J2ME상에서 kXML Parser를 이용한 MIDlet 응용 설계 및 구현

박 영 수† · 장 덕 철††

요 약

J2ME 기반의 MIDP는 기존 데스크탑 클라이언트에 비해 메모리, 스크린, 그리고 네트워크나 처리속도측면에서 많은 제한점을 가지고 있다. 특히 휴대폰의 경우 각 기기별 스크린 크기가 다르고, 통신사별 서비스 방식이 다르므로, 서버측면에서 보면 이를 위한 세심한 처리과정이 있어야 한다. 이것은 서버의 부하를 유발하는 원인이 될 수밖에 없다. 따라서 본 논문에서는 서버 측에선 XML 문서 형태로 서비스를 하고, J2ME 기반의 클라이언트에선 kXML 파서를 이용한 MIDlet 프로그램을 수행하게 함으로서, 서버의 부하를 줄임과 동시에 J2ME 클라이언트의 단점을 극복하는 방안을 제시하고자 한다.

Design & Implementation of MIDlet Application using kXML Parser on J2ME

Young-Soo Park† · Duk-Chul Jang††

ABSTRACT

Compared with the desktop, the MIDP on J2ME has many limitations on memory capacity, screen size, networking ability and processing speed. In addition, screens vary with device providers in size, and service platforms vary with mobile service providers. These factors require additional functions to server and cause overheads of the server. On this paper, we propose the way to reduce overheads of the server and to overcome disadvantages of the J2ME clients, with the server that provides the service of data in XML document type and the client on MIDP that is developed including the kXML parser.

키워드 : XML, Parser, J2ME, MIDP(Mobile Information Device Profile), MIDlet, Mobile

1. 서 론

최근 인터넷의 급속한 발달로 인해 서버의 역할이 더욱 중요시되고 있다. 또한 서버가 하나의 서비스를 하는데 있어서도 유·무선과 같은 다양한 형태의 클라이언트의 등장으로 인해 서버를 구조적으로 분리하여 분산처리 하거나, 각 클라이언트별로 적합한 언어나 기기사양에 맞는 서비스를 준비하여 제공해야만 한다. 이는 결과적으로 서버의 부하를 증가시키거나 다양한 형태의 서버 측 프로그램을 준비하고 관리해야 하는 부담을 갖게 된다.

따라서 본 논문에서는 간단한 질의어만으로도 서버와 데이터베이스간의 문서가 용이하게 작성될 수 있는 XML(eXtensible Markup Language) 문서를 기반으로 서버를 구성하고, 이를 클라이언트에 맞게 서비스를 제공하려고 한다. 첫째, 기존 데스크탑 형태의 클라이언트는 XSLT(XML Transformations)를 통해 XML과 XSL(eXtensible Stylesheet Language)을 HTML 문서 형태로 서비스를 제공한다[1]. 둘째, J2ME(Java 2 Micro Edition) 기반의 모바일 클라이언트에는 기존의 WAP 프로토콜을 이용한 WML 문서형태가 아닌 HTTP 프로토콜을 이용하여 순수 XML 형태로 서비스를 제공한다[2]. 이럴 경우 모바일 클라이언트 측면에서는 XML 문서를 파싱해야 할 파서를 내장해야 하는 부담을 갖게 되지만, 최근 개발된 모바일 장치를 위한 크기가 작은 XML 파서를 내장함으로서, 서버의 부하를 줄이고 각각의 클라이언트에 맞는 서비스를 제공할 수 있다고 본다.

본 논문의 구성은 2장에서는 관련 연구를 소개하고, 3장에서는 XML 기반의 서버 설계와 J2ME 기반에서 XML 파서를 이용한 모바일용 클라이언트 MIDlet을 설계한다. 4장

† 정 회 원 : 광운대학교 대학원 컴퓨터과학과
†† 정 회 원 : 광운대학교 컴퓨터과학과 교수
논문접수 : 2003년 9월 5일, 심사완료 : 2003년 10월 17일

에서는 3장의 설계를 기반으로 모바일용 XML 클라이언트와 서버를 구현하고, 끝으로 5장에서 결론을 맺는다.

2. 관련 연구

본 장에서는 데이터베이스와 웹 서버간의 연동이 용이한 언어인 XML, 그리고 모바일 클라이언트의 기반이 되는 J2ME와 본 논문에서 이용하게될 클라이언트용 XML Parser에 관해 살펴본다.

2.1 XML

XML은 어떤 데이터 형태든 구조적 방법으로 표현하기에 가장 적합한 언어이다. 또한 HTML에 비해 태그의 활용 면에서 훨씬 자유로우며, 사회 여러 분야에서 사용되는 데이터를 XML 형태로 생성할 수도 있고, 수학, 화학, 그래픽 등 다른 분야로의 확장도 가능하다[3-5]. 그리고 MS-SQL이나 Oracle 등 최근의 데이터베이스들은 간단한 질의어만으로도 쉽게 XML 문서가 생성될 수 있는 방법을 제공하고 있다. 따라서 기존의 서버 측 프로그램 개발에 들였던 노력을 절감할 수 있는 부수적인 이점을 갖게 되었다.

