# XML 문서의 객체지향적 관리를 위한 XML DOM 소프트웨어의 설계 및 구현

선승상*, 박상윤**, 엄영익***
성균관대학교 전기전자 및 컴퓨터공학부
e-mail : {threes, bronson, yieom}@ece.skku.ac.kr

# Design and Implementation of XML DOM Software for the Object-oriented Management of XML Documents

Seung Sang Sun*, Sang Yun Park**, Young Ik Eom***
School of Electronics and Computer Engineering, Sungkyunkyan University

요 약

인터넷 사용자의 급증 및 인터넷 기반 응용 개발의 필요성은 기존 웹 환경의 기능성 및 구조성 등에 대한 확장을 요구하게 되었다. 이러한 배경 하에서 차세대 웹 문서를 위한 표준으로 XML이 탄생하게 되었고 DOM 인터페이스를 통한 XML 문서의 관리가 객체 지향 웹 기술을 위한 이슈로 부상하게 되었다. 본 논문에서는 차세대 웹 표준 언어인 XML 및 문서 객체화 기술인 DOM 등을 분석하고, XML 파서 모듈, DOM 처리 모듈 및 파서/DOM 연동 모듈들로 구성된 XML 문서의 객체화된 관리를 위한 XML DOM 소프트웨어를 설계하고 구현한 결과를 제시한다.

## 1. 서 론

인터넷 사용자의 급증 및 인터넷 기반 응용 개발의 필요성은 기존 웹 환경의 기능성 및 구조성에 대한 확장을 요구하고 있다. 그러나 기존 웹 표준 언어인 HTML은 이러한 요구 사항을 지원하기에 많은 한계성을 내포하고 있다. 이러한 배경 하에서 차세대 웹 문서를 위한 표준 언어로 XML(eXtensible Markup Language)이 탄생하였고[1][2], 구조화된 웹 문서의 조작을 지원하는 DOM(Document Object Model)이 소개되었으며[7][8][9], 현재 DOM 인터페이스를 이용한 XML 문서의 조작 및 관리가 차기 객체 지향 웹 기술의 이슈로 부상하게 되었다[2].

본 논문에서는 차세대 웹 표준 언어인 XML, 문서 객체화 기술인 DOM 및 DOM을 통한 XML 문서의 관리 기법 등을 분석하고, XML 파서 모듈, DOM 처리 모듈 및 파서/DOM 연동 모듈 등으로 구성된 XML DOM 소프트웨어의 설계/구현 사항과 구조/기능을 설명하며, 각 모듈들을 통합한 XML 문서의 용이한 관리를 위한 XML DOM 소프트웨어에 대한 평가 사항 등을 소개한다.

본 논문의 2장에서는 XML과 DOM 관련 기술의 개요 및 현황을 분석하고, 3장과 4장에서는 XML DOM 소프트웨어의 설계 및 구현 사항들을 기술한다. 5장에서는 XML DOM 소프트웨어 개발 환경과 시험 결과를 예시하고 6장에서는 요약 및 XML DOM 소프트웨어의 응용 분야 등을 소개한다.

## 2. 관련연구

### 2.1 XML 개요

HTML(HyperText Markup Language)은 SGML(Standard Generalized Markup Language)의 한 응용으로 문서의 내용을 표현하고 간단한 외양을 제어할 수 있으며, 현재까지 이식성과 편이성의 장점으로 인하여 웹 문서 저작의 보편적 언어로 사용되어 왔다. 그러나 HTML은 태그의 사용이 제한되어 자유로운 사용이 불가능하고, 문서들 간에 단일 링킹만이 가능하므로 다중 링킹을 통한 문서의 다중

연결을 지원하지 못하며, 문서 구조와 표현을 혼합하여 표현하는 한계점을 가진다.

이러한 배경 하에서, 1996년, 웹 표준화 단체인 W3C(World Wide Web Consortium) 산하 XML Working Group은 구조화된 DTD(Data Type Definitions)를 지원하며 문서 저작 시에 문서의 구조적 유효성을 검사할 수 있는 차기 웹 표준 언어인 XML을 정의하였다. XML은 HTML의 한계를 극복하고, SGML의 다양한 기능성을 수용하고 SGML의 문법 및 구조를 단순화하여 SGML의 장점을 유지하면서 SGML의 복잡성을 배제하였다. 따라서 XML은 일반화된 마크업을 사용한 태그 집합을 생성할 수 있고, 자기-서술적(self-describing) 문서 구조 정보의 표현이 가능하므로, 문서 구조 정보와 실제 문서간의 구조적 유효성(validity)을 검사할 수 있으며, 문서를 분할하여 부분적으로 조작할 수 있는 기능 등을 제공한다. 또한, HyTime을 기반으로 한 하이퍼링크 기능을 제공하기 위해 XLL(eXtensible Linking Language)과 연동하며, 문서의 스타일 정의를 위해 DSSSL을 기반으로 한 XSL(eXtensible Stylesheet Language)을 이용한다[1][13].

