유·무선 인터넷 환경에서 XML 웹서비스를 위한 통합 XML Parser 구현

권두위* · 도경훈**

Implementation of the XML Parser for Integrated XML Web Service in Wired and Wireless Internet Environment

Doo Wy Kwon* · Kyeong Hoon Do**

이 논문은 2009년도 동서대학교 동서프런티어과제 연구비 지원에 의하여 연구되었음.

요 약

XML 웹서비스는 인터넷을 이용한 오픈 네트워크를 통해 단일 또는 다수의 비즈니스 업체 간의 기존 컴퓨터 시스템 프로그램을 결합시키는 표준화된 소프트웨어 기술로서 기업의 수익증대와 비용절감의 효과를 얻을 수 있을 것으로 기대된다. 특히 모바일 매쉬업(Mashup)을 위해서는 XML 웹서비스의 처리가 필수적이다. 본 논문에서는 WIPI 환경과 웹서비스 상에서 사용할 수 있는 XML 파서를 구현하였다. 그리고 HTTP 통신이 WIPI상에서 가능하도록 하여 XML 웹서비스의 서비스 개발에 적용함으로써 모바일 환경에서의 XML 웹서비스 인터페이스를 구현하였다. 유무선 환경에서 동시 사용가능하게 적용한 것은 어떠한 환경에서도 적용 가능한 파서임을 나타낸다. 또한 기존의 파서들과의 성능비교를 통해 제안한 파서가 속도면에서 뛰어난 점을 보여준다. 이에 대한 실제 응용으로서 WIPI용 Java언어를 기반으로 인터넷 서점에 적용하였다.

ABSTRACT

XML Web Service is a new distributed computing paradigm for combining all computer systems through the on-line standardized open network, so it is expected that companies can get the results of increasing profits and reducing costs through this. Especially XML Web Service is essential in mobile environment. In this paper, a new XML Web Service is proposed and it is based on WIPI-based XML standard technology and mobile web service standard. The proposed XML Web Service embodied XML Parser which can be used in WIPI environment and made HTTP communication. In addition, as an example of the practical application, it was applied to an inventory Management, Book Store using JAVA language for WIPI.

키워드

XML Parser, RPC, WIPI, SOAP, XML Web Service

Key word

XML Parser, RPC, WIPI, SOAP, XML Web Service

* 동서대학교 대학원 유비쿼터스IT학과

** 동서대학교 컴퓨터정보공학부 (교신저자)

접수일자 : 2009. 11. 11

심사완료일자 : 2009. 11. 26

Ⅰ. 서 론

웹서비스란 인터넷상에서 단일 비즈니스 또는 다수의 비즈니스 업체 간의 기존 컴퓨터시스템 프로그램을 결합시키는 표준화된 소프트웨어 기술로서 이러한 표준 기술을 이용해 모든 비즈니스 기능 또는 서비스를 가능케 하는 활동이다. 인터넷을 통한 웹서비스는 거래업체간의 이질적인 운영시스템과 이질적인 프로그램언어 간의 커뮤니케이션 차이를 극복해주는 연결고리 역할을 해준다.

XML 웹서비스의 기본적인 요소는 XML(인터넷을 통해 데이터를 교환하는 표준양식), UDDI(Universal Description, Discovery, and Integration : 업체가 자사의 웹서비스를 온라인 디렉터리에 등록·광고하거나 외부에서 웹서비스를 검색), WSDL(Web Service Description Language : 웹서비스를 정의하는 언어), 및 SOAP(Simple Object Access Protocol : 인터넷을 통해 웹서비스의 통신)이다.

XML 웹서비스(XML Web Service)는 인터넷상에서 분산 컴퓨팅 환경으로 이동하기 위한 기본적인 구축 블록(Fundamental Building Block)이다. 공개 표준 응용 프로그램간의 커뮤니케이션 및 협력의 집중은 XML 웹서비스가 응용프로그램 통합을 위한 플랫폼으로 자리 잡는 환경을 만들어 오고 있다. 응용프로그램은 여러 언어의 소스에서 다중 웹 서비스를 사용하여 구축되는데, 사용되는 소스들이 어디에 위치하는지 또는 어떻게 실행되는지 상관없이 협력한다. XML 웹서비스는 표준 웹 프로토콜을 사용하며, 클라이언트와 서버간의 통신을 XML 문서로 제공한다.[1]

본 논문에서는 이러한 XML 웹서비스를 실현하고자 인터넷 표준 기술인 XML과 HTTP를 사용하여 XML 웹서비스를 구축하였다. XML기술은 표현 가능성이 한정되어 있는 텍스트 기반의 프로토콜 기술들을 대체 할 것으로 예상되며 주고받을 데이터와 서비스 인터페이스도 XML로 표현하는데 주고받을 데이터 및 서비스 호출 메시지를 위해 XML형식의 SOAP을 사용하였다.

