論文
3-8-13

XML 문서를 위한 DTD 저작 도구의 설계 및 구현 (Design and Implementation of DTD Authoring Tools for XML Documents)

김 현 주*
(Hyun-Ju Kim)

요 약

XML은 전자도서관, 전자상거래, 웹 응용 등의 분야에서 다양하게 응용되고 있다. 이러한 XML 문서에 대한 생성, 관리, 검색 등에 대한 연구는 XML 응용 시스템 개발에 있어서 필수적인 항목이다.

본 논문에서는 XML 문서의 문서 구조정보를 편리하게 관리해주는 XML DTD 편집기 도구를 설계하고 구현하였다. 본 논문에서 제안한 저작도구는 사용자의 편의성, 문서 DTD 구문 규칙에 대한 신뢰성 보장 및 문서 구조의 가시성 등의 특징을 가지고 있다.

ABSTRACT

XML is a markup language which has been accepted in various fields such as digital libraries, electronic commerce, and web applications. Research for creation, storage, management, and retrieval of XML documents is essential to develope XML application systems. This paper presents design and implementation details of powerful and convenient DTD authoring tools for XML documents. The design principles are authoring convenience, semi-automatic creation of valid and reliable document DTD by systematic guidance to reduce the possibility of syntax errors, and visualization of document structures.

1. 서론

오늘날 컴퓨터 기술의 발전과 네트워크의 급속한 확산에 따라 기존의 종이를 이용한 정보매체에서 전자문서로의 변화가 가속화되고 있다. 한편 현재 널리 사용되고 있는 워드프로세서의 문서 포맷은 독자적인 프리젠테이션 방식을 사용하기 때문에 이기종간 호환성 등의 문제점을 갖고 있다.

따라서 인터넷상의 문서를 생성·관리하기 위한 표준 포맷의 필요성을 절실히 느끼게 되었으며, 이러한 표준화 작업의 일환으로 ISO(the International Organization for Standardization)를 주축으로 SGML (Standard Generalized Markup Language)[2]을 표준 포맷으로 제안하였다[1]. 하지만 SGML은 사용법이 매우 복잡하고 웹에서의 활용성이 떨어져 XML (eXtensible Markup Language)이 대두되게 되었다.

* 정회원 : 진주산업대학교 컴퓨터공학과 전임강사

논문접수 : 2002. 7. 23.
심사완료 : 2002. 8. 14.

XML은 W3C(World Wide Web Consortium)에서 제안한 것으로서 웹 상에서 SGML의 사용을 보다 쉽고 간단하게 하기 위해 고안되었다. XML은 문서 타입들의 정의를 쉽게 지원하고, SGML로 정의된 문서들의 저작과 관리를 용이하게 하고, SGML과 HTML 양자간의 상호 운용성과 용이한 구현을 위해 고안되었다. 또한 XML은 인터넷에서의 복잡하고 구조화된 문서에 대하여 문서자료의 저장, 관리 및 검색 등을 용이하게 할 수 있을 뿐만 아니라, 인터넷 기반 응용 시스템인 전자상거래, 전자 도서관, 가상대학 등의 구축에서 중요한 역할을 할 것으로 예상된다. 따라서 XML 문서의 효과적인 관리를 위한 다양한 연구가 필요하다.

최근의 XML 관련 연구는 응용 도구 개발과 XML 문서 관리 시스템 개발 등의 두가지 분야로 구분할 수 있으며, 본 논문은 XML 문서 저작 시스템 개발에 대한 연구이며, 이러한 XML 문서 저작 시스템은 DTD(Document Type Definition) 편집기 개발, XML 문서 편집기 개발, Style Sheet 편집기 개발 등으로 나누어진다. 첫 번째로 DTD 편집기 개발 분야이다. 이는 XML 문서의 구조를 편집할 수 있는 기능을 지원한다. XML 문서 구조는 DTD 규칙에 따라 작성된다. 이들의 표현 양식은 EBNF (Extened Backus-Naur Form)로 구성된다. 따라서 EBNF를 효과적으로 편집할 수 있는 도구가 DTD 편집기이다. 두 번째는 XML 문서 편집기 개발 분야이다. 이는 DTD에서 정의된 문서구조와 태그를 사용하여 XML 문서 인스턴스를 생성시키는 도구이다. 마지막으로 Style Sheet 문서 편집기 개발 분야이다. 이는 XML 문서 인스턴스를 브라우저에 표현하기 위한 규칙을 편집하는 도구이다.