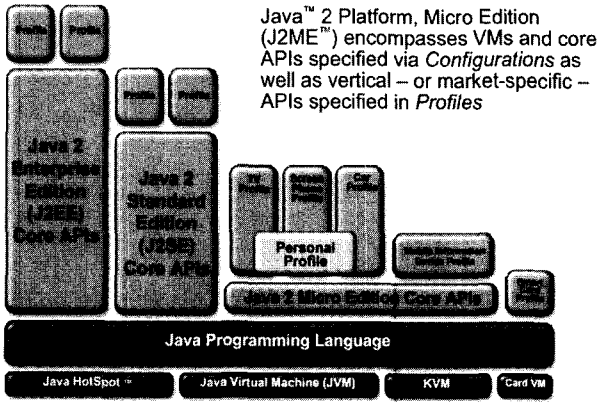
또한, XML은 다른 형태로의 문서 변환이 용이하다[6]. 원시 XML 문서는 스타일 시트인 XSL을 이용하여 변환 프로세서인 XSLT에 의해 XML, HTML, WML 등 다양한 형태로 출력이 가능하기 때문이다. 이 때 XSL이 존재하는 유형은 크게 3가지로 구분할 수 있는데, 첫째 XML 문서에 XSL이 포함된(embedded) 유형, 둘째 XML 문서에 XSL의 경로 참조(referenced)만 갖는 유형, 그리고 마지막으로 XML 문서와 XSL가 서로 독립적으로 분리(unlinked)되어 있는 유형이 있다[1, 6].

이 중 세 번째 경우는 하나의 XML 문서에 여러 개의 XSL을 준비하여 필요시 XSLT에 의해서 다른 형태로 변환이 가능하므로 본 논문에서는 유·무선 클라이언트의 서비스 요청에 따라 동적으로 변환이 가능한 분리된(unlinked) 유형을 선택하였다[1, 2].

2.2 J2ME

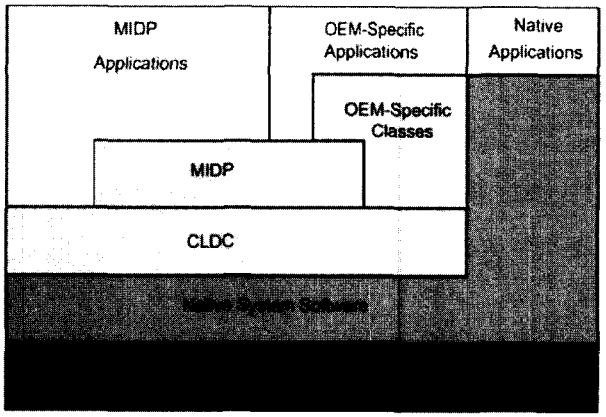
최근 휴대폰 보급의 확대와 함께 무선 인터넷 이용자의 급속한 증가 추세를 보이고 있고, 업계에서는 지속적인 디바이스 개발에 힘입어 조만간 모바일 스테이션 모뎀(MSM)의 계발로 인한 전송속도 및 실행속도의 증가를 기대할 수 있을 것으로 보인다.

모바일 클라이언트의 대표적인 플랫폼을 보면 (그림 1)과 같이 Sun Microsystems사의 J2ME와 Microsoft사의 ME (Mobile Explorer)를 들 수 있다.



(그림 1) J2ME 플랫폼 구조

J2ME 플랫폼은 (그림 2)의 CLDC/MIDP를 기반으로 WML 및 XML 형식의 문서를 지원하며[5, 7-9], WAP 게이트웨이를 통하여 서비스 할 수 있도록 하고 있다[10]. 반면 ME는 OS 커널을 기반으로 한 브라우저 형태로 HTML 및 WML 문서형식을 지원하며, WAP 게이트웨이를 이용하지 않고 HTTP 프로토콜을 이용하여 직접 서버에 접속이 가능하지만, XML 문서 형식을 지원하지는 않는다. 따라서 본 논문에서는 <표 1>과 같이 J2ME 기반의 플랫폼 상에서 HTTP 프로토콜을 사용하고[11-13], <표 2>에서 제시된 kXML 파서를 이용하여 XML 문서를 처리하고자 한다[14, 15].