또한 국내 표준 모바일 플랫폼규격 WIPI(Wireless Internet Platform for Interoperability)기반의 모바일 단말로의 내장을 위한 XML 파서(XML Parser)를 구현하고, 모바일 장치에서의 외부 DTD사용이 줄어드는 것을 염두에 두었다.

XML 웹서비스에서는 다양하고 많은 XML 파서를 개발하여 사용하고 있지만 아직까지 XML 파서의 기준이나 표준은 정해져 있지 않다. 이에 본 논문에서 Pull모델을 변형한 이벤트방식의 XML 파서를 제안하였다.

본 논문에서는 설계 및 구현한 유무선 파서를 중심으로 기술한다. 2장 관련연구에서는 XML 웹서비스에 대한 기술과 함께 SOAP의 기능에 대한 기술과 SOAP의 통신에 대해 기술하고, 3장 XML 파서 설계에서는 제안한 파서의 구조와 파싱과정, 기존의 파서와의 성능을 비교를 하여 기술한다. 4장 구현에서는 WIPI API의 구현 과정과 인터넷서점에 적용하여 개발한 WIPI API의 활용을 보여주고, 웹서비스에서 개발한 인터넷서점에서 활용을 보여준다. 5장 결론에서는 유·무선 환경 개발 과정에서 제기된 문제점과 개선 방안에 대하여 기술한다.

Ⅱ. 관련연구

2.1 XML 웹서비스

XML 웹서비스란 "인터넷을 이용한 오픈 네트워크를 통해 단일한 비즈니스 또는 다수의 비즈니스 업체 간의 기존 컴퓨터 시스템 프로그램을 결합시키는 표준화된 소프트웨어 기술"로서 이러한 표준 기술을 이용해 모든 비즈니스를 가능하게 하는 활동을 일컫는다. 이러한 XML 웹서비스는 PC, PDA, 핸드폰 등 다양한 디바이스를 통해서 접근 가능하다.

XML 웹서비스의 특징은 다음과 같다.[2]

- 시스템 구조의 유연성 : 메인프레임 또는 서버, 클라이언트 방식과 달리 유연한 소프트웨어 구조를 통해 이질적인 데이터 표준을 유연하게 통합/운영
- 사용의 편리성 : 사용자는 소프트웨어를 설치한 후 자연스럽게 서비스를 제공받게 되며, 인터넷을 연결할 수 있는 유/무선 단말기를 통해 장소에 관계없는 접근이 가능
- 기존의 시스템의 통합 환경을 제공 : 이질적인 어플리케이션간의 통합 서비스를 제공받을 수 있고, 새로운 시스템과의 통합도 자동적으로 이루어짐
- 비용 효율적 : 분산 시스템의 소프트웨어 간 통합을 자

동화적으로 이행해줌으로써 개별 기업마다 투입해야 하는 IT개발 및 운영비용을 절감

또한, XML 웹서비스의 장점은 다음과 같다.[3,4]

- 서비스 이용도(Availability) : 이미 존재하는 인터넷 관련 기술을 이용
- 서비스 이용 용이성(Transparency) : HTTP가 되는 곳에서는 쉽게 이용가능
- 서비스 추상화(Encapsulation) : 내부가 어떻게 구현되었는가에 대해 사용자가 알 필요가 없음
- 플랫폼 독립성(Platform Independent) : 특정 기술에 얽매이지 않음
- 표준기반(Standard Based) : HTTP, XML기반 기술 이용
- 상호운용성(Interoperability) : 표준 기술사용으로 상호운용성 확보
- 지원 용이성(Support) : 표준 기술사용으로 다양한 기술적 지원 가능
- 표준화된 서비스 : 어려운 기술적인 내용은 감추고 분산되어 있는 프로그램을 사용자가 원하는 서비스의 형태로 제공함. 표준화는 개발자나 개발 업체의 관점에서 장점이고, 서비스는 사용자의 관점에서의 장점