현재까지 XML 관련 소프트웨어로는 David Megginson에 의해 개발된 SAX, James Clark에 의해 개발된 expat, Edinburgh 대학에서 개발한 LT XML, IBM에 의해 개발된 XML4J, Tim Bray에 의해 개발된 Lark 및 Peter Murray-Rust에 의해 개발된 JUMBO 등이 있다[12].

### 2.2 DOM 개요

DOM은 DTD 또는 마크업 정보 등을 포함하는 구조적 문서를 대상으로 그 구성 요소들을 객체화하여 문서의 구조와 내용에 대한 접근 및 조작을 지원하는 언어 독립적인 트리 기반의 인터페이스를 제공한다[7][8][9]. DOM은 구조화된 문서들이 내부적으로 표현되고 조작되는 방법을 정의하는 추상적이고 개념적인 모델로서, XML 문서의 표현, 생성 및 조작을 지원할 수 있다.

DOM과 관련하여 1998년 10월에 W3C에 의해 문서에 대한 객체 정의를 표현하는 DOM(core) Level 1 명세서가 발표되었고, 1999년 3월에 DOM Level 1 명세서에 정의된 인터페이스를 기본으로 하여 CSS(Cascading Style Sheet) 모델, 이벤트 모델 및 질의(query) 등에 대한 인터페이스들을 추가한 DOM Level 2 명세서가 발표되었다[7][8][9].

DOM은 구조적 문서를 트리 기반의 계층적 구조로 재구성하여 이에 대한 접근 및 조작을 가능케 하는 표준 API(Application Programming Interface)를 제공함으로서 사용자들로 하여금 간단한 스크립트 언어나 프로그램을 사용하여 DOM 인터페이스를 통해 새로운 문서를 생성하고 조작할 수 있게 하며, 기존의 정적 문서를 동적으로 재구성하여 문서의 내용 및 구조를 수정할 수 있게 한다[7][8][9].

DOM Level 1에는 XML 및 HTML 문서의 접근 및 조작을 위한 기본 인터페이스(fundamental interface)들과 확장 인터페이스(extended interface)들이 정의되어 있는 core DOM 과 DOM HTML이 포함되며, DOM Level 2는 DOM Level 1을 확장한 인터페이스로서 이에는 문서의 스타일 및 조작에 대한 View, Stylesheets, CSS, Events, Traversal, Range 등이 추가되어 있다.

DOM은 문서의 구조를 표현하는 인터페이스들은 원시 데이터형인 Node 인터페이스를 상속함으로써, 문서 내의 엘리먼트들을 노드화 및 객체화하고, 이에 대한 조작 인터페이스를 제공한다. DOM은 XML 문서를 계층적 구조의 노드 집합으로 구성하는데, 즉, Document, Element, Attribute, Text, Processing Instruction, CDATASection 및 Comment 등의 XML 문서의 구성 요소들은 원시 객체인 Node를 상속받아 실체는 원시 데이터형 별로 다르지만 동일한 노드로 표현된다. XML 문서는 단일 Document 노드를 가지는 논리적 구조로 구성되는데, Document 노드는 부모 노드를 갖지 않는 최상위 루트 노드로서 여러 개의 자식 Element 노드들을 가질 수 있다.

