# mSAX : 임베디드 시스템을 위한 XML SAX2 파서*

문미경°, 고민정, 강미연, 음두헌
덕성여자대학교 전산학과
moon@namhae.duksung.ac.kr°, mjko@namhae.duksung.ac.kr,
mykang@icantek.com, dheum@duksung.ac.kr

# mSAX : a XML SAX2 Parser for Embeded Systems

MiKyoung Moon°, MinJeung Ko, MiYeon Kang, DooHun Eum
Dept. of Computer Science, Duksung Women's Univ.

**요약**

임베디드 시스템은 임의의 시스템에 내장된 형태로 구성되고, 특정 하드웨어 구성요소를 제어하기 위하여 사용되며, 제한되고 전문화된 기능을 수행하는 장치이다. 임베디드 시스템에서 호스트간의 데이터 전송을 용이하게 하는 문서 정의 형식으로 향후 XML이 사용될 전망이다. 임베디드 시스템은 특정한 기능을 수행하기 위한 최소 용량이 요구되고 임베디드 시스템의 다양한 메모리 제한성에 부합할 수 있게 컴포넌트화도 요구된다. 본 논문에서는 임베디드 시스템에서 실행되는 응용을 위한 XML SAX2파서, mSAX를 소개한다.

## 1. 서론

요즘 IT 기술의 급속한 발전과 함께 일상생활에서도 흔히 접하게 된 것 중의 하나가 임베디드 시스템이다. 일반적으로 임베디드 시스템은 임의의 시스템에 내장된 형태로 구성되고, 특정 하드웨어 구성요소를 제어하기 위하여 사용되며, 제한되고 전문화된 기능을 수행하는 장치로 정의된다[1]. 이러한 임베디드 시스템들은 네트워크를 수단으로 디지털 TV, 인터넷 셋톱박스, 개인 휴대 통신단말기 및 각종 정보형 단말기 등으로 상호 연결되어 강력하고 유용한 시스템으로 발전하고 있다.

임베디드 시스템 상에서 호스트간의 데이터 전송을 용이하게 하는 문서 정의 형식으로 1996년 W3C(World Wide Web Consortium)[2]에서 제안된 XML이 사용될 전망이다. XML은 인터넷 상에서 구조화된 전자문서를 표현하고 처리하기 위한 표준으로서 그 중요성은 이미 널리 확대되고 있다. XML 문서의 사용이 빈번해 짐에 따라 문서의 사용범위가 커지고 새로운 시스템으로의 응용도 커지고 있다. XML은 내용과 스타일을 구분하여 처리하기 때문에 XML 문서 그 자체만으로도 데이터를 다양한 형태로 표현하고 변환이 가능하다. 네트워크의 문서 표준인 XML을 기반으로 한 임베디드 시스템에 탑재하기 위해서는 이러한 XML 문서를 분석하고 처리하기 위한 XML 파서가 경량화, 맞춤화 되는 것이 필수적이다.

본 논문에서 소개하는 mSAX(mobile SAX)는 well-formed XML만을 처리하는 최적화 XML SAX2 파서이다. 유선 네트워크 환경을 위한 상용 XML 파서, Xerces2, crimson1.1.3, piccolo1.3[3] 등은 경량화와 유지/보수/재사용성을 위한 컴포넌트화를 고려해야 하는 임베디드 시스템 환경에는 적합하지 않다. 반면 mSAX는 목적에 따라 필요한 기능만을 선택하여 구성할 수 있도록 컴포넌트화를 목표로 구현된 SAX2 API를 지원하는 기본 파서이다. 기존의 XML 파서보다 5~36배정도 용량을 줄인 경량의 파서이다. 2절에서는 mSAX의 개요를 설명하고 3절에서는 구현 및 실행 예를 간략히 보인 후, 마지막 절에서는 본 논문의 결론과 앞으로의 연구 방향을 정리한다.