2.2 SOAP

2.2.1 SOAP의 구조

SOAP은 분산 환경에서 정보를 교환하기 위한 간단하고 가벼운 표준 XML 프로토콜로 정의할 수 있는데, 그 특징으로는 XML형식의 문서이며 XML 웹서비스에서 메시징 스택에 해당한다. 단순히 메시지 형식만을 정의하여 이기종간의 데이터 교환 문제를 해결한 프로토콜이며 응용 프로그램 사이에서 공유되는 데이터 형식을 정의하는 프로토콜이다. 이러한 SOAP의 설계 목표는 그 정의에도 나타나 있듯이 간단함(Simplicity)과 확장성(Extensibility)에 있다. 즉, 분산 프로그래밍을 위한 어떠한 특별한 내용이나 제한 사항도 담고 있지 않다.

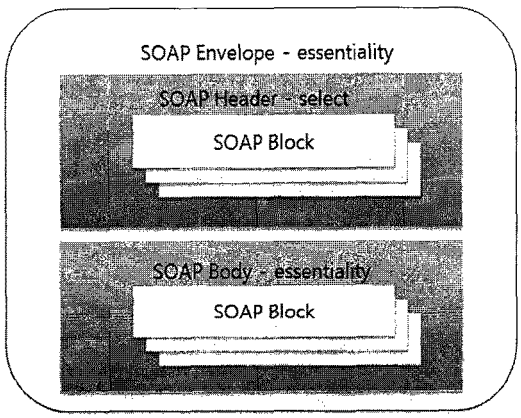


그림1. SOAP 메시지 구조
Fig.1. SOAP Message Structure

SOAP 헤드(Header)에는 메시지를 처리하는 방법, 메시지 라우팅, 인증 및 트랜잭션들을 정의한다. 그리고 SOAP 바디(Body)에는 실제의 XML형식 메시지가 정의되는 부분으로 호출하고자 하는 함수명, 인자 등이 기술된다. SOAP 바디내에는 SOAP 블록(Block)형태의 바디 블록들이 정의된다.

SOAP 메시지를 이용하여 RPC(Remote Procedure Call)를 처리하는 방법은 웹서비스에서는 SOAP HTTP Request로 서비스를 호출하고 서비스 호출의 결과는 HTTP SOAP 메시지로 전달받는 것을 의미하며 필요한 정보는 목적지 URL, 메서드 이름, 메서드 용법, 메서드 인자, 헤더 정보가 있다.[5]

2.2.2 SOAP의 엘리먼트

(가) <Envelope>엘리먼트

<Envelope>엘리먼트는 SOAP메시지의 루트 엘리먼트이다. 자식 엘리먼트로 <Header>와 <Body>엘리먼트가 올 수 있다. <Header>엘리먼트는 생략이 가능하지만, <Body>엘리먼트는 반드시 한 개 이상 있어야 한다.

(나) <Header>엘리먼트

<Header>엘리먼트에는 클라이언트 인증, 트랜잭션 관리와 같은 기능을 추가할 수 있다. <Header>엘리먼트의 자식 엘리먼트들을 헤더 엔트리라고 한다. 헤더 엔트리는 반드시 네임스페이스 접두사를 사용한 Qualified Name으로 기술해야 한다.

(다) <Body>엘리먼트

<Body>엘리먼트는 최종 수신자가 수신해야 할 정보가 담겨 있다. 일반적으로 원격 프로시저 호출(RPC) 요청, 원격 프로시저의 실행 결과, 실행 시 발생한 에러 등이다. 이와는 다르게 원격 프로시저 호출 이외에도 단순한 정보 전달용으로 사용될 수 있다.

(라) <Fault>엘리먼트

<Fault>엘리먼트는 요청 SOAP메시지를 처리하는 도중 발생한 오류 정보를 발신자로 보내는 응답 SOAP메시지 내에 기술하기 위해 사용된다. <Body>엘리먼트의 자식 엘리먼트로 작성된다. <Fault>엘리먼트는 응답 SOAP메시지 안에 오직 하나만 있을 수 있다.