본 논문에서 제안하고 개발한 XML DTD 문서 저작 시스템은 다음의 특징을 가진다. 첫째 편의성이다. 사용자가 각각의 편집기가 유사한 사용자 인터페이스를 제공하여 각각의 툴을 쉽게 익힐 수 있도록 하고 각 문서의 문법을 정확히 몰라도 쉽게 문서를 만들 수 있게 한다. 둘째 가시성이다. 사용자가 현재 작업하고 있는 일을 한눈에 알 수 있는 인터페이스를 제공하며 할 수 있는 작업과 할 수 없는 작업을 구별하여 보여준다. 셋째 신뢰성이다. 사용자가 만든 각각의 문서가 문법에 맞는지 체크하여 발생할 수 있는 문법 오류의 가능성을 최소화하도록 한다.

2. 관련 연구

2.1 XML 문서 모델링

XML 문서를 보는 관점에 따른 모델에는 이벤트 기반의 모델인 SAX(Simple API for XML)[6]와 문서의 각 구성요소를 개체 트리로 간주하는 DOM (Document Object Model)[13]등으로 구분할 수 있다.

첫 번째로 SAX를 이용하는 문서 편집기[5]이다. 이는 XML 문서를 하나의 긴 문자열로 간주하며, 각 노드에 대한 이벤트를 발생시켜 문서 구조 정보를 생성한다. 이때 생성된 구조정보는 구문 분석 트리를 이용하지 않고 발생된 이벤트에 대한 구조 정보만을 생성한다. 즉 문서 편집기는 필요한 노드의 구조정보만을 유지하기 때문에 자원을 효율적으로 사용할 수 있다. 그러나 XML 문서 구조가 수정되거나 응용프로그램용 XML 문서를 공유하고자 할 때, 문서 전체의 구문 정보를 유지하지 못해 일부분 수정된 문서의 구조 정보 유지가 어렵다.

두 번째로 DOM을 기반으로 하는 문서 편집기이다. 이는 XML 문서를 하나의 트리 구조로 간주하여 문서의 구조정보를 관리하며, 직관적으로 사용방법이 매우 쉽다. 또한 문서 인스턴스를 쉽게 수정·관리할 수 있다. 그러나 DOM 방식은 XML 전체 문서를 메모리에서 유지해야 하므로 수정 처리에 대한 오버헤드가 발생한다.

2.2 상용 XML 저작시스템

최근에 많은 XML 문서 저작 시스템들이 개발되어 출시되어 있다. 이들 XML 문서 저작 시스템들은 DTD 편집기, XML 편집기, CSS 편집기 등으로 구분되어 있다. 본 논문에서는 DTD 편집기만을 대상으로 국내·외의 상용 모델들에 대해 고찰해본다.

먼저, 국외 DTD 편집기는 IBM사의 Visual DTD [9], Microstar사의 Near & Far Designer v3.0[10]가 있고, 국내 DTD 편집기는 (주) 언어 기술의 DQuilter v1.0[11], 다산 기술의 TagFree 2000 DTD Editor[12] 등이 있다.

국외 DTD 편집기인 IBM사의 Visual DTD는 엘리먼트를 트리로 표현하여 각 엘리먼트와 애트리뷰트의 편집이 가능하며 DTD의 구조를 시각적으로 보여주는 기능을 가지고 있다. 또한 기존의 DTD 문서가 가지고 있던 화이트 스페이스를 그대로 유지하며 유효성 체크가 가능하다. 하지만 Visual DTD에서는 문서구조와 상관없는 노테이션과 엔티티도 트리로 표현한다. 문서 구조의 전체가 아니라 선택된 엘리먼트에서부터의 부분적인 구조를 보여주어 사용자가 문서 구조의 루트를 지정하기 전에는 전체 문서의 구조를 알기가 어려운 점이 있다.

국내 DTD 편집기인 (주) 언어 기술의 DQuilter v1.0은 DTD 문서의 구조 트리를 지원하여 현재 작업중인 DTD 문서의 구조를 쉽게 알 수가 있다. 또한 텍스트 편집을 지원하며 한글 태그가 가능하다는 잇점이 있다. 하지만 각 구성요소(엘리먼트, 애트리뷰트, 엔티티, 노테이션)의 추가 및 수정에 관한 인터페이스가 어려워 사용법이 쉽지 않고 텍스트 편집을 지원함에도 불구하고 화이트 스페이스가 유지되지 않는다.