* 본 연구는 한국과학재단 목적기초연구 (R06-2002-003-01004-0) 지원으로 수행되었음.

## 2. mSAX2의 개요

본 절에서는 mSAX의 동작원리를 소개하고 상용 파서들과의 크기 비교를 통해 임베디드 시스템을 위한 mSAX의 적합성을 보인다.

경량의 XML SAX2 파서인 mSAX는 임베디드 시스템 응용의 메모리 제한 요구에 맞게 그림 1과 같이 SAX 파서의 핵심 인터페이스들 중 ContentHandler와 ErrorHandler[3] 인터페이스만을 지원하였다. mSAX의 동작원리를 살펴보면 Application은 createXMLReader 메소드로 파서 인스턴스를 생성한다. Application의 ContentHandler와 ErrorHandler는 SetContentHandler와 SetErrorHandler 메소드를 통해 mSAX와 연결시킨다. Application에서 parse 메소드를 호출하면 mSAX의 Parser 클래스는 nextLex 메소드로 XML 문서의 어휘를 분석하는 Lexer 클래스가 넘겨준 토큰을 받아 구문분석을 시작한다. 구문분석 중에 Lexer로부터 받은 토큰이 prolog, element, attribute 등인 경우 mSAX는 event를 Application으로 보내 Application의 Handler에서 처리하도록 한다. Application이 해당 Handler의 구현에 따라 처리를 하면 다시 mSAX는 문서를 마칠 때까지 이 과정을 반복한다.

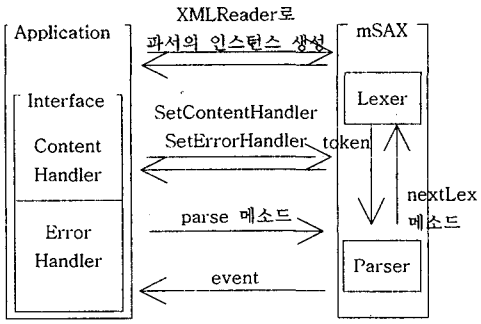


그림 1. mSAX의 동작원리

상용 파서인 Xerces2, crimson1.1.3, piccolo1.3 등은 유선 인터넷 환경을 전제로 하였기 때문에 mSAX에서 제공하는 최소한의 기능 외에도 DOM, DTD, XML Schema, Namespaces 등을 모두 지원한다[3][4]. 이런 파서들은 SAX 파서의 핵심 인터페이스인 ContentHandler, ErrorHandler, DTDHandler, EntityResolver와 그 밖의 기능들을 모두 지원하기 때문에 표 1과 같이 경량화, 컴포넌트화가 되지 않아 임베디드 시스템의 메모리 제한성을 만족시키지 못한다.

표 1. mSAX의 경량화

| Parser | Xerces2 | crimson1.1.3 | piccolo1.3 | mSAX |
|---|---|---|---|---|
| 용량 | 1.54M | 293KB | 217KB | 42.3KB |
| 지원 사항 | DOM, SAX, XML Schema JAXP, Namespace 등 | | | SAX |

이에 반해 mSAX는 임베디드 시스템 응용이 요구하는 경량화와 컴포넌트화를 고려하여 최소한의 기능을 제공하고 well-formed XML 문서만을 파싱하는 기능을 제공한다. 핵심 인터페이스들 중 가장 기본적인 ContentHandler와 ErrorHandler 두 개의 인터페이스를 지원하여 임베디드 시스템에 적합한 용량으로 경량화 하였다.

## 3. mSAX 구현 및 실행 예

본 절에서는 mSAX가 수행되는 과정을 그림 2의 클래스 계층도를 이용해 설명하고 예제 XML 문서를 처리한 결과를 보인다. 그림 2에서 Lexer 클래스는 XML 문서를 읽어 파서가 처리할 수 있는 최소단위인 토큰으로 분리한다. 인터페이스 XMLReader로부터 구현된 Parser 클래스는 Lexer가 보내준 토큰들이 well-formed XML 문법에 맞는지 확인하는 구문분석 기능을 제공한다.

