XML 응용 개발환경

배재대학교 김창수* · 정회경**

1. 서 론

정보화 사회로 되어감에 따라 컴퓨터의 보급이 늘면서 컴퓨터를 이용한 문서처리의 중요성이 증가해가고 있고, 컴퓨터 관련 분야의 기술진보로 문서를 전자적으로 처리하기 위한 노력으로 워드 프로세서, 전자출판시스템과 같은 전자 문서처리 시스템이 현실화되고 디지털 도서관, 인트라넷 상에서 CALS, 전자상거래 등 대량의 전자문서를 관리하거나 구축하여 효율적인 정보 서비스를 요구하고 있다. 그러나 이러한 시스템으로 작성된 문서는 각기 독자적인 문서구조와 각기 다양한 내용정보를 갖고 있으므로 서로 다른 시스템 및 장치를 갖는 문서처리환경에서 이들 문서의 교환 및 공유를 위한 표준 문서구조 모델이 요구되어 왔다.

이에 따라 ISO(International Organization for Standardization)에서는 OSI(Open System Interconnection) 환경에서 서로 다른 이 기종간의 효율적인 문서정보 교환을 위한 전세계 공통 방식의 모델로 IS 8613, ODA/ODIF(Open Document Architecture/Open Document Interchange Format) 및 SGML(Standard Generalized Markup Language)을 국제표준으로 제정하였다. 또한 인터넷의 발전으로 인터넷상에서 구조화된 전자문서를 표현하고 처리하기 위한 표준으로 W3C(World Wide Web Consortium)에서는XML(eXtensible Markup Language)이 발표되었다[1].

쉽고 일관된 방법으로 인터넷상에서 데이타 전송을 용이하게 하는 문서 정의 형식으로 1996년 W3C에서 제안된 XML은 실제로 SGML의 부분집합으로 웹을 대상으로 구조화된 문서의 전달 및 응용 프로그램에서 처리를 보다 쉽게 하는 표준 텍스트 형식으로 이는 현재 HTML(Hyper Text Markup Language)의 단순한 마크업 언어를 뛰어 넘어 웹에서 사용할 수 있는 메타 미디어의 유용성을 제공한다. 또한 SGML에 기반을 둔 유연한 언어이므로 문서를 구조적으로 정의하고 문서의 구조적 형태를 이어 받아 XML은 내용과 스타일을 구분하여 처리한다. 이는 XML의 장점으로 들 수 있는 데 XML 문서 그 자체만으로 데이타를 다양한 형태로 표현하고 변환이 가능함을 말한다.

XML은 구조화된 정보를 사용자와 제공자들의 효율적인 정보교환뿐만 아니라 범용적인 시스템으로 작업을 제공하는 효과적인 수단이 된다. 특히 XML은 기능 면으로는 구조화된 문서의 작성을 지원하므로 복잡한 문서의 작성을 용이하게 하며, 인터넷에 기반한 언어이기 때문에 SGML보다 인터넷 언어로서 사용이 용이하다.

본 고에서는 차세대 인터넷 문서 표준인 XML을 기반으로 새로운 응용을 개발할 때 필요한 XML 프로세서에 대해 기술하고 이벤트 기반 API인 SAX(Simple API for XML)와 객체 모델기반 API인 DOM(Document Object Model) 프로세서에 대해 기술한다.

2. XML 프로세서와 API

XML 프로세서는 "XML문서를 읽어서 문서

* 학생회원
** 종신회원

내용과 문서구조에 대한 접근을 제공하는 소프트웨어"이다. 다시 말하면 프로세서는 문서와 DTD를 읽어 문서의 정확성을 검증하고, 문서내의 엘리먼트, 속성, 컨텐트 트리, 엔티티 정보를 생성해 내며, 이를 이용하여 XML 응용은 해당 정보(주로 엘리먼트)에 대한 특정 동작을 기술함으로서 XML 문서의 처리를 수행하도록 지원하는 기능을 제공한다.

XML 프로세서를 이용하는 방법에는 크게 두 가지 방법이 존재한다. 하나는 XML 브라우저를 이용하여, 문서를 파싱, 해석하며 XSL (eXtensible Stylesheet Language)이나 CSS (Cascading Style Sheets)와 같은 스타일시트 정보를 이용하여 적절히 표시/인쇄하는 프로그래밍이 필요없는 방법이고, 둘째는, XML 프로세서가 제공하는 API를 이용하여 XML 문서에 대한 검증, 접근, 갱신, 수정 등의 처리 작업을 할 수 있도록 XML 응용 프로그래밍 하는 방법이다.