(그림 2) CLDC/MIDP 구조

〈표 1〉 타 시스템과의 비교

구 분	Sun Micro-systems 계열	Microsoft사 계열	본 연구 시스템
플랫폼	J2ME CLDC/MIDP	ME OS커널	J2ME CLDC/MIDP
문서 형식	WML/XML	WML	XML/HTML
프로토콜 (게이트웨이)	WAP	HTTP	HTTP
파서 위치	서버 (게이트웨이)	서버 (게이트웨이)	클라이언트 (모바일 폰)

2.3 XML Parser

일반적으로 파서는 서버에서 주로 사용되고 크기 또한 큰 것으로 알려져 있지만, MIDP상에서 MIDlet과 함께 JAR 파일 형태로 사용 가능한 작은 크기의 모바일 클라이언트용 파서들도 여러 종류가 공개되어 있다. 이들 파서들을 유형별로 구분해보면 크게 Model 파서, Push 파서, Pull 파서로 나누어볼 수 있다[14].

〈표 2〉 MIDP에 적당한 XML 파서

Parser Name	License	Size	MIDP	Type
ASXMLP 020308	Modified BSD	6kb	yes	push, model
kXML 2.0	CPL	9kb	yes	pull
kXML 1.2	EPL	16kb	yes	pull
MinML 1.7	BSD	14kb	no	push
NanoXML 1.6.4	zlib/libpng	10kb	patch	model
TinyXML 0.7	GPL	12kb	no	model
Xparse-J 1.1	GPL	6kb	yes	model

- Model 파서는 하나의 완전한 문서를 읽어 들여 처리하며, 다른 종류의 파서에 비해 메모리 사용에 있어 현저히 큰 편이다.
- Push 파서 역시 문서 전체를 읽어 들이며, 여러 종류의 문서를 만나면, listener 객체에 이를 통지한다.
- Pull 파서는 한번에 문서의 일부만을 읽어 들이며, 응용 프로그램은 이 파서를 통해 요구되는 문서의 다음 내용을 요청할 수 있다. 따라서 본 논문에서는 메모리와 파서의 크기를 고려하여 pull 파서 중 kXML 파서를 사용하기로 하였다[15].

3. XML 기반의 유 · 무선 통합 C/S 설계

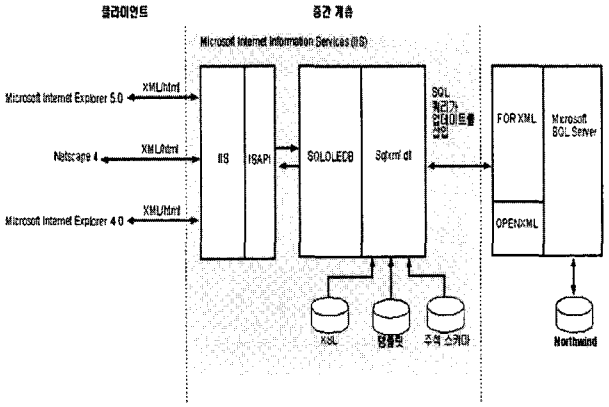
본 장에서는 2장에서 언급한 내용들을 기반으로 서버와 데이터베이스간의 XML 문서 생성 과정과 구조를 설명하고, 생성된 XML 문서가 각각의 유·무선 클라이언트에 적용되는 과정에서 발생되는 문제점과 해결 방안을 제시하고자 한다.

3.1 XML 서버 설계 시 고려 사항

기존의 웹서버 상에서 asp, jsp, php, cgi 등의 서버프로그램과 HTML 문서를 사용할 경우 발생할 수 있는 문제점들은 다음과 같다.

첫째, 기존의 서버프로그램을 이용하여 HTML 문서로 출력하는데 있어 실제 표현하고자 하는 데이터를 구조적으로 표현하기 어렵다. 둘째, 서버 프로그램의 작성이 어렵고 복잡하며 사용 프로그램별로 다양하여 유지 보수에 어려움이 있다. 셋째, 각기 다른 종류의 유·무선 클라이언트에 맞는 서비스를 제공하기에 적합하지 않고, 무선 클라이언트만을 위한 서버를 부가적으로 추가 설치해야 하는 문제점이 있다.

모바일 클라이언트 중 휴대폰의 경우 디스플레이가 기존 데스크탑보다 현저히 작으므로, 일반 클라이언트를 기준으로 작성된 문서를 WML이나 HTML 형태로 변환하여 제공할 경우, 데이터양이 너무 많아 메모리 부족 현상이 발생할 수 있다. 또한 디스플레이 크기를 고려하지 않아 내용을 읽는 시간보다 스크롤 하는데 더 많은 시간을 소비해야 하는 문제가 발생한다. 끝으로, 이 모든 문제점들을 서버가 모두 처리해야 하므로, 서버의 속도 저하와 부하를 유발시킬 수 있다.