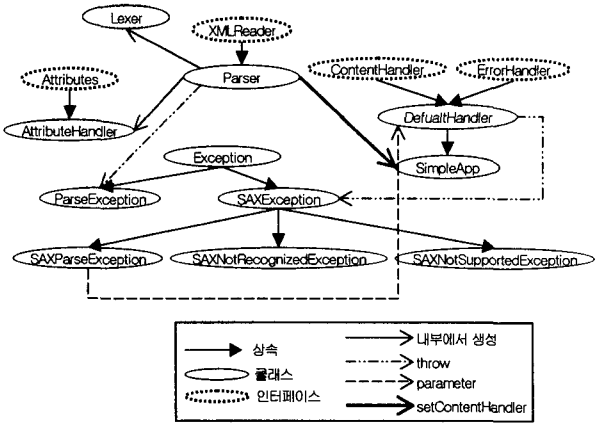


그림 2. 클래스 계층도

DefaultHandler는 인터페이스인 ContentHandler와 ErrorHaldler로부터 구현되어 element와 error에 관한 event를 처리하는 메소드를 가진다. 본 논문에서는 DefaultHandler 클래스의 메소드 중 문서의 시작과 끝을 인식하면 호출되는 메소드인 startDocument, endDocument와 element의 시작과 끝을 인식하면 호출되는 메소드인 startElement, endElement와 element 사이의 데이터를 인식하면 호출되는 characters를 재정의하여 구현하였다. 구문분석 중 문서의 시작과 끝, element의 시작과 끝, 또는 시작 element 내의 attribute가 인식되면 이 메소드들을 이용해 SimpleApp에게 event 형태로 전달한다. 같은 방식으로 구문분석 중 error가 발생하면 warning, error, fatalError 메소드를 이용해 SimpleApp에게 전달한다. ParseException 클래스는 구문분석 중 발생하는 error를 처리하고 SAXException 클래스는 event 처리 중 발생하는 에러를 처리한다.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!-- 이 문서는 mSAX를 실행하기 위한 문서입니다. -->

<band type="progressive">
        <name>King Crimson</name>
        <bass>Boz</bass>
        <drums> Ian Wallace</drums>
</band>
```

그림 3. 예제 XML 문서

그림 3은 prolog, 주석, element, 그리고 attribute만으로 구성된 간단한 well-formed XML 문서이다. 이 문서에서 band element는 자식으로 name, bass, drums element를 갖고 band element는 type이란 attribute를, 자식 element들은 character data를 갖는다.

```
package parser;
import java.io.*;
import org.xml.sax.XMLReader;
 < 중간생략 >
class SimpleApp extends DefaultHandler {
  public SimpleApp()   {  super();  }
  public static void main(String [ ] args) throws Exception
    {
          XMLReader xr =
          XMLReaderFactory.createXMLReader("parser.Parser");
          SimpleApp handler = new SimpleApp();
          xr.setContentHandler(handler);
          xr.setErrorHandler(handler);