첫번째 방법은 XML 브라우저(인터넷 익스플로러, 넷스케이프)의 기능에 전적으로 의존하는 방법이며, 이를 이용하면 간단하게 XML 문서를 다룰 수는 있으나 다양한 XML 응용에 모두 적용하기에는 융통성이 적다.

두번째 방법은 XML 프로세서가 제공하는 API를 이용하는 방법으로 다양한 XML 응용에 입맛에 맞추어 필요한 기능만 사용해서 XML 문서정보를 다룰 수 있다. 초기에는 자체 API를 이용해 XML 프로세서를 다루게 하였으나, XML 응용 개발을 쉽게 하기 위해 표준 API가 만들어져 사용되고 있다. 이러한 표준 API로는 이벤트 기반의 SAX와 객체 모델 기반의 DOM이 있다.

이벤트에 기반한 접근 방법은 XML 문서를 파싱할 때, 문서의 전체 구조 정보를 가지지 않으며 단순히 문서 내에 특정 엘리먼트를 만났을 때, 프로그래머가 수행토록 요구된 작업을 할 수 있도록 이벤트를 발생시켜, 이를 이용해 필요한 정보를 접근할 수 있도록 한 API이다.

객체 모델에 기반한 접근 방법은 XML 문서를 파싱하여 문서의 구조정보와 컨텐트 모두를 객체로서 메모리에 올려놓고 이를 이용하는 방법이다. 이 경우, 문서 전체에 대한 구조 정보를 트리에 기반한 객체로서 이용 가능하므로 XML 에디터와 같이 XML 문서를 구조적으로 변경하는 작업등에 적합하며, 여러 응용들이 XML 문서를 메모리 상에서 공유할 수 있다는 장점이 존재한다. 그러나, 문서전체에 대한 정보를 메모리 상에 올려놓음으로써 크기가 큰 XML 문서를 처리할 시에는 그에 따른 메모리 사용량도 늘어나는 단점이 있다[2~5].

그림 1은 객체모델 기반의 DOM과 이벤트 기반의 SAX의 XML 정보처리 환경에서의 역할을 보이고 있다. DOM은 XML 문서나 DTD를 파싱해 내부 트리로 만들며, 애플리케이션은 이 트리를 탐색, 조작하는 식으로 작업을 수행하고 SAX는 파싱 이벤트를 애플리케이션의 콜백(callback)을 통해 직접 알려준다.

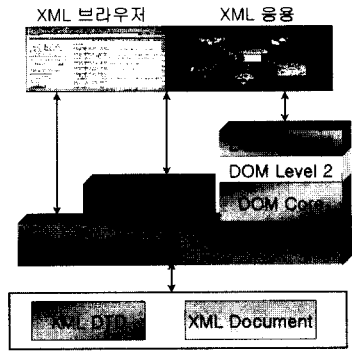

그림 1 XML 정보처리 환경에서 DOM 과 SAX의 역활

3. DOM 개요

DOM은 HTML과 XML 문서를 위한 응용프로그램 인터페이스이며, DOM은 문서를 접근하고 조작하기 위한 방법으로서 문서의 논리적 구조를 정의하고 있어 사용자들은 문서를 생성하고, 그 문서의 구조에 따라 항해(Navigation:문서 트리 정보의 운행)하고, 엘리먼트와 문서 내용을 추가/수정/삭제할 수 있다. DOM은 현재 세 개의 Level이 발표 되었으며, DOM Level 1 권고안은 W3C에 의해 98년 10월에 권고안으로 제정되었으며, DOM Level 2는 역시 권고안으로 제정되어 있고, DOM Level 3도 작업기술서(Working Draft)로 만들어지기 위해 요구사항이 만들어져 W3C에 제출되어 작업중에 있다[1][6].

DOM Level 1은 XML과 HTML문서 정보를 위한 DOM Core API와 추가된 HTML 문서정보를 위한 DOM HTML API로 이루어져 있다. 그리고, DOM Level 2는 문서에 부가적으로 연결되어 CSS 같은 스타일정보에 대한 접근 및 처리, UI에서의 이벤트 처리, 접근제어 및 네임스페이스의 지원을 정의하고 있으며, DOM Level 3은 문서의 구조 검증을 위해 DTD나 스키마와 같은 문서 형 정의부에 대한 정보를 포함한 웹 문서의 적재(Loading)나 저장(Saving)문제, XPath를 이용하여 DOM 트리에 대한 질의, 문서 보기 및 포맷팅 그리고 키 이벤트와 이벤트그룹 처리에 대한 API를 준비하고 있으며, 그 이상의 레벨에선 운영체제 수준의 윈도우 인터페이스, 질의언어, 멀티쓰레드(Multithreading) 및 동기화 문제, 보안 및 저장소에 관련된 문제를 다루고자 하고 있다[3].