(그림 3) XML 웹서버 구조

3.2 XML 서버 측 설계

첫째, 위에서 언급한 문제점 중 데이터의 구조적인 표현이 용이한 XML을 선택하였다. (그림 3)과 같이 XML은 문서의 타입을 정의하는 DTD를 이용하여 문법적으로 구조적 표현이 가능하며, XML 문서를 이해하는데 도움을 준다. 또한 XDR 스키마를 이용하면, XML 문서의 구조적으로 표현함에 있어 DTD보다 확장성이 우수하고, DOM 트리 내에 있는 노드를 이용하여 XML 문서의 요소나 속성을 변경할 수도 있으며, 데이터베이스의 열 값과 XML 문서의 요소가 서로 매핑 될 수 있으므로 본 논문에서는 (그림 3)과 같은 XML 서버 구조를 선택하였다.

둘째, XML 문서는 FOR XML을 사용하여 SQL 질의어 형태로 작성이 가능하며, 이 질의어를 데이터베이스가 해석하여 동적으로 XML 문서를 생성하고, XML 템플릿은 미리 지정한 IIS 가상 디렉토리에 XML 문서를 저장해준다. 또한 HTTP를 사용하여 SQL 서버로의 접근이 가능하므로, 클라이언트에서 요구하는 내용을 URL의 Xpath 질의 및 연

산을 이용하여 XML 문서의 질의어 매개변수로 사용하게 하면, 기존의 서버 프로그램이 수행하던 역할을 대신할 수 있어 프로그램 개발 시간과 노력을 줄일 수 있다. 그리고 사용 가능성이 높다고 판단되는 질의어들은 데이터베이스 내의 저장 프로시저로 미리 작성하여 필요할 때마다 호출하여 사용할 수 있게 함으로서, 데이터베이스가 질의어를 해석하는 시간을 줄일 수 있어 서버와 데이터베이스간의 응답 속도를 높일 수 있도록 설계하였다.

셋째, XML는 각기 다른 유·무선 클라이언트에 따라 XSL 스타일 시트를 달리 적용하여 서비스할 수 있도록 하며, 특히 모바일 클라이언트에 대해서는 다음절에서 언급되는 것과 같이 모바일 기기에 적합하도록 변환된 XML 문서 형태로, 클라이언트의 메모리가 수용할 만큼의 범위 내에서 필요한 부분만을 요약하여 서비스를 할 수 있도록 한다. 또한 기존 데스크탑 클라이언트와는 달리 모바일 클라이언트는 서버로부터 서비스를 받기 위해 우선 서버에 접속하여 원하는 응용 프로그램의 jad와 jar 파일을 다운받아 설치할 수 있도록 해당 파일을 서버에 업로드하고, MIME 타입을 설정해 둔다.

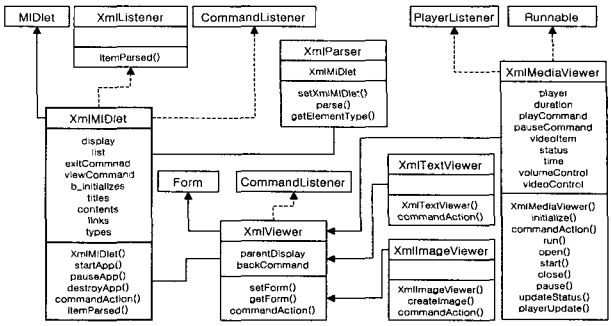
3.3 Mobile용 XML 클라이언트 설계 시 고려 사항

기본적으로 모바일 클라이언트 측 설계 시 고려된 사항은 다음과 같다. 첫째, 일반 데스크탑 클라이언트에 비해 메모리, 속도, 디스플레이의 크기가 작으므로, XML 문서에서 그다지 중요하지 않거나 불필요한 것들은 제거된 형태(Rich Site Summery)로 받을 수 있도록 XML 서버 측 설계 시 함께 고려하였다. 둘째, 한 레코드를 읽어 들임에 있어 이미지, 사운드, 동영상에 비해 상대적으로 크기가 작은 텍스트는 타이틀과 함께 메모리 허용 범위 내에서 우선 읽어 들이고, 이미지, 사운드, 동영상 등은 사용자가 해당 타이틀을 선택할 때, 서버에 접속하여 가져올 수 있도록 한다.