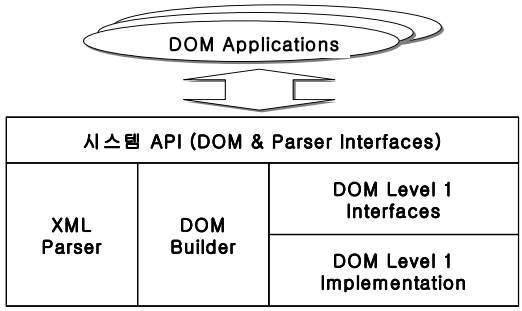
현재까지의 DOM 표준화 동향으로는 1997년 9월 발표된 DOM 요구 명세서, 1997년 12월 발표된 DOM Core Level 1 명세서, 1998년 10월 발표된 DOM Level 1 권고 명세서, 1998년 12월 발표된 DOM Level 2 요구 명세서 및 1999년 10월 발표된 DOM level 2 candidate 권고 명세서가 있다. 또한, 현재까지의 구현 제품으로는 IBM의 XML4J, Fourthought의 4DOM, Enno Derksen의 XML::DOM, InDelv의 InDelv Java DOM, Docuverse의 Docuverse DOM SDK 및 DataChannel의 DataChannel DOM Builder 등이 있다[14].

## 3. XML DOM 소프트웨어 설계

### 3.1 전체 모듈 설계

그림 1에서 예시하는 바와 같이 XML DOM 소프트웨어는 DOM Builder 블록, DOM Interfaces 블록, DOM Implementation 블록 및 XML 파서 블록 등으로 구성된 XML 파서 모듈, DOM 처리 모듈, 파서/DOM 연동 모듈 등으로 구성된다.

XML 파서 블록의 입력으로 전달된 XML 문서는 파서에 의해 어휘 분석 및 파싱 과정을 거쳐 파싱 테이블을 구성하고, 파싱 결과를 DOM Builder 블록에 전달하며, DOM Builder 블록은 전달된 토큰들을 문서의 구성 요소의 원시 데이터형에 따라 노드화 및 객체화하여 계층적 구조를 갖는 DOM 트리를 형성한다. 이 때, DOM Builder 블록은 파서와의 연동을 위해 연동 인터페이스를 파서에게 제공하고, 파서는 연동 인터페이스를 호출하여 파싱된 토큰을 전달할 수 있다. 그리고 DOM Interfaces 블록은 W3C의 DOM Core Level 1 명세서에서 정의하고 있는 인터페이스들을 포함하는데, DOM 응용이 DOM 인터페이스를 통해 내부적으로 객체화된 문서에 대한 생성, 접근 및 조작을 할 수 있도록 하는 API를 제공한다.




(그림 1) XML DOM 소프트웨어의 전체 모듈 구조

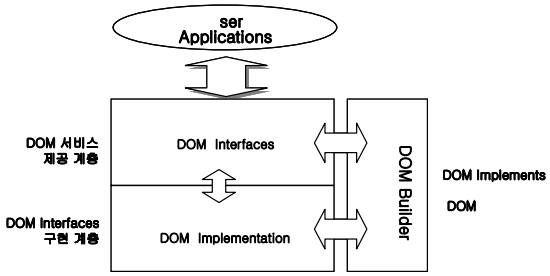
### 3.2 세부 블록 설계

그림 2는 DOM Interfaces, DOM Implementation 및 DOM Builder 블록에 대한 세부적인 구조를 예시한다.

DOM Interfaces 블록은 W3C의 DOM Core Level 1의 Java binding에 정의된 인터페이스들을 포함하는데, 이 인터페이스들은 사용자에게 XML 문서를 생성하고 조작할 수 있도록 하는 인터페이스를 제공한다.

또한, DOM Implementation 블록은 DOM Interfaces 블록의 인터페이스들의 구현 코드를 포함하고, DOM 처리 모듈의 확장을 위해 DOM Core Level 1 인터페이스에 포함되지 않은 어트리뷰트 리스트(ATTLIST) 선언, 엘리먼트 선언, XML 선언, DTD, Notation, Entity 등의 외부 경로 저장을 위한 인터페이스들을 포함한다.

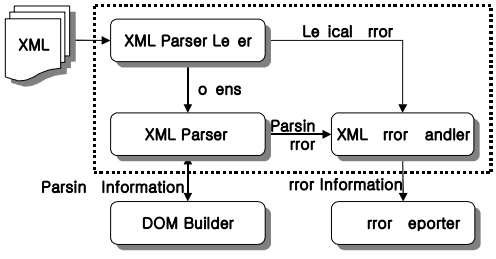
DOM Builder 블록은 파서와 DOM 처리 모듈간의 연동 인터페이스를 제공하고 파서로 부터 전달된 XML 문서의 파싱 정보를 바탕으로 DOM 트리를 생성한다. DOM Builder 블록은 XML 문서의 DTD와 document 간의 구조적 유효성 검사를 위하여 DTD 트리와 document 트리를 독립적으로 생성한다.