현재 DOM Level 1과 DOM Level 2 가 그 모습이 완전히 제시되어 있고, 그 이상은 준비중이다[7].

DOM의 중심부분(Core)은 가장 핵심적인 인터페이스들로 구성되고, 추상화된 계층 구조(Hierarchy Structure) 구성과 특성들은 인터페이스 상속을 통해 노드(Node)들이 가지게 된다. 노드는 문서를 구성하는 최소 단위 인터페이스로 노드 타입에 따라 관계(Relation)에 범위가 정해진다. 그림 2는 DOM 노드간의 관계를 정의한다.

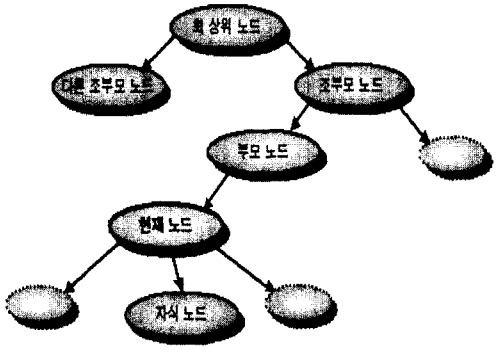

그림 2 DOM의 노드간의 관계

그림은 현재 노드와 각 노드와의 관계를 도식화 한 것이다. 문서의 최상위를 나타내는 최상위 노드(Root Node)와 그 자식간의 관계로 표현한다. DOM에서 기준이 되는 노드는 현재 노드인데, 현재 노드는 부모와 자식노드의 정보로 구성된다. 또한 각 노드는 노드 형(Type)을 가지는데 다음과 같다. Node, DocumentFragment, Document, CharacterData, Text, Comment, CDATASection, Attributes, Element, DocumentType, Notation, Entity, EntityReference, ProcessingInstruction으로 구성되며 이외에 문서 관리에 도움되는 인터페이스들로 구성된다.

그림 3은 DOM으로 표현한 XML구조 요소 관계를 나타낸다

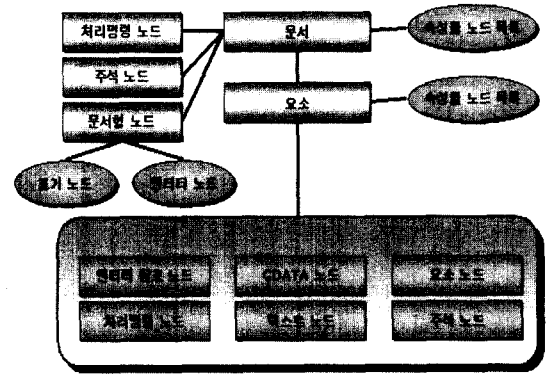

그림 3 DOM으로 표현한 XML구조 요소 관계

XML 문서는 구조화된 문서이기 때문에 DOM 처리가 가능하다. DOM 처리를 위해 그림과 같이 XML의 주요 구성 요소를 나눈다. 각각의 노드는 계층구조를 가지고 있는데 자식 노드를 가질 수 있는 노드가 모두 같은 것은 아니다. XML DOM 구조의 최상위 노드로 문서(#document)가 되면 자식요소로 처리 명령 노드(processing instruction)와 문서형 노드(document type), 그리고 주석 노드(#comment)를 가진다. 최상위 노드는 자식 요소 이외에 속성들 노드 목록(attributes)을 가진다. 다음으로 문서의 내용을 이루는 요소(element)를 가진다. 문서를 구성하는 최상위 요소와 자식 요소들로 XML 문서의 실례를 구성한다. 자식 요소를 구성할 수 있는 요소들은 엔티티 참조 노드(entityreference), CDATA 노드(#cdata-section), 요소 노드(element), 처리 명령 노드(pro-

cessinginstruction), 텍스트 노드(#text), 주석 노드(#comment)이다.

DOM을 사용할 때 많이 사용하게 될 메소드들은 다음과 같다.