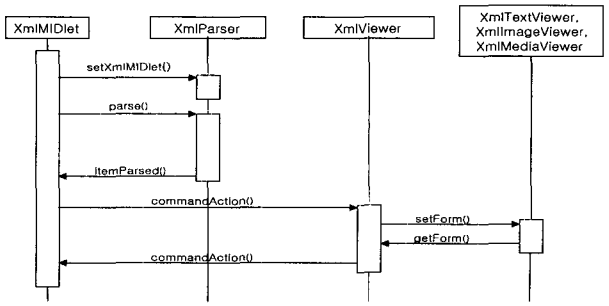
3.4 Mobile용 XML 클라이언트 측 설계

J2ME 기반의 MIDP에서는 서버에서 제공하는 HTML이나 XML 문서를 파싱할 수 없다. 따라서 WAP 게이트웨이 또는 서버를 두고, WAP 프로토콜을 이용하여 WML 문서로 변환하여 제공하고 있다. 본 논문에서는 2장에서 언급한 것과 같이 HTTP 프로토콜을 이용하며, MIDlet에 kXML 파서를 내장하여, 필요한 부분만을 요약하여 서비스 받을 수 있도록 함으로서 메모리 문제를 줄이고자 한다.

본 논문에서 제안하는 모바일용 XmlMIDlet 클래스와 시퀀스 다이어그램은 (그림 4), (그림 5)와 같다. XmlMIDlet 프로그램은 응용 프로그램에 포함된 아이템 항목을 나열하



(그림 4) 모바일용 XmlMIDlet 클래스 다이어그램



(그림 5) 모바일용 XmlMIDlet 시퀀스 다이어그램

고, 선택할 수 있도록 한다. 사용자가 해당 아이템 항목을 선택하면 XmlParser는 XmlListener를 초기 설정하고, XML 문서의 URL을 파싱한다. 그리고 HTTP 컨넥터를 통하여 해당하는 XML 문서를 읽어 들인 후 XML 문서를 시작 태그와 끝 태그를 기준으로 파싱하여 타이틀과 내용 그리고 링크가 있을 경우 링크주소 뿐만 아니라 링크된 파일의 타입까지 추출하여 XmlListener에 넘기고, 그 결과는 다시 XmlMIDlet 응용 프로그램에 의해 선택된 아이템 항목의 타이틀들을 모바일 클라이언트에 디스플레이 한다. 이때 사용자가 타이틀을 검색 후 원하는 타이틀을 선택하게 되면 해당되는 내용에 따라 XmlViewer를 통해 문자, 이미지, 사운드 또는 동영상 형태로 보여질 수 있게 설계하였다.

4. 구 현

본 장에는 3장에서의 설계를 기반으로 XML 기반의 서버 측과 모바일 클라이언트 측을 구현하고, 그 결과를 보이고자 한다.

4.1 구현 환경

본 논문의 구현을 위해 사용된 운영체제는 Microsoft Windows 2000 Advanced Server이고, 데이터베이스는 Microsoft SQL Server 2000을 사용하였으며, 데이터베이스와 서버 그리고 서버와 클라이언트간의 통신을 위해 Microsoft IIS v5.0을 사용하였다. 클라이언트 측 개발을 위해 Sun Mi-

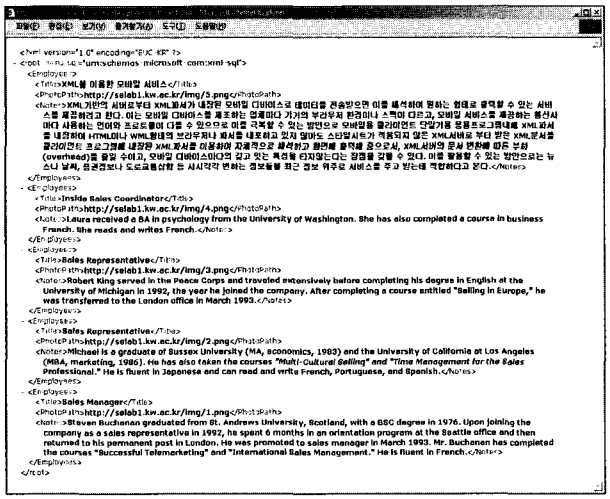
crosystems사의 Java 2 SDK SE v1.4.1, J2ME CLDC v1.04, J2ME MIDP v2.0 FCS, J2ME Wireless Toolkit v2.0 및 모바일 미디어를 위한 MMAPI Emulator for WTK v1.0을 사용하였고, 모바일용 클라이언트 측 XML Parser로는 CPL의 kXML v2.0을 사용하였다.

4.2 서버 측 구현

우선 데이터베이스로부터 서버로 데이터를 읽어들여 XML 문서의 생성할 수 있도록 다음과 같이 SQL 질의어를 포함하는 XML 문서를 작성하였다.

```
<?xml version = "1.0" encoding = "EUC-KR" ?>
<root xmlns : sql = "urn : schemas-microsoft-com : xml-sql">
 <sql : query> SELECT Field1, Field2, Field3 FROM
    Table_Name order by Index_ID desc FOR XML AUTO,
    Elements
 </sql : query>
</root>
```

위에서 보는바와 같이 SQL 질의는 테이블 Table_Name으로부터 Field1, Field2, Field3 필드의 데이터를 Index_ID의 역순으로 가져오게 된다. 이는 최근에 등록된 글부터 보여주기 위함이며, RSS를 적용하여 반듯이 필요하다고 판단되는 필드만을 출력할 수 있게 하였다. 필요에 따라서는 Field명 대신 @Field를 사용하여 클라이언트로부터 파라미터 형식으로 받아 처리하는 것도 가능하다. FOR XML AUTO문은 데이터를 자동으로 XML 문서 형식으로 출력하기 위함이고, Elements는 데이터베이스의 각 필드들이 속성(attribute)이 아닌 요소(element)로서 출력되게 하기 위함이다. 이 문서를 실행한 결과는 (그림 6)과 같다.

