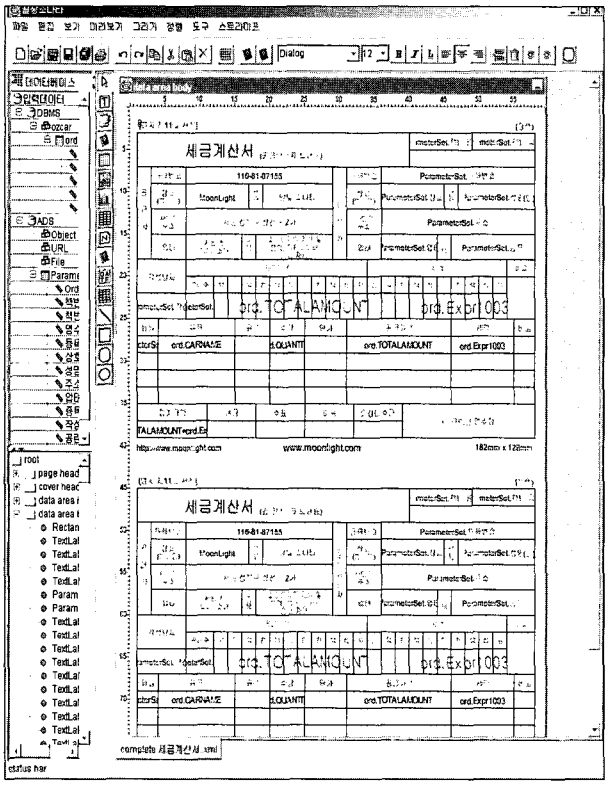
성 능	폼 타입 1	폼 타입 2
파일 크기	9KB	123KB
폼 로드 시간	약 2초	약 7초



(a) <폼 타입 1>의 폼 디자인 화면

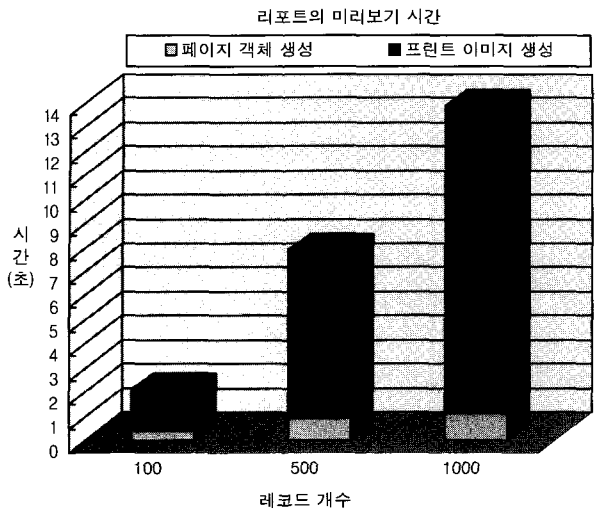


(b) <폼 타입 1>의 첫페이지의 미리보기 화면



(c) <폼 타입 2>의 폼 디자인 화면

(그림 8)



(그림 9) <폼 타입 1>의 미리보기에 걸리는 시간

5. 결 론

본 논문은 리포트 생성기인 MoonLight 시스템을 설계 구현한 내용을 논하였다. 리포트의 템플릿 폼은 XML을 이용하여 정의되었으며, MoonLight은 사용자가 리포트의 템플릿 폼을 정의하고 이 폼에 데이터베이스를 연결하여 리포트를 구성하고 프린터로 출력하는 기능을 제공한다. MoonLight은 객체지향 언어인 자바로 작성되었으므로 플랫폼에 종속되지 않는 장점을 지니고 있으며 리포트를 프린트하는 모듈을 독립적으로 작성하였기 때문에 이 모듈을 HTML 파일에 내장하여 웹 환경에서 리포트를 출력할 수 있다. 본 논문에서는 구현된 MoonLight의 구성, 동작 과정, 그룹핑 알고리즘, 리포트 프린트 알고리즘을 기술하였으며, MoonLight의 XML 폼 포맷을 보였다. MoonLight은 XML과 데이타베이스, 그리고 자바를 이용하여 응용 프로그램을 개발하는 연구에 하나의 좋은 모델이 될 수 있을 것이다. 또한 본 논문에서는 MoonLight의 실행 성능을 보이기 위해 데이타베이스와 연결된 페이지 객체 생성 시간과 리포트의 렌더링에 소모되는 시간을 측정하였다. 성능 측정 결과, 리포트를 프린트 이미지로 렌더링하는 시간이 전체 시간을 지배하였다. MoonLight의 성능을 개선하기 위해서는 프린트 이미지를 렌더링하는 시간을 줄이기 위한 기술적인 연구가 진행되어야 할 것이다.

참 고 문 헌

[1] Bill Venners, "Inside the Java Virtual Machine," McGraw-Hill, 1998.
[2] Paul Spencer, "Professional XML Design and Implementation," Wrox Press, 1999.
[3] William J. Pardi, "XML in Action Web Technology," Microsoft Press, 1999.
[4] Brett Spell, "Professional Java Programming," Wrox Pres, 1999.
[5] David M. Geary, "Graphic Java Mastering the JFC," Swing, Sun Microsystems Press, Vol.2, 1999.
[6] Duane K. Fields and Mark A. Kolb, "Web Development with Java Server Pages," Manning Press, 2000.
[7] 김창수, 정회경, "XML 응용 개발 환경," 한국정보과학회지, 제19권 제1호, pp.15-23, 2001.
[8] 박상원, 정재목, 정태선, 김형주, "XML과 데이타베이스", 한국정보과학회지, 제19권 제1호, pp.24-30, 2001.
[9] Jason Hunter, "Java Servlet Programming", O'Reilly, 2001.
[10] OZ, http://www.forcs.com/oz2.5.htm.
[11] Elixir, http://www.elixirtech.com.
[12] sitraka software, "JClass PageLayout Programmer's Guide," http://www.sitraka.com.
[13] http://www.activeintra.com.
[14] http://www.java4less.com.
[15] World Wide Wep Consortium, http://www.w3.org/.
[16] XML 표준, http://www.w3.org/XML/.
[17] XForms 표준, http://www.w3.org/TR/xforms/.
[18] MathML 표준, http://www.w3.org/Math/.
[19] SVG 표준, http://www.w3.org/Graphics/SVG/.
[20] VoiceXML 표준, http://www.w3.org/Voice/.

황 기 태
e-mail : calafk@hansung.ac.kr
1986년 서울대학교 컴퓨터공학과(학사)
1988년 서울대학교 대학원 컴퓨터공학과 (공학석사)
1994년 서울대학교 대학원 컴퓨터공학과 (공학박사)
2000년~2001년 University of California, Irvine,의 방문 교수
1994년~현재 한성대학교 컴퓨터시스템공학부 부교수
관심분야 : 고성능 입출력 시스템, 인터넷 시스템, 모바일 보안 등