▶Document.getDocumentElement() : XML 문서의 최상위 엘리먼트 리턴

▶Node.getFirstChild(), Node.getLastChild() : 주어지는 노드의 처음과 마지막 노드 리턴

▶Node.getNextSibling(), Node.getNext Sibling() : 주어진 노드의 이전 이후 노드 리턴

▶Node.getAttribute(attrName) : 주어진 노드에 파라미터로 들어온 attrName을 가진 속성 리턴

DOM을 사용하여 XML 응용을 생성하는 과정은 다음과 같다

1) 파서 객체를 생성
   DomParser를 이용해 파서 객체를 생성
2) 파서 객체를 이용하여 XML 문서 파싱
   Document 객체를 얻는다
3) 파서 객체로부터 생성된 Document 객체 처리
   각 노드의 타입을 얻고 그 타입에 따라서 트리구성

그림 4는 아래 XML 문서에 대한 NodeType, name, Value를 생성하여 트리 형태로 보이는 프로그램 예와 결과이다[8~10].

```
<?xml version="1.0"?>
<AUCTIONBLOCK>
  <ITEM>
    <TITLE>Vase and Stones</TITLE>
    <ARTIST>Linda Mann</ARTIST>
    <DIMENSIONS>20x30 inches</DIMENSIONS>
    <MATERIALS>Oil</MATERIALS>
    <YEAR>1996</YEAR>
    <BIDS>
      <BID>
        <PRICE>3000</PRICE>
        <TIME>2:47:58 PM</TIME>
        <BIDDER>opening price</BIDDER>
        <TIMESTAMP>1298</TIMESTAMP>
      </BID>
    </BIDS>
    <MTIME>
      <TIMESTAMP>1315</TIMESTAMP>
      This is an example of mixed content
    </MTIME>
  </ITEM>
</AUCTIONBLOCK>
```

```
dim strStruct
dim elementNum
dim code(100)
dim selectedNodes(100)
dim selectedNodesType(100)
dim selectedNodesLength
dim displayFrame
dim treeViewNS
treeViewNS = "urn:tv" 'namesspace 노드
dim tempNode
sub windowLoad()
set displayFrame = parent.display
displayFrame.document.open()
displayFrame.document.write("")
displayFrame.document.close()
sub loadXML()
'xml 파일 적재 및 객체 생성
  if (XMLFile.value = "") then
    alert("An xml file must be specified for loading to
      occur.")
  else
    xmldoc.async = false
    xmldoc.load(XMLFile.value)
    if (xmldoc.parseError.errorCode <> 0) then
      alert(errtxt)
      windowLoad()
    end if
    if (not isnull(xmldoc.documentElement)) then
      displayTree()
      selectionString.value = "xmldoc."
    end if
  end if
end sub
sub displayTree()
'트리 구조 보기
  if (xmldoc = null) then
    alert("No xml file has been loaded.")
  else
    selectedNodesLength = 0
    strStruct = ""
    buildTree xmldoc.documentElement, 0, 0
    displayFrame.document.open()
    displayFrame.document.write(strStruct)
    displayFrame.document.close()
  end if
end sub
 sub buildTree(node, level, last)
 ' 생성 트리 재귀적 호출
 ' level 은 트리 깊이.
 ' 종단 엘리먼트는 boolean
    if (level <> 0) then
'공백 추가
dim j
for j = 0 to level - 2
'텍스트 노드 처리
 if (node.nodeType = TEXT_NODE) then
                    addhtml("<FONT COLOR=" & quot
                    & textColor & quot & ">" & node.
                    text & "</FONT>")
   elseif (node.nodeType = COMMENT_NODE) then
     else
'각 엘리먼트에 속성 추가
      dim elementID
     ElementID = "TreeViewer_" & elementNum
  addhtml("<SPAN ID=" & quot & elementID & quot &
">" & node.nodeName & "</SPAN>")
     elementNum = elementNum + 1 Increment ID counter.
    dim tvNode
     set tvNode = xmldoc.createNode("attribute",attID,tvs
```

```
        ViewNS)
        tvNode.text = elementID
        node.attributes.setNamedItem tvNode
      dim attNum
        attNum = node.attributes.length
      if (attNum > 0) then
          dim I
        '속성 보기
          for i = 0 to attNum - 1
            then
            on error resume next
            err.clear
            dim test
            test = node.attributes.item(i).namespace
            if err.number <> 0 then
            else
    addhtml("<B> : </B><SPAN ID=" & quot & elementID
    & "_att_" & i & quot & " STYLE=" & quot & "color:
    green" & quot & " >" & node.attributes.
    item(i).nodeName & "</SPAN>") ' + " = \"" + node.
    attributes.item(i).text + "\"";
            end if
            on error goto 0
          next
        end if
      end if
'현재 노드 아래의 child 노드
    dim children
      children = node.childNodes.length
      if (children <> 0) then
        for i = 0 to children - 1
          if (i = (children - 1) ) then
        buildTree node.childNodes.item(i), (level + 1), 1
          else
           '마지막 엘리먼트
            code(level) = 0
            buildTree node.childNodes.item(i), (level + 1), 0
          end if
        next
      end if
    end sub
```