Ⅲ. XML Parser 설계

3.1 XML Parser

파서는 프로그래밍이라는 전체적인 구도에서 보면 작은 부분에 지나지 않지만 매우 중요한 역할을 한다. XML문서를 다루는 어플리케이션은 XML문서 안에 있는 정보에 접근하여야 하는데 XML 파서가 그 역할을 한다. 대표적인 XML 파서로는 SAX(Simple API for XML)와 DOM(Document Object Model)이 있다. SAX와 DOM 모두 XML문서의 정보에 접근 가능한 어플리케이션을 개발하도록 API를 제공한다.

SAX와 DOM 각각의 API는 다른 특성을 지니고 있다. SAX는 소프트웨어나 컴퓨터 자체에 의해서 생성된 데이터 정보를 읽어 들이는 프로그램에 적당하고, DOM은 문서에 저장된 정보를 읽어 들이는 경우에 적당하다, 즉, 개발 환경에서 XML문서가 데이터를 읽어 들이는 경우에 적합하다. 개발환경에서 XML문서가 데이터를 포함하고 있으면 SAX를 사용해서 읽는 것이 쉽고, XML이 문서를 포함하고 있을 경우에는 DOM을 사용해서 읽는 것이 쉽다. 이에 본 논문에서 제안한 파서는 SAX방식을 이용하였다.

3.2 제안한 XML 파서

현재 파싱의 모델로는 Object, Push 및 Pull의 모델이 존재한다. 3가지의 파서들중 속도가 가장 빠르고 가벼운 파서는 Pull 모델의 파서이다. Pull 파서는 기존의 Push 모델을 변형한 것이다. 그림 2는 XML 파서의 SOAP문서의 데이터를 서로 전송하고 비교하는 다이어그램이다.

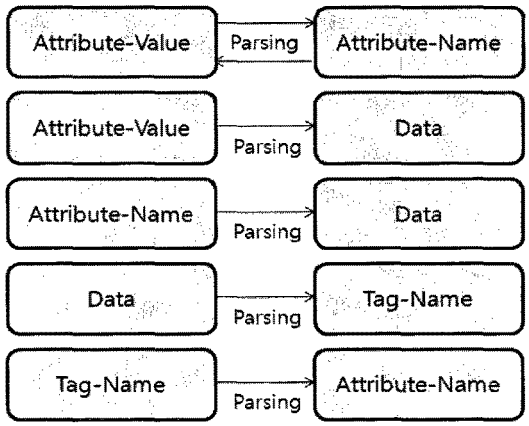


그림 2. 데이터 전송 다이어그램
Fig.2. Data Transmission Diagram

XML 파서는 속성 값과 속성이름을 서로 비교하여 전송되어 오는 문서에서 속성을 찾아낸다. 그리고 속성 값과 속성이름에서 각각의 데이터가 유효한 데이터 인지를 파싱한다. 그 다음으로 그 데이터 값을 Tag와 비교하고 그 Tag를 통해 속성이름 동일한 값인지를 확인하여 파싱하게 된다. XML 문서를 제안한 파서를 이용하여 태그들을 파싱할 때의 파생클래스의 상태 정의는 다음과 같다.

```
class DXml_Data : public DXmlState
{
Public:
  DXml_Data(DxmlParser & Parser);
  void Parse(const char data);
} //Data parsing

class DXml_Attr : public DXmlState
{
Public:
  DXml_Attr(DxmlParser & Parser);
  void Parse(const char data);
} //Attribute-Value parsing
```

```
class DXml_Tag : public DXmlState
{
Public:
 DXml_Tag(DxmlParser & Parser);
 void Parse(const char data);
} //Tag name parsing

class DXml_AttrValue : public DXmlState
{
Public:
 DXml_AttrValue(DxmlParser & Parser);
 void Parse(const char data);
} //Attribute-name parsing
```

본 논문에서 구현하는 파서는 응용 프로그램에서 파싱의 요청이 생기면 파일의 Token을 추출하고 이벤트형을 정의하며 정의된 이벤트형은 응용프로그램의 요청시 반환하게 된다. 이벤트를 처리하는 부분은 callback 메서드로 정의하였다. 구현된 XML 파서는 원격 접속을 위한 HTTP클라이언트로 동작될 수 있다. 그림 3은 제안한 XML 파서의 내부 구조도이다.