(그림 2) DOM 처리 모듈 세부 설계

그림 3에서 예시하는 바와 같이 XML 파서 모듈은 XML 파서 렉서(Lexer) 블록과 XML 파서 블록, 그리고 오류 처리기 블록으로 구성된다. XML 파서 렉서 블록은 XML 문서를 입력으로 받아들여 토큰을 추출하여 XML 파서 블록에 전달하고, XML 파서는 토큰을 파싱한 후 파싱 테이블을 생성한다.

XML 문서가 성공적으로 파싱되면 XML 파서는 DOM Builder 블록으로 파싱 상태 정보 및 토큰 스트림을 전달하고, 렉싱 및 파싱 과정 중에 에러가 발생할 경우, XML 오류 처리기에 의해 오류 정보를 오류 통보기(Error Reporter)에 전달한다.




(그림 3) XML 파서 세부 블록 설계

## 4. XML DOM 소프트웨어 구현

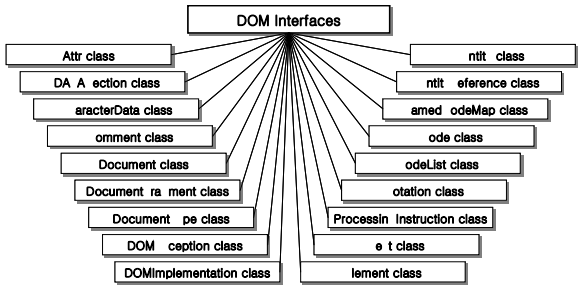
본 논문에서 기술한 XML DOM 소프트웨어는 Windows 95 운영체제와 UNIX 환경에서 JDK 버전 1.2 기준 JBuilder(version 3.0) 소프트웨어를 사용하여 구현되었으며, XML DOM 소프트웨어의 기능 검사를 위해 Java Swing 패키지를 이용하여 검사 소프트웨어를 구현하였다.

### 4.1 DOM 처리 모듈 구현

DOM 처리 모듈의 클래스들은 XML 문서에 대한 생성, 접근 및 조작 기능을 지원하기 위한 인터페이스를 제공하는 DOM Interfaces 블록의 클래스들을 상속한다. 또한 DOM Builder 블록과 연동하여 파싱 토큰의 원시 데이터형별로 노드화되어 DOM 트리의 구성 요소가 된다.

#### 4.1.1 DOM Interfaces 블록

그림 4에서 예시하는 DOM Interfaces 블록은 DOM Core Level 1에 정의된 바에 따라 XML 문서의 구성 요소별로 Attr, CDATASection, CharacterData, Comment, Document, DocumentFragment, DocumentType, DOMException, DOMImplementation, Entity, EntityReference, NamedNodeMap, Node, NodeList, Notation, ProcessingInstruction, Text, Element 클래스 등을 포함하고, 소프트웨어 사용자들로 하여금 객체화된 XML 문서의 내용을 생성, 접근, 및 조작할 수 있는 기능을 제공한다.




(그림 4) DOM Interfaces 블록의 상속 계층도

#### 4.1.2 DOM Implementation 블록

DOM Implementation 블록은 DOM Interfaces 블록의 클래스들을 상속하여 구현된다. 구현 클래스들 중에서 NodeImpl 클래스는 DOM의 원시 데이터형인 Node 인터페이스를 구현하는 클래스로서, 트리 구성을 위하여 파서로부터 전달된 토큰의 객체화 및 노드화 기능을 제공한다. NodeImpl 클래스를 상속하는 구현 클래스들은 기본적으로 자식 노드들을 가질 수 있으며, 자식 노드를 조작할 수 있는 메소드들을 포함한다. 또한, 구현상에서 java.util.Vector를 상속함으로서, 자식 노드들에 대한 삽입, 삭제 및 조작의 편이성을 제공한다. 그림 5에서는 DOM Implementation 블록의 클래스들간의 상속 관계를 나타내는 계층도를 예시한다.

대부분의 DOM Implementation 블록의 클래스들은 트리 구성을 위해 DOM의 원시 데이터형인 Node 인터페이스를 구현한 NodeImpl 클래스를 상속함으로써 노드화될 수 있다. DOM Implementation 블록의 클래스들은 DOM Core Level 1의 기본 및 확장 인터페이스들을 포함하는 DOM Interfaces 블록의 클래스들을 상속하여 구현되고, 부가적으로 XML 문서의 객체화를 위해 AttrDefImpl, AttlistDefImpl, ElementDefImpl, XMLDecl 및 ExternalID 등의 클래스들을 포함한다.