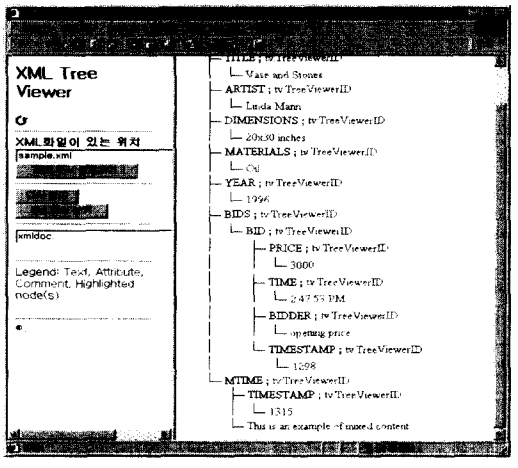

그림 4 XML 트리 뷰

4. SAX 개요

SAX는 객체 모델 기반이 아닌 이벤트 시퀀스로 XML 문서의 정보에 접근하는 이벤트 기반의 XML 분석을 위한 표준 인터페이스이다. SAX는 주요 업체들의 주도로 상당히 빨리 만들어진 규약으로 W3C가 아닌 XML-DEV 메일링 리스트 회원들에 의해 공동 개발되었다. 1998년 5월에 SAX1.0이 발표되었으며, 2000년 5월에 외부 엔티티 접근, namespace를 지원하는 SAX2.0이 발표되었다. 현재 많은 파서와 애플리케이션이 DOM과 함께 SAX를 지원하고 있다. 이벤트 기반 방식인 SAX는 핸들러(Handler)를 SAX 파서와 함께 등록한 후 파서가 XML 문서를 파싱 하다가 XML 태그를 만나면 그 태그에 대응하는 함수를 호출하는 구조를 가진다. 이러한 특징 때문에 DOM처럼 객체 모델의 트리를 만드는 과정이 없어 SAX는 DOM을 사용하는 XML 문서 처리를 빠르면서 낮은 수준의 메모리를 사용하여 처리할 수 있다. 따라서 SAX를 사용하는 서블릿이나 네트워크 기반 프로그램에서는 XML 문서의 처리를 위한 XML 데이타의 전송과 수신에 있어서 빠르고 메모리를 적게 사용하기 때문에 DOM을 사용하는 것 보다 나은 성능을 기대할 수가 있다[4].

그림 5는 SAX 클래스와 인터페이스 계층도이다.

org.xml.HandlerBase 클래스는 DtdHandler, DocumentHandler, Entity Resolve, Error Handler 인터페이스를 지원하며, Throwable 클래스는 하위 Exception 클래스를 가진다.

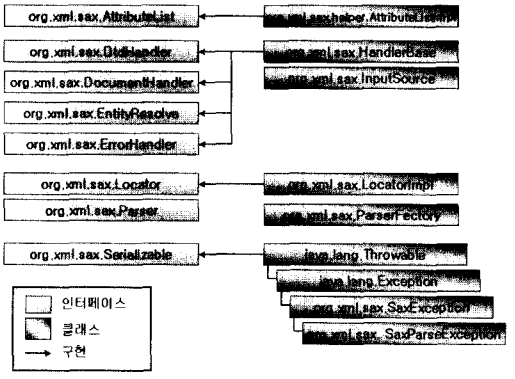

그림 5 SAX 클래스와 인터페이스 계층도

Visual Basic에 의한 SAX에서 개발자를 위해 제공하는 중요한 인터페이스와 메소드는 다음과 같다.

▶XMLReader(IVBSAXReader)
: 어플리케이션에서 파싱 기능 제공
- Content

▶ContentHandler(IVBSAXContentHandler)
: XML 문서 내용 제공
- StartDocument : 문서의 시작을 알림
- EndDocument : 문서의 끝을 알림
- startElement : 엘리먼트 시작을 알림.
- endElement : 엘리먼트 끝을 알림
- characters : DOM의 text 노드와 유사하게 character 데이타를 가진 것

▶ErrorHandler (IVBSAXErrorHandler)
: 파싱 과정에서 오류 제어
- Recoverable errors
- Fatal errors
- Warning

▶Locator (IVBSAXLocator)
: 이벤트가 발생한 라인과 컬럼 위치 제공

▶Attributes (IVBSAXAttributes)
: 문서내의 속성값에 접근 제공
- GetIndexFromName : 주어진 속성의 인덱스 값 반환
- GetLocalName : 속성의 로컬 이름 제공
- GetType : DTD 내에 선언된 속성 형 반환
- GetValue : 속성의 텍스트 값 제공
- Lenght : 속성의 총 개수를 반환하는 프로퍼티.