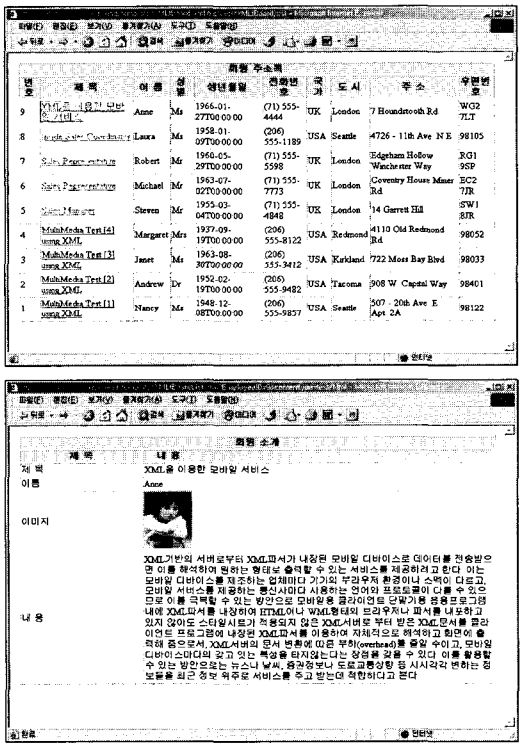


(그림 6) XML문서의 실행 결과

이렇게 만들어진 XML 문서는 유·무선 클라이언트들의 호출 시 각각의 클라이언트들을 위해 준비된 XML 스타일시트인 XSL 문서들과 결합하여 클라이언트 측에 전달된다. 각 클라이언트에서 호출되는 경로는 다음과 같다.

```
http://iisserver/virtualroot/virtualname/[pathinfo][XpathExpression]?
xsl = Style_Sheet.xsl & ContentType = Content_Type
```

위에서 "?" 다음에 나오는 XSL 문서는 필요에 따라 하나의 XML 문서에 각기 다른 Style_Sheet를 적용할 수 있으며, ContentType은 text/html, text/xml, text/plain, image/jpeg 등 표현해야 할 데이터의 내용과 클라이언트 측의 환경을 고려하여 설정할 수 있다. 본 논문에서는 ContentType을 일반 데스크탑 클라이언트에는 text/html, 모바일 클라이언트에는 text/xml을 각각 적용하기로 하였다. 일반 데스크탑 클라이언트에서 실행한 결과는 (그림 7)과 같다.



(그림 7) 데스크탑 클라이언트에서 실행 화면

4.3 모바일 클라이언트 측 구현

본 논문에서 구현한 모바일 클라이언트 프로그램은 클라이언트를 수행하기 위한 **XmlMIDlet**과, kXML Parser를 이용하기 위한 **XmlParser**, Parsing된 결과를 저장하기 위한 인터페이스로 **XmlListener**, 그리고 최종 사용자에게 텍스트, 이미지, 사운드 및 동영상을 보여주기 위한 **XmlViewer**로 구성된다.

XML 문서의 파싱된 결과를 저장하기 위한 인터페이스

인 XmlListener 알고리즘은 다음과 같다.

```
public interface XmlListener {
    public void itemParsed (String Title, String Contents,
                            String Link, String mime_Type) ;
    public void exception(java.io.IOException ioe) ;
}
```

XML 문서를 kXML 파서를 이용하여 태그들을 파싱하는 XmlParser 알고리즘은 다음과 같다.

```
// import 될 부분 기술
public class XmlParser {
    protected XmlListener RssListener ;
    public void setXmlListener (XmlListener listener) {
        RssListener = listener ; }
    public void parse(final String url) {
       // 파싱 하려는 데이터가 URL인 경우 http Connector를 통해
       // XML문서를 open한다. }
    public void parse(InputStream in) throws IOException {
       // Reader를 통해 입력 스트림을 읽어들이기 위한 초기설정.
       // kXMLParser를 통해 Xml.Start_Tag를 찾기 위한 초기설정
       // 파싱 이벤트 pe를 초기화.
       while (tracking) {
         if (pe.getType == Xml.Start_Tag) {
            // Start Tag 초기화
            while ((pe.getType() !=Xml.End_Tag) {
         if (pe.getType() = Start_Tag&&
            pe.getName().equals(@Field_Name)) {
         // 해당 Tag의 Field 내용을 읽어들여 저장.
                }
             // RssListener.itempased를 호출.
             }
         else {
              // pe.getType() != Xml.End_Tag인 동안 계속 읽음
         }
       if (pe.getType() == Xml.End_Tag) {
           tracking = false ;
       }
}
```