### 4.2 파서/DOM 연동 모듈 구현

파서와 DOM 처리 모듈간의 연동을 위한 파서/DOM 연동 모듈은 DomBuilder 블록으로 구성된다. 표 1은 DomBuilder 블록의 클래스들이다.

<표 1> 파서/DOM 연동 모듈 클래스들

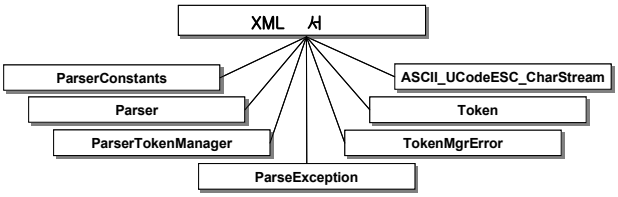
| 클래스 이름 | 설 명 |
|---|---|
| DomBuilder | TvList에 저장된 객체들로부터 노드화 및 객체화 수행 |
| TvList | 토큰들을 객체화 |
| TypedValue | 저장된 객체를 리스트화하여 토큰테이블 구성 |

파서/DOM 연동 모듈의 클래스들 중에서, 파서와 DOM 처리 모듈간의 연동을 위한 DomBuilder 클래스는 파서에게 연동 인터페이스를 제공하여, 전달된 토큰을 객체화하는 기능을 수행한다.

### 4.3 XML 파서 모듈 구현

그림 5에서는 XML 파서 모듈의 구현 클래스들을

예시한다.




(그림 5) XML 파서 모듈의 클래스들

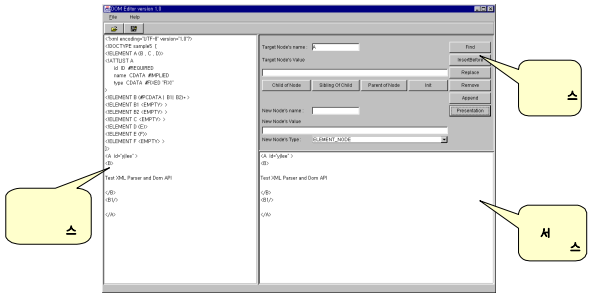
Parser 클래스는 XML 문서를 파싱하는 클래스로서, XML 문법의 BNF 표현에 준하여 XML 문서가 문법에 맞게 작성되었는지를 검사하는 기능을 수행하고, ParserConstans 클래스는 XML 문서를 파싱하는데 필요한 토큰들의 상수, 토큰의 이미지 등을 정의한 클래스이다.

Token 클래스는 XML 파서에서 입력되는 토큰 스트림에 대한 토큰의 종류, 토큰의 위치 및 이미지와 같은 특수 토큰 등을 정의하고, TokenInfo 클래스는 토큰에 대한 정보를 나타내는 토큰 ID, 이름, 값, 타입, 부모의 ID 등을 정의하며, ParserTokenManager 클래스는 Token 클래스와 TokenInfo 클래스에 정의된 토큰들을 이용하여 XML 문서에 존재하는 토큰들을 처리한다.

UnicodeEncoder 클래스는 입력되는 XML 문서를 유니코드로 인코딩하는 기능을 수행하고, ASCII_UCodeESC_CharStream 클래스는 토큰을 생성하기 위해 XML 문서를 스트림으로 처리해 주는 기능을 수행하며, ParseException 클래스는 파싱 과정에서 발생할 수 있는 예외 사항을 처리하는 클래스이다.