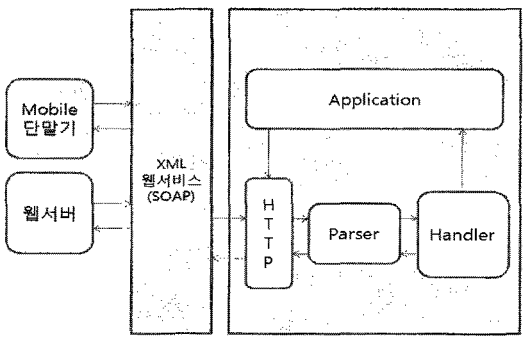


그림 3. 제안한 XML 파서의 구조.
Fig.3. Structure of the Proposed XML Parser

그림 4와 5는 기존의 파서들과 제안한 파서의 성능을 벤치마킹한 결과이다.

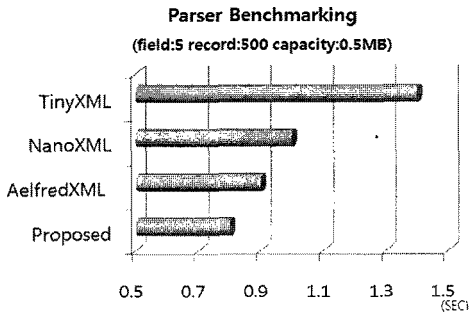


그림 4. 모바일기기에서의 파싱 속도 비교
Fig.4. Comparison of the Parsing Speed on Mobile Device

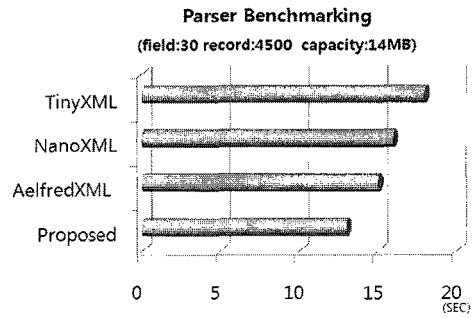


그림 5. 웹서버에서의 파싱 속도 비교
Fig.5. Comparison of the Parsing Speed on Web Server

속도비교를 위해 사용한 기존의 파서로는 TinyXML [6], NanoXML[7] 및 AElfredXML[8] 파서를 사용했다. 그림 4는 무선 인터넷환경에서 기존의 파서들과의 파싱 속도를 비교하여 나타내었다. 제안한 파서는 기존의 파서보다 파싱속도가 뛰어난 점을 알 수 있다. 그림 5는 유선환경에서 기존의 파서들과의 파싱속도를 비교하였다. 유선환경에서도 기존의 파서들보다 속도가 향상되었음을 알 수 있다.

비교한 XML 파서들의 파싱정확도는 모두 100%를 보였고, 제안한 XML 파서가 다른 파서들보다 파싱속도가 뛰어난 결과를 볼 수 있다.

Ⅳ. 구 현

인터넷 서점을 구현하기 위해서 XML과 SOAP, UDDI 및 WSDL을 사용하였고, 응용을 위해서 서버측은 SUN 사의 J2ME, 클라이언트측에서는 WIPI 라이브러리를 사용하였다. 서버는 Tomcat 5.5, Apache Axis v1.4로 설치하였으며, WIPI 에뮬레이터로는 Aroma WIPI Emulator를 사용하였다. 인터넷 서점의 서버와 클라이언트는 HTTP 통신을 이용하여 서비스에 필요한 데이터를 송수신한다. 서버에서는 클라이언트에서 송신하는 SOAP메시지를 파싱하여 클라이언트에서 요청한 데이터만 DB의 데이터와 비교하고, 비교한 결과값을 SOAP 메시지로 클라이언트에 보내게 된다. 클라이언트 또한 서버에서 송신해온 SOAP 메시지를 파싱하여 결과값을 출력한다.

```
Request

POST /axis/services/SuGang HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.4
Host: 127.0.0.1:5678
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 338

<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <ViewSJDB soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </soapenv:Body>
  </soapenv:Envelope>
```