그림 6은 SAX를 사용하여 XML 문서를 읽고 폼창에 문서 태그를 출력하는 프로그램의 예와 결과를 보이고 있다.

ContentHandler 구현

```
Private Sub IVBSAXContentHandler_startElement(
strNamespaceURI As String,
strLocalName As String,
strQName As String,
 ByVal attributes As MSXML2.IVBSAXAttributes)
    Dim i As Integer
    Form1.TextBox1.text = Form1.TextBox1.text & "<" &
strLocalName
'속성 출력
    For i = 0 To (attributes.length - 1)
        Form1.TextBox1.text = Form1.TextBox1.text & " " &
attributes.getLocalName(i) & "=""" & attributes.getValue(i) &
"""
    Next
    Form1.TextBox1.text = Form1.TextBox1.text & ">"
'qu 엘리먼트 만나면 파싱 중단
    If strLocalName = "qu" Then
        Err.Raise vbObjectError + 1, "ContentHandler.start
Element", "Found element <qu>"
    End If
End Sub
Private Sub IVBSAXContentHandler_endElement(strNamespace
URI As String, strLocalName As String, strQName As String)
    Form1.TextBox1.text = Form1.TextBox1.text & "</" &
strLocalName & ">"
End Sub
Private Sub IVBSAXContentHandler_characters(text As String)
    text = Replace(text, vbLf, vbCrLf)
    Form1.TextBox1.text = Form1.TextBox1.text & text
End Sub
'문서 및 파서 경로
Private Property Set IVBSAXContentHandler_documentLocator
(ByVal RHS As MSXML2.IVBSAXLocator)
End Property
'문서의 끝
Private Sub IVBSAXContentHandler_endDocument()
End Sub
Private Sub IVBSAXContentHandler_endPrefixMapping
(strPrefix As String)
End Sub
'공백처리 이벤트
Private Sub IVBSAXContentHandler_ignorableWhitespace
(strChars As String)
End Sub
'PI 처리
Private Sub IVBSAXContentHandler_processingInstruction
(target As String, data As String)
        Form1.TextBox1.text = Form1.TextBox1.text & "<?"
& target & " " & data & ">"
End Sub
'엔티티 이벤트
Private Sub IVBSAXContentHandler_skippedEntity(strName As
String)
End Sub
문서의 시작 이벤트
Private Sub IVBSAXContentHandler_startDocument()
End Sub
Private Sub IVBSAXContentHandler_startPrefixMapping
(strPrefix As String, strURI As String)
End Sub
```

ErrorHandler 구현

```
'XML 문서 오류 발생시 폼 창에 오류 위치 및 메시지 제공
'fatal error 위치 및 내용
Private Sub IVBSAXErrorHandler_fatalError(
```

```
    ByVal lctr As IVBSAXLocator,
    ByVal msg As String,
    ByVal errCode As Long)

    Form1.TextBox1.text = Form1.TextBox1.text & "*** error
*** " & msg

End Sub
'오류, 경고 위치 및 내용
Private Sub IVBSAXErrorHandler_error(
    ByVal lctr As IVBSAXLocator,
    ByVal msg As String,
    ByVal errCode As Long)

End Sub

Private Sub IVBSAXErrorHandler_warning(
    ByVal lctr As IVBSAXLocator,
    ByVal msg As String,
    ByVal errCode As Long)

End Sub
```