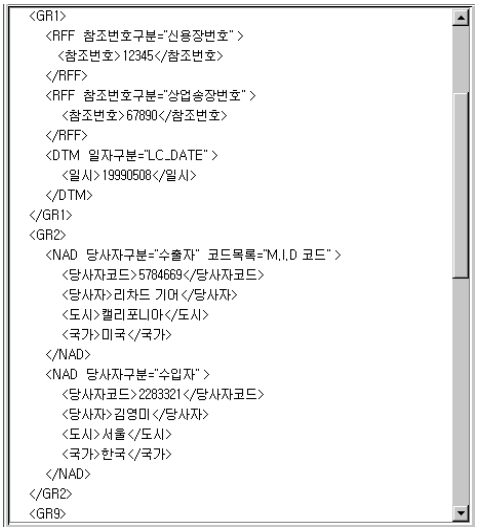
## 5. 시험 및 평가

그림 6에서는 XML DOM 소프트웨어의 기능 시험을 위하여 개발한 XML DOM 구동 소프트웨어의 실행 과정을 예시한다. XML DOM 구동 소프트웨어의 사용자 인터페이스는 파싱 결과 및 트리 검색 결과의 출력을 위한 파싱 결과 및 트리 검색 결과 인터페이스, DOM 처리 모듈을 실험하기 위한 DOM 트리 조작 인터페이스, 오류 보고 및 DOM 인터페이스 조작 결과의 출력을 위한 오류/문서 출력 사용자 인터페이스로 구성된다. 파싱 결과 및 트리 검색 결과 인터페이스는 입력된 XML 화일의 파싱 결과로 만들어진 DOM 트리를 검색하여, DOM 트리의 각 노드의 이름과 값을 출력하고, 트리 조작 인터페이스는 파싱 후 생성된 DOM 트리에 대한 조작이 가능하도록 DOM 응용에게 DOM 인터페이스 제공한다. 또한, 오류 및 문서 출력 사용자 인터페이스는 XML 문서의 파싱이나 DOM 인터페이스의 조작 과정에서 오류가 발생했을 경우에 발생하는 오류 및 DOM 인터페이스 구동 결과를 출력한다.



(그림 6) XML DOM 소프트웨어 실험 응용의 사용자 인터페이스

그림 7에서는 XML 문서의 파싱 결과 생성된 DOM 트리를 파싱 결과 및 트리 검색 인터페이스를 통해 검색 및 출력한 결과를 예시한다.

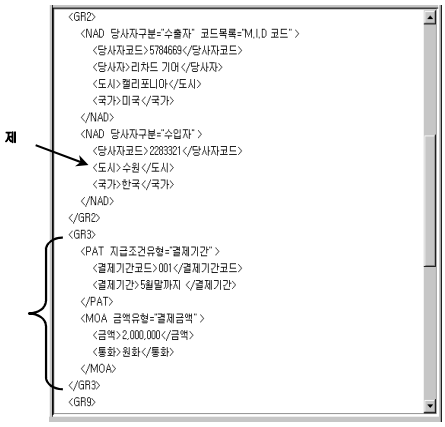


```
<GR1>
  <RFF 참조번호구분="신용장번호">
    <참조번호>12345</참조번호>
  </RFF>
  <RFF 참조번호구분="상업송장번호">
    <참조번호>67890</참조번호>
  </RFF>
  <DTM 일자구분="LC_DATE">
    <일시>19990508</일시>
  </DTM>
</GR1>
<GR2>
  <NAD 당사자구분="수출자" 코드목록="M,I,D 코드">
    <당사자코드>5784669</당사자코드>
    <당사자>리차드 기어</당사자>
    <도시>캘리포니아</도시>
    <국가>미국</국가>
  </NAD>
  <NAD 당사자구분="수입자">
    <당사자코드>2283321</당사자코드>
    <당사자>김영미</당사자>
    <도시>서울</도시>
    <국가>한국</국가>
  </NAD>
</GR2>
<GR9>
```

(그림 7) XML 문서의 파싱 및 DOM 트리 생성 결과

DOM 인터페이스의 정상적인 작동을 검사하기 위해, 그림 7에서 예시하는 문서의 트리를 조작한다. 조작 내용으로는 “PAT”과 “MOA” 엘리먼트를 가지는 “GRE3” 엘리먼트를 “GRE9” 엘리먼트 노드의 앞에 삽입하고, “PAT” 엘리먼트에는 “001”이라는 값을 가지는 “결제기간코드” 엘리먼트와 “5월말까지”이라는 값을 가지는 “결제기간” 엘리먼트를 삽입한다. “MOA” 엘리먼트에는 “2,000,000”이라는 값을 가지는 “금액” 엘리먼트와 “원화”라는 값을 가지는

“통화” 엘리먼트를 삽입한다. 또한 “GR2”의 두 번째 자식 노드인 “수입자” 어트리뷰트를 가지는 “NAD” 엘리먼트의 자식 노드 중에서 “김영미”라는 값을 가지는 “당사자” 엘리먼트를 삭제하고, “도시” 엘리먼트의 내용을 “수원”으로 대치한다. 그림 8은 구성된 트리의 조작한 결과를 에러 및 문서 출력 사용자 인터페이스를 통해 출력한 결과를 예시한다.