그림 6. 요청 SOAP 메시지
Fig.6. Request SOAP Message

클라이언트가 서버에 접속시 그림 6과 같이 SOAP메시지를 송신한다. 또한 서버는 정상 접속되었음을 알리는 SOAP메시지를 그림 7과 송신한다.

```
Response

POST /axis/services/SuGang HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime, multipart/related, text/*
User-Agent: Axis/1.4
Host: 127.0.0.1:5678
Cache-Control: no-cache
Pragma: no-cache
SOAPAction: ""
Content-Length: 338

<?xml version="1.0" encoding="UTF-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <ViewSJDB soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </soapenv:Body>
  </soapenv:Envelope>
```

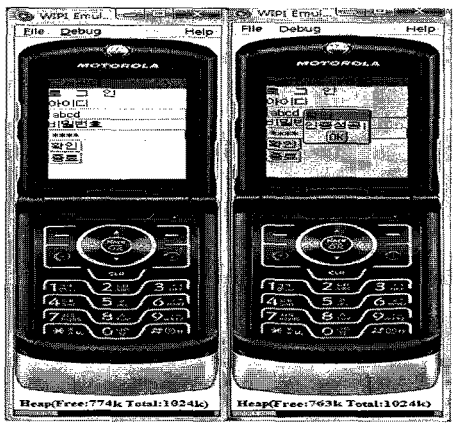
그림 7. 응답 SOAP 메시지
Fig.7. Response SOAP Message

송수신되는 SOAP메시지는 요청하는 질의에 따라 Body부분이 변동되게 된다.

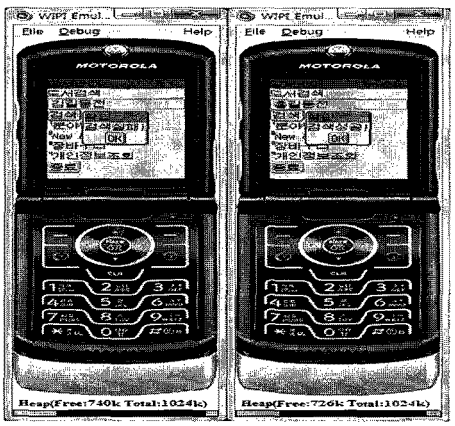
그림 8(a)에서 송수신된 XML문서를 XML 파서가 파싱후 로그인되는 API화면이다. 서버는 JAVA의 Class를 Deploy시켜 모든 데이터를 처리 한다. 또한 수신한 SOAP문서는 제안한 XML 파서를 이용하였다.

그림 8(b)은 검색과정을 나타낸 그림이다. 검색을 한 내용이 Database에 존재하게 되면 검색결과는 나타내고, 검색내용이 존재를 하지 않으면 검색 실패가 나타나게 된다.

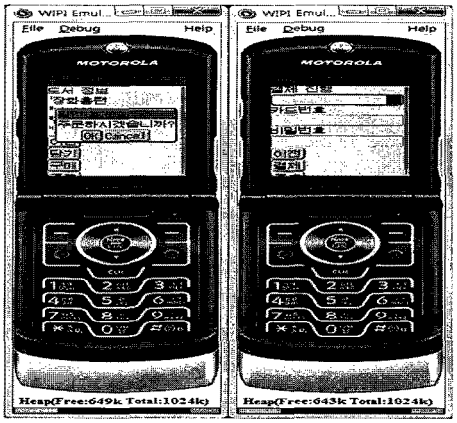
그림 8(c)은 클라이언트에서 물품의 구매를 요청하거나 결제 현황을 알아보는 과정을 보여주는 그림이다. 모바일 기기를 통해 구매 및 결제 현황을 소비자가 수정함으로써 제품 구매를 할 수 있다.



(a)



(b)



(c)

그림 8. WIPI API구현. (a) 로그인 (b) 검색
(c) 구매 및 결제
Fig.8. WIPI API : (a) login, (b) search and
(c) purchase and payment.

그림 9. 웹서비스 구현
Fig.9. Implementation of Web Service

그림 9는 웹서버 상에서 XML 파서를 이용하여 웹서비스를 구축하여 성능을 테스트 하였다. 인터넷 서점의 초기화면과 세부 메뉴를 이용할 수 있다.