3. XML DTD 저작 시스템 설계

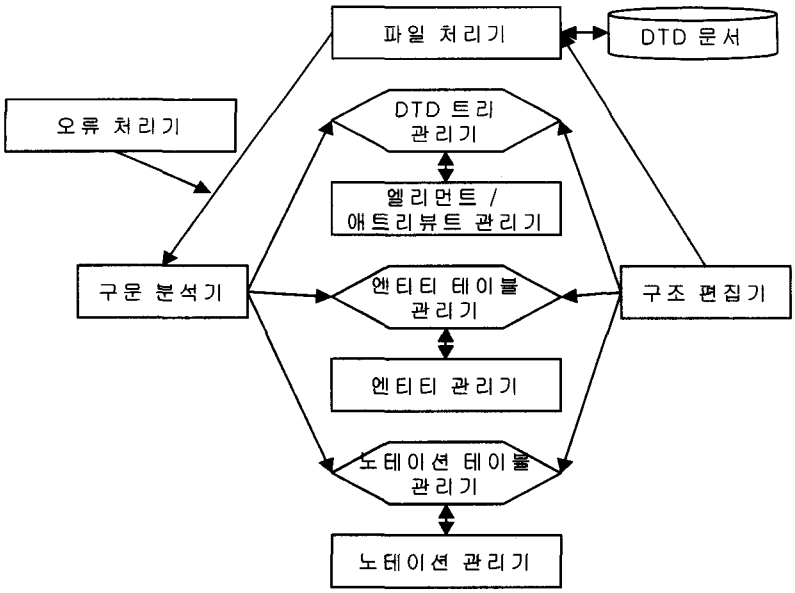
3.1 시스템의 설계 정책

본 연구에서 개발한 시스템은 편의성, 신뢰성, 가시성을 가지도록 구현한다. 첫 번째는 사용자의 편의성이다. 이는 사용자로 하여금 XML DTD를 만드는데 있어 각각의 문법을 정확히 알지 못하더라도 문서를 저작할 수 있도록 하며 편리한 사용자 인터페이스를 제공하는데 목적을 둔다. 두 번째로 신뢰성이다. 이는 사용자가 작성한 각각의 문서로부터 발생 가능한 문법 오류를 최소화하는데 목적은 둔다. 마지막으로 가시성이다. 이는 사용자가 각각의 DTD를 저작하는데 있어 자신의 작업 내용과 작업 효과를 한눈에 알 수 있도록 하는데 목적을 둔다.

3.2 시스템 전체 구성도

[그림 1]은 DTD 편집기 구성도이다. DTD 편집기는 파일을 열고, 저장하는 파일 관리기, DTD 문서의 구문을 분석하는 구문 분석기, 구문 분석과정에서의 오류 검정과 오류 처리를 위한 오류 처리기, 구문 분석과정에서 DTD 문서의 구조를 분석하여 엘리먼트, 애트리뷰트, 엔티티, 노테이션을 추출하여 이들을 각각 관리하는 엘리먼트/애트리뷰트 관리기, 엔티티 관리기, 노테이션 관리기가 있으며 DTD 문서를 편집할 수 있는 편집기로 구성된다.

파일 관리기는 이미 만들어진 DTD 문서를 입력받거나 새로운 DTD 문서를 작성하거나 또는 기존의 DTD 문서를 수정한 후에 저장하는 기능을 제공한다. 입력된 DTD 문서는 구문 분석기를 통해 문서의 구조를 파악한다. 이때 오류가 있을 경우에는 오류 처리기가 이를 처리하게 된다. 해당 DTD 문서에 오류가 없다면 엘리먼트, 애트리뷰트, 엔티티, 노테이션을 각각의 관리기가 관리하게 된다. 엘리먼트/애트리뷰트 관리기는 DTD 문서로부터 엘리먼트와 애트리뷰트를 추출하여 이들에 대한 정보를 저장한다. DTD 트리 관리기는 엘리먼트/애트리뷰트 관리기로부터 저장되어진 정보를 이용해 사용자에게 트리 형태로 DTD 문서에 대한 구조를 보여준다. 엔티티 관리기와 노테이션 관리기는 DTD 문서로부터 엔티티와 노테이션을 추출하여 각각 필요한 정보를 관리한다. 엔티티와 노테이션은 사용자에게 테이블 형태로 보여지는데 이는 엔티티 테이블 관리기와 노테이션 테이블 관리기가 담당한다. 편집기는 사용자 인터페이스에 해당하며 DTD 문서를 편집하는 기능을 제공한다.