(그림 8) 삽입/삭제/대치 오퍼레이션의 수행 후의 DOM 트리 검색 결과

본 논문에서 기술한 XML DOM 소프트웨어는 트리 기반으로 구현하였고, 구조적인 문서를 트리 형태로 객체화하여, 생성된 문서의 내용에 대한 접근과 조작의 편이성을 제공한다. 또한 XML 문서의 DTD 부분을 트리 형태로 객체화할 수 있는 기능을 제공하므로, DTD를 이용한 레파지토리를 구현하는데 사용할 수 있다.

## 6. 결 론

인터넷 서비스의 확장 및 사용자 요구의 다양화는 전자 문서의 기능성, 구조성 등에 대한 확장을 요구하게 되었고, DOM 등의 문서 객체화 기술의 발전은 웹 문서 조작을 통한 웹 응용의 일반적인 개발 환경을 제공할 수 있게 되었다. 이러한 배경 하에서 웹 문서에 대한 차세대 표준으로써 XML이 탄생하게 되었으며, 문서 객체화 기술을 통한 DOM 응용의 개발이 차기 객체지향 웹 패러다임의 이슈가 되고 있다.

본 논문에서는 차기 웹 표준 언어인 XML과 문서 객체화 기술인 DOM을 분석하고, 이를 바탕으로 XML 문서의 객체화된 관리를 지원하는 XML 파서 모듈, DOM 처리 모듈 및 파서/DOM 연동 모듈로 구성된 XML DOM 소프트웨어의 설계 및 구현 사항을 소개하였다.

차기 웹 기술에서는 구조화된 웹 문서의 사용과 객체화된 웹 문서의 관리가 요구되므로, 웹 환경에서 객체화된 문서의 모델링, 조작, 교환 및 적재를 지원하는 문서 객체화 기술의 사용은 필연적이라고 할 수 있다. 본 논문에서 소개한 DOM 소프트웨어는 웹을 통한 문서의 객체 서비스를 가능하게 하는 목적으로 제작되었으며, XML 편집기, 문서 객체화를 통한 데이터베이스 적재 툴 등 다양한 응용 분야에 적용시킬 수 있다.

## 7. 참고문헌


[1] W3C, "Extensible Markup Language (XML) 1.0 : W3C Recommendation“, http://www.w3.org/TR/1998/REC-xml-19980210.html, Feb. 1998.
[2] OASIS, "XML Web Page," http://www.oasis-open.org/cover/xml.html, 1999.
[3] C. F. Goldfarb and P. Prescod, The XML Handbook, Prentice Hall, 1998.
[4] M. Leventhal, D. Lewis, and M. Fuchs, Designing XML Internet Applications, Prentice Hall, 1998.
[5] C. S. Horstmann and G. Cornell, Core Java 1.2, Vol. 1, Prentice Hall, 1999.
[6] D. Chang and D. Harkey, Client/Server Data Access with Java and XML, John Wiley & Sons Pub., 1998.
[7] W3C, "DOM Level 1 Specification : W3C Recommendation", http://www.w3.org/TR/1998/PR-DOM-Level-1-19980818/, Aug. 1998.
[8] W3C, "DOM Level 2 Specification : W3C Working Draft", http://www.w3.org/TR/1998/WD-DOM-Level-2-19981228/, Dec. 1998.
[9] W3C, "DOM Level 2 Specification : W3C Working Draft", http://www.w3.org/TR/1999/WD-DOM-Level-2-19990923/, Sep. 1999.
[10] H. Maruyama, K. Tamura, and N. Uramoto, XML and Java, Addison Wesley, May 1999.
[11] S. McGrath, XML by Example, Prentice Hall, 1998.
[12] W3C, "Extensible Markup Language (XML)," http://www.w3.org/XML/, Feb. 1998.
[13] N. Pitts-Moultis and C. Kirt, XML Black Book, Coriolis, 1999.
[14] W3C, "Document Object Model (DOM)," http://www.w3.org/DOM/, Dec. 1999.
[15] M. Morrison, et. al., Java 1.1, Sams.net, 1997.
[16] R. Eckstein, M. Loy, and D. Wood, Java Swing, O'reilly and Associates Inc., 1998.
[17] K. Sankar, et. al., Java 1.2 Class Library, SAMS, 1999.