V. 결 론

현재 기업이나 공공 기관에서 웹서비스 이용 시 XML 문서에 대한 XML 파서의 표준이 존재하지 않아 많은 기업에서 필요에 따라 개별적으로 제작을 하게 된다. 또한 서비스를 보다 광범위하고 사용하기 쉽게 해주는 XML 웹서비스는 차세대 인터넷으로 주목받고 있으며, 국제 표준인 XML이라는 Markup Language를 사용하여 전 세계 공개된 웹서비스가 있다면 24시간 언제라도 접근하여 사용가능하게 되었다.

인터넷은 정보 제공 서비스 외에도 다양한 서비스를 제공하기 위하여 많은 발전을 하고 있다. 웹서비스 제공자가 제공할 서비스를 개설하여 UDDI를 정의함으로써 서버 관리자와 사용자에게 정보를 전달될 수 있도록 하고 있다. 이들 간의 통신은 HTTP를 사용하는 쉽고 간편한 프로토콜인 SOAP을 이용한다.

본 논문에서는 XML 웹서비스를 위해 유·무선 환경에서 사용가능한 XML 파서를 개발하고 SOAP을 통해 필요한 정보를 송수신을 하였으며, 인터넷 서점에서 필요로 하는 기능들을 모바일 기기와 웹서비스에 적용하였다. 모바일 기기는 사용자의 요구사항에 맞추어 사용자의 편리성을 증가시키고 장소의 제약을 없애주고 있어 그 효과는 상당히 기대된다. XML 파서를 사용함에 따라 서비스 제공하는 회사는 이기종간의 호환문제를 해결하여, 비용과 인력 손실을 줄일 수 있다.

그러나 무선이라는 점을 감안하여 사용자가 지불하는 높은 비용과 신뢰성, 동시 접속의 문제점도 안고 있다. 그 비용과 시간에서 손실이 많이 발생을 하겠지만, 앞으로 증가추세로 볼 때, 이 문제는 해결 될 것으로 보인다. 앞으로 XML 파서와 검색 알고리즘을 더욱더 보완하고 모바일 기기의 강점을 이용하여 개발해 나간다면, 실용성과 상업성면에서도 많은 이익 창출과 기대효과를 보게 될 것이다. 본 연구는 WIPI기반이었으나, 다른 모바일 플랫폼에서도 사용가능하게 버전업한다면 항만, 물류 등 다양한 분야에 유용하게 사용될 것이다.

감사의 글

본 연구는 2009년도 동서대학교 동서프런티어과제 연구비 지원에 의하여 이루어진 연구입니다.

참고문헌

[1] 권수갑, "Web Services 동향", 중소기업청 정보화 지원단, 2003

[2] 오지훈, "Web Services 통신 프로토콜 분석", 인프라벨리, 2006

[3] E. A. Lee, "What ahead for Embedded Software?", IEEE Computer, Vol. 33, 2000

[4] 홍준성 "모바일 플랫폼 기술현황 및 발전방향", 한국정보과학회지, 22권, 1호, pp. 8-15, 2004

[5] 강미연, "WIPI 기반의 모바일 단말을 위한 내장형 XML파서 및 뷰어", 한국정보과학회, 31권, pp. 865-867, 2004

[6] TinyXML, "http://www.grinninglizard.com"

[7] NanoXML, "http://nanoxml.cyberelf.be"

[8] AElfred XML, "http://saxon.sourceforge.net"

저자소개

권 두 위(DooWy Kwon)

2007년 2월 : 동서대학교 컴퓨터공학과 졸업 (공학사)
2009년 2월 : 동서대학교 유비쿼터스IT학과 졸업 (공학석사)
2009년 3월~현재 : 동서대학교 유비쿼터스IT학과 박사과정
※관심분야 : 모바일컴퓨팅, 무선센서네트워크

도경훈(KyeongHoon Do)

1990년 2월 : 경북대학교 전자공학과 졸업(공학사)
1992년 2월 : 경북대학교 전자공학과 졸업(공학석사)
1995년 8월 : 경북대학교 전자공학과 졸업(공학박사)
1996년 3월~현재 : 동서대학교 컴퓨터정보공학부 부교수
※관심분야 : 영상처리, 인공지능, 모바일컴퓨팅