모바일 클라이언트용 응용 프로그램인 XmlMIDlet의 알고리즘은 다음과 같다.

```
// import 될 부분 기술
public class XmlMIDlet extends MIDlet implements
    CommandListener, XmlListener {
    // 초기 변수 선언
    public XmlMIDlet() { // 초기 값 설정 }
    public void startApp() {
        if (Initialized == false) {
            // Command, List, Display 초기화
            getApproperty("XmlMIDLet.URL") ;
            XmlParser parser = new XmlParser() ;
            parser.set_XmlListener(this) ;
            parser.parse(url) ;
            Initialized = true ;
        }
        else
            Display.setCurrent(TitleList) ;
    }
 public void pauseApp() {}
 public void destroyApp(boolean unconditional) {}
 public void commandAction(Command c, Displayable s) {
  if (c ==DetailsCommand || c == List.SELECT_COMMAND) {
     int selection = TitleList.getSelectedIndex() ;
     // selection의 mime_Type에 따라 각기 다른 인자로
     // XmlViewer 클래스의 인스턴스 호출
     display.setCurrent(XmlViewer) ;
  }
 }
 public void itemParsed (String Title, String Contents,
                         String Link, String mime_Type) {
  // Element 추가
  // Display 설정
  // Title List에 추가
 }
  // 예외처리
}
```

사용자가 선택한 Title의 내용의 MIME 종류에 따라 해당하는 내용을 보여주기 위한 XmlViewer 알고리즘은 다음과 같다.

```
// import 될 부분 기술
public class XmlViewer extends Form implements
     Runnable, CommandListener, PlayerListener {
// 각종 변수 설정 및 초기화
public XmlViewer (String Title, String Contents,
                  Display parentDisplay) {
          // 텍스트 출력을 위한 XmlTextViewer 호출 }
public XmlViewer (String Title, String Contents,
                  String Link, Display parentDisplay) {
          // 이미지 출력을 위한 XmlImageViewer 호출 }
public XmlViewer (String Title, Display parentDisplay) {
          // 동영상 출력을 위한 XmlMediaViewer 호출 }
// 기타 예외 처리
}
```

4.4 모바일 클라이언트 측 구현 결과

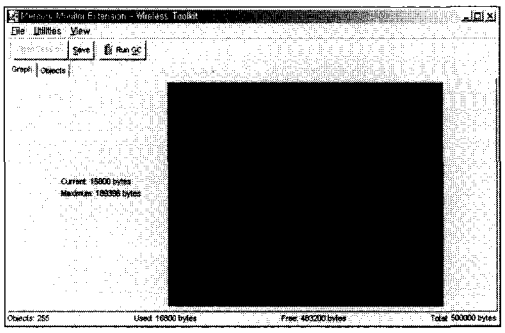
(그림 8)(a)는 사용자가 선택할 수 있도록 타이틀을 출력한 화면이다. (그림 8)(b)는 타이틀을 선택한 경우 텍스트를 출력한 결과 화면이고, (그림 8)(c)는 텍스트와 이미지를 동시에 출력한 화면이며, (그림 8)(d)는 동영상을 출력한 화면이다.

(그림 8)에서 텍스트와 이미지를 포함하는 (그림 8)(a), (그림 8)(b), (그림 8)(c)는 DefaultColorPhone으로 실행하였고, 동영상을 포함하는 (그림 8)(d)는 MMEmulator로 실행하였다.

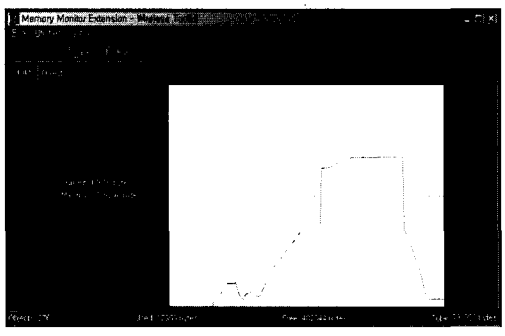
(a) 타이틀 (b) 텍스트 (c) 이미지 (d) 동영상

(그림 8) 구현된 결과의 실행화면

간단한 성능 분석을 위해 텍스트와 이미지의 메모리 사용량을 각각 비교해 보면 (그림 9)와 같다. (그림 9)(a), (그림 9)(b) 두 그림 모두 처음 도입 부분은 응용 프로그램을 로딩 하는데 사용한 메모리 양이고, 두 번째는 타이틀을 로딩 하는데 사용한 메모리 양으로 메모리 사용에 큰 차이가 없다. 그러나 실제 데이터를 읽어 들이는 세 번째 부분에 있어서 (그림 9)(a)는 메모리 사용양의 증가가 거의 없는 반면 (그림 9)(b)는 새로운 이미지를 로딩 해야 함으로 메모리 사용량이 급증함을 볼 수 있다. 따라서 기존의 WAP이나 ME에서와 같이 대량의 데이터를 처음에 모두 읽어 들이는 방법보다는 본 논문에서 제시한 방법이 메모리 활용 면에서 더 우수하고도 본다.