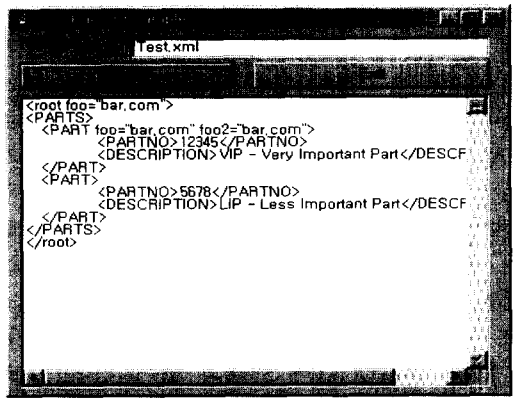

그림 6 SAX를 이용한 프로그래밍 결과

5. DOM과 SAX 차이점

DOM은 XML 문서의 구조와 저장된 정보를 기반으로 노드를 구성하고 계층적인 객체 모델의 형태로 접근이 가능하다. 즉, XML 문서를 수정할 때 직접 XML을 다룰 필요가 없고 메모리에 저장되어 있는 객체 모델을 다루면 된다. 결국 객체 모델을 표현한 노드와의 상호작용이 곧 XML 문서와의 상호작용이 된다. 그렇기 때문에 DOM을 지원하는 메소드를 사용하면 각 노드의 참조, 생성, 삭제, 수정과 그 내용의 변경을 쉽게 할 수 있다. XML 문서의 정보가 트리 형태로 표현되므로 트리를 가시화 하기 위해 GUI (Graphic User Interface)를 추가하게 되면 사용자의 입장에서는 문서의 구조와 정보를 한 눈에 파악할 수 있고 그것의 수정 또한 쉽게 할 수 있다. 반면 SAX는 노드의 트리 형태가 아닌 이벤트 기반으로 XML 문서의 정보에 접근하며 핸들러를 SAX 파서와 함께 등록한 후 파서가 XML 문서를 파싱 하다가 XML 태그를 만나면 그 태그에 대응하는 함수를 호출하는 구조를 가지고 있다. DOM과 달리 객체 모델의 트리를 생성하지 않기 때문에 문서를 읽고 쓰는 속도가 빠르지만 XML 데이타를 읽으면서 각각의 태그마다 파서가 이벤트를 생성하기 때문에 많은 프로그래밍 작업이 필요하다[2~4].

표 1은 DOM과 SAX에 대한 비교표이다.

표 1 DOM과 SAX 비교

API	DOM	SAX
접근 방법	· 트리 기반	· 이벤트 기반
장점	· 빈번한 수정, 저장 · XSLT 지원	· 메모리 효율이 좋다 · 사용이 쉽다 · 빠른 속도
단점	· 메모리 사용량 많다 · 속도가 느리다	· 구조 접근 어렵다 · 브라우저 지원 · 읽기 전용
적용 분야	· 구조적 접근이 필요할 경우 · 특정 부분으로 이동할 경우 · 문서 정보를 쉽게 파악하고자 할 경우	· 문서 일부분만 읽을 경우 · 유효성 처리 · 데이타의 변환 · 동일 오류 처리 · 엘리먼트 일부 추출

그림 7은 MSXML 파서의 DOM과 SAX의 XML 문서 처리 과정을 보이고 있다.

MSXML 파서는 DOM Level2와 SAX2를 지원하며 XML 파일 분석을 위해 DOM을 사용할 때 DOM은 완전한 문서 트리를 메모리에 구축을 하지만 SAX2는 문서를 이동한 후 분석 이벤트(예:새로운 구성 엘리먼트의 시작이나 종료)의 호출을 통지해준다.