          // Parse each file provided on the command line.
          for (int i = 0; i < args.length; i++)   {
              FileReader r = new FileReader(args[i]);
              xr.parse(new InputSource(r));
            }
    }
  public void startDocument ()                {
          System.out.println("[SAX API - startDocument]");
    }
  public void endDocument () { ..... }
  public void startElement (String uri, String localName,
          String qName, Attributes attributes)    {
          System.out.println("[SAX API - startElement - " +
                  qName + "]");
    }
  public void endElement (String uri, String localName,
          String qName)     { ..... }
  public void characters (char ch[], int start, int length)    {
          System.out.println("[SAX API - characters]");
    }
}
```

그림 4. 응용 프로그램

그림 4는 DefaultHandler를 상속받아 일부 메소드만 재정의한 응용 프로그램(SimpleApp.java)이다. 구문분석 중 event가 발생하면 이 응용 프로그램의 해당 메소드를 호출하여 실행하게 된다. 본 논문에서 재정의한 메소드는 해당 event 발생을 메시지 출력으로 알리나 실제로 응용 프로그램을 구현할 경우, 출력문 대신 event 발생 시에 필요한 수행코드를 작성하면 된다. SimpleApp는 main 메소드부터 실행되기 시작한다. XMLReaderFactory의 createXMLReader 메소드를 이용해 XMLReader 클래스로부터 xr 객체를 생성한다. DefaultHandler와 상속관계에 있는 SimpleApp 클래스의 객체인 handler를 생성한다. DefaultHandler는 인터페이스인 ContentHandler와 ErrorHaldler로부터 구현되어 element와 error에 관한 event를 처리하는 메소드를 가지므로 handler 객체도 같은 기능을 한다. XMLReader의 setContentHandler 메소드 수행에 의해 xr 객체는 handler 객체의 참조를 갖게 되고 구문분석 중 파서가 'element의 시작'과 같은 event를 인식하면 handler 객체의 해당 메소드에서 이를 처리한다.

그림 5는 그림 4의 예제 프로그램을 실행시킨 결과이다. 문서의 시작과 끝, element의 시작과 끝 등을 인식한 후 메시지를 출력한 결과이다. 예를 들어 그림 3의 시작 element인 <band type= ... >를 인식하면 SAX API -- startElement - band라는 메시지를 출력하게 된다.

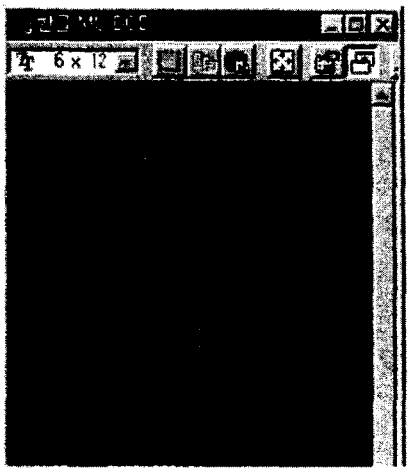

그림 5. 실행 결과

## 4. 결론

요즘 각광받고 있는 임베디드 시스템에서 호스트간의 데이터 전송을 용이하게 하는 문서 정의 형식으로 XML이 사용될 전망이다. XML에서 제공하는 모든 기능을 포함하는 응용을 처리하는 XML 파서는 임베디드 시스템에 탑재되기에 용량이 너무 크다. 임베디드 시스템의 메모리 제한 요구를 충족시키는 XML 파서를 탑재시키기 위해 기능별 경량화와 더불어 컴포넌트화는 필수이다. 본 논문에서 소개하는 mSAX는 표 1과 같이 상용 파서들과 비교해 경량이므로 임베디드 시스템 응용에 적합하다. 현재 mSAX는 well-formed XML 문서를 위한 파서이지만 향후 다른 기능을 지원하는 컴포넌트들을 개발하여 컴포넌트의 조립을 통해 원하는 기능을 다양하게 수행하는 컴포넌트 기반 파서로 발전시킬 계획이다.

## 참고문헌

[1] 김문자, 조인준, 조기환, "네트워크 기반 대규모 임베디드 시스템의 상호협동을 위한 Smart Message 기법", 정보처리학회지, 제9권 제1호, 2002, pp.60-62.

[2] W3C, Extensible Markup Language (XML) Version 1.0, http://www.w3.org/TR/Rec-xml, Feb. 10, 1998.

[3] David Brownell, "SAX2", O'Reilly & Asociates, 2002, pp.14-17, pp.33-44.

[4] David Megginson, Simple API for XML (SAX), http://www.megginson.com/SAX/index.html.

[5] Edward A. Lee, "What's Ahead for Embedded Software?", IEEE Computer, September 2000, pp.18-26.