(a) 텍스트



(b) 이미지

(그림 9) 메모리 사용량 그래프

5. 결 론

본 논문에서는 데이터베이스와의 연동부분과 데이터의 구조적 표현, 그리고 다른 문서로의 변환이 용이한 XML 서버를 기반으로 특별한 서버 프로그램 없이 XML 문서만으로도 이를 대체할 수 있도록 서버의 설계를 단순화하였으며, 모바일 클라이언트에 XML 파서를 내장함으로 서버의 부담을 줄일 수 있도록 하였다.

또한 기존의 WAP 방식에서는 사용자가 요구할 경우 클라이언트가 일방적으로 일정 양만큼의 데이터를 한번에 모두 전송 받는데 비해, 본 논문에서는 모바일 클라이언트 측면에서 pull 방식의 XML 파서를 사용하여 사용자가 요구한 부분만을 읽어 들이게 함으로 불필요한 메모리 낭비를 줄일 수 있도록 하였다.

향후 연구 과제로는 체계적이고 최적화된 디자인 패턴을 개발하고, 이를 적용한 XML 서버 프로그램과 MIDlet 프로그램을 자동생성 할 수 있는 도구의 개발과, SOAP 기반하에서 보다 효율적인 성능을 제공할 수 있는 XML 서버와 모바일 클라이언트의 설계에 대한 연구가 필요하다.

참 고 문 헌

[1] 채진석, "XML 문서와 DB의 효율적인 연동을 위한 XML 스키마 기술 연구", 한국정보과학회 데이터베이스 연구, 제16권 제2호, pp.23-34, 2000.

[2] 이봉규, "XML기반의 Mobile기술에 관한 연구", 정보처리학회지, 제8권 제3호, pp.54-60, May, 2001.

[3] H. M. Deitel and P. J. Deitel, "XML How to Program," Prentice Hall, 2001.

[4] J. Craig Cleaveland, "Program Generators with XML and JAVA," PH PTR, 2001.

[5] Kevin Dick, "XML A Manager's Guide," Second Edition, Addison Wesley, 2002.

[6] Neil Bradley, "The XSL Companion," Addison Wesley, 2000.

[7] John W. Muchow, "Core J2ME Technology & MIDP," Prentice Hall, 2001.

[8] Jonathan Knudsen, "Wireless JAVA : Developing with J2 ME," second edition, Apress, 2003.

[9] Michael Kroll and Stefan Haustein, "Java 2 Micro Edition Application Development," SAMS, 2002.

[10] Ben Forta, "WAP Development with WML and WMLS cript," SAMS, 2000.

[11] Danny Ayers and Adnrew Patzer, "Professional Java Server Programming," Wrox Press, 1999.

[12] Ronald Ashri and Danny Ayers, "Professional Java Mobile Programming," Wrox Press, 2001.

[13] Yu Feng and Jun Zhu, "Wireless Java Programming with J2ME," SAMS, 2001.

[14] Jonathan Knudsen, "Parsing XML in J2ME XML in a MIDP Environment," http://wireless.java.sun.com/midp/articles/parsingxml/, Mar., 2002.

[15] http://kxml.org/api/org/kxml2/io/KXmlParser.html.

박 영 수

e-mail : yspark@kw.ac.kr

1996년 광운대학교 전산대학원 전자계산학과(이학석사)

1999년 광운대학교 대학원 컴퓨터과학과 수료

1998년~현재 경복대학 겸임교수

관심분야 : XML, 분산처리, 객체지향프로그래밍, 소프트웨어 공학, 무선인터넷, 모바일컴퓨팅 등

장 덕 철

e-mail : dcjang@kw.ac.kr

1982년 고려대학교 대학원 경영정보학박사

1981년~1982년 버클리 대학교 객원 교수

1993년 광운대학교 전산사회교육원 원장

1996년 광운대학교 전산대학 원장

1997년 광운대학교 이과대학 학장

1997년~현재 광운대학교 컴퓨터과학과 교수

관심분야 : 소프트웨어공학, 버전 제어, 컴포넌트, 객체지향설계 방법론 등