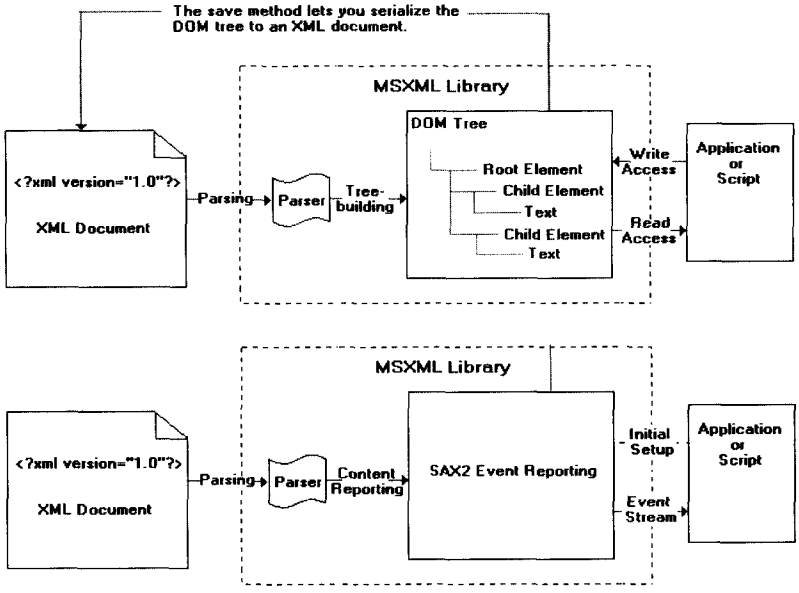

그림 7 MSXML 파서의 DOM과 SAX의 XML 문서 처리 과정

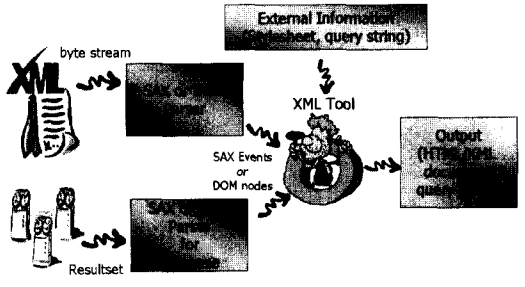

그림 8 DOM, SAX를 이용한 데이타베이스 및 파일 처리

6. 결 론

정보의 개방환경에서 컴퓨터를 이용한 문서의 처리와 이 기종 시스템간의 정보 교환이 날로 증대해져 가고 있으며 문서의 표현 방법과, 문서량의 증대로 사용자의 요구는 다양해져 가고 있다. 이에, 문서의 표준으로 등장한 XML의 중요성은 이미 널리 확대되었다. XML 문서의 사용이 빈번해 짐에 따라 문서의 사용범위가 커지고 새로운 시스템으로의 응용도 커지고 있다.

XML은 자기 서술적인 메타 언어로서뿐만 아니라 데이타로서 사용이 가능하며, 표현 정보와 내용이 분리되어 있어 새로운 시스템 응용간에 데이타의 교환에 대한 표준으로 빠르게 자리 잡아가고 있다. 대부분의 XML 응용들은 DOM, SAX와 같은 표준 API를 사용하고 있으며, 전자문서 처리 환경에 적용이 가능케 하며 응용 도구들이 데이타베이스나 XML 파일에 직접 접근하여 데이타를 스타일시트나 XML 기반의 질의 언어인 XQL(XML Query Language) 등의 처리로 다양한 결과를 얻을 수 있다(그림 8)[10~12].

XML에서 DOM과 SAX와 같은 XML 프로세서의 역활은 단순히 웹 상에서의 문서표현 및 전달을 위한 기능을 뛰어 넘어 웹에서 사용할 수 있는 메타 미디어의 유용성을 제공하며, DOM과 SAX를 이용하여 응용 프로그램을 구축하여 기존 응용 시스템에서 다루어졌던 데이타 및 문서의 많은 부분을 대체 할 것이며 새로운 응용 및 시스템을 도출해 차세대 인터넷 기술의 대표적인 요소기술로서 자리 매김과 함께 응용 프로그램 구축에 다양한 아이디어를 제공해 줄 것이다.

참고문헌

[1] W3C, Extensible Markup Language (XML) Version 1.0, http://www.w3.org/

TR/REC-xml, Feb. 10, 1998.
[2] W3C, Document Object Model Level 1, http://www.w3.org/TR/REC-DOM-Level-1/, Oct. 1, 1999.
[3] DOM1 reference, http://www.zvon.org/xxl/DOM1reference/reference/index.html
[4] David Megginson, "Simple API for XML(SAX)", http://www.megginson.com/SAX/index.html
[5] 이경하,이강찬,이규철, "XML프로그래밍",정보과학회 제18권 4호 pp4-12.
[6] XMLlab-XML 강좌-,http://www.xmlab.com/
[7] W3C, Document Object Model Level 2, http://www.w3.org/TR/2000/WD-DOM-Level-2-HTML-20001113/, Nov. 13, 2000.
[8] MSDN Online (XML), http://msdn.microsoft.com/xml/
[9] William j. Pardi, "XML in Action Web Technology", Microsoft Press.
[10] XML APIs for Database, http://www.javaworld.com/javaworld/jw-01-dbxml-p.html
[11] Mark Johnson, XML for the Absolute Beginner, http://www.javaworld.com/javaworld/jw-04-1999/jw-04-xml.html
[12] The W3C's proposed XSLT standard: http://www.w3.org/TR/1999/PR-xslt-19991008
[13] Sun's Java API for XML Parsing (JAXP), http://developer.java.sun.com/developer/earlyAccess/xml
[14] XT is a popular XSLT processor, http://www.jclark.com/xml/xt.html
[15] The GMD-IPSI XQL, http://xml.darmstadt.gmd.de/xql/index.html
[16] Boumphrey Frank, "XML Applications", WROX Press.

김 창 수

1996 배재대학교 전자계산학과(학사)
1998 배재대학교 전자계산학과(석사)
1999~현재 배재대학교 컴퓨터공학과(박사과정)
관심분야:멀티미디어 문서정보처리, SGML/XML, XML/EDI, XSLT
E-mail:sungu@markup.paichai.ac.kr

정 회 경

1985 광운대학교 컴퓨터공학과(학사)
1987 광운대학교 컴퓨터공학과(석사)
1993 광운대학교 컴퓨터공학과(박사)
1994~현재 배재대학교 컴퓨터공학과 교수
관심분야:하이퍼미디어/멀티미디어 문서정보처리, SGML/XML, HyTime, IETM, DSSSL/XSL, XML/EDI
E-mail:hkjung@mail.paichai.ac.kr