[그림 1] DTD 에디터 전체 시스템 구성도
[Fig. 1] System Architecture of DTD Editor

3.3 자료구조

DTD 문서 편집기는 DTD 문서의 엘리먼트와 애트리뷰트, 엔티티, 노테이션을 관리하기 위해 각각의 자료구조를 필요로 한다. 엘리먼트 객체와 애트리뷰트 객체는 DTD 문서를 트리로 표현했을 때 트리의 각 노드가 된다. 엘리먼트와 애트리뷰트의 자료구조는 <표 1>과 같다.

노드 이름은 트리로 표현했을 때 각 노드의 이름이 되는데 엘리먼트 이름, 애트리뷰트 이름, 그룹, #PCDATA를 가질 수 있다. 노드 타입은 각 노도의 타입을 의미하며 엘리먼트인지 애트리뷰트인지 그룹인지 내용모델인지를 구별해 준다. 내용 모델은 이 객체가 내용 모델일 경우에 값을 가지면 ANY, EMPTY, 내용 모델을 가질 수 있다. 연결자는 엘리먼트의 발생 순서를 의미하며 |나 ,를 가진다. 발생 지시자는 엘리먼트의 발생 회수를 의미하며 +, *, ?를 가진다.

애트리뷰트 타입은 CDATA, ID, IDREF, IDREFS, ENTITY, ENTITIES, NMTOKEN, NMTOKENS, NAME TOKEN GROUP, NOTATION을 가진다. 애트리뷰트 값은 애트리뷰트가 가질 수 있는 값이 정해져 있을 때 Enumeration으로 나열되어진다. 애트리뷰트 초기 플래그는 #IMPLIED, #REQUIRED, #FIXED의 값을 가지거나 아무런 값을 가지지 않는다. 애트리뷰트 초기 플래그가 #FIXED일 때 애트리뷰트 기본값을 가지게 된다. 그룹 연결자는 그룹안에서의 연결자를 이야기하는데 역시 |나 ,를 가진다. 그룹 발생 지시자는 그룹안에서의 발생 지시자를 의미하며 +나 * 또는 ?를 가진다. 내용모델 발생 지시자는 내용모델 안에서의 발생 지시자를 의미하며 +나 * 또는 ?를 가진다.

<표 1> 엘리먼트/애트리뷰트 클래스의 자료구조

<Table 1> Data Structure of the Element/Attribute Class

항 목	기 능
노드 이름	엘리먼트 이름, 애트리뷰트 이름, 그룹, #PCDATA
노드 타입	엘리먼트, 애트리뷰트, 그룹 내용모델
내용 모델	ANY, EMPTY, 내용모델
	|, ,
발생 지시자	+,*,?
애트리뷰트 타입	CDATA, ID, IDREF, IDREFS, ENTITY, ENTITIES, NMTOKEN, NMTOKENS, NAME TOKEN GROUP, NOTATION
애트리뷰트 값	애트리뷰트가 가질 수 있는 값의 리스트
애트리뷰트 기본 플래그	#IMPLIED, #REQUIRED, #FIEXED
애트리뷰트 기본값	애트리뷰트가 기본으로 가지는 값
그룹 연결자	|, ,
그룹 발생 지시자	+, *, ?
내용 모델 발생 지시자	+, *, ?

<표 2>는 엔티티 객체의 자료구조이다. 엔티티 객체는 DTD 문서의 엔티티 자료에 대한 정보를 저장하게 된다. 엔티티의 이름은 해당 엔티티의 이름을 가진다. 엔티티가 인자 객체/일반 객체 선언은 해당 엔티티가 인자 객체(parameter entity)인지 일반 객체(general entity)인지 구별하는 값을 가지게 된다. 인자 객체는 DTD 문서 안에서만 사용되는 엔티티를 의미하고 일반 객체는 XML 문서에 사용되는 엔티티를 의미한다. 엔티티 내부/외부 객체 선언은 해당 엔티티가 내부 엔티티(internal entity)인지 외부 엔티티(external entity)인지 구별하는 값을 가지게 된다. 내부 엔티티는 엔티티의 치환용 텍스트가 DTD 문서 안에 정의되어 있음을 의미하고 외부 엔티티는 엔티티의 치환용 텍스트가 외부 파일에 정의되어 있음을 의미한다. 외부 시스템(SYSTEM) 엔티티 값과 외부 퍼블릭(PUBLIC) 엔티티 값은 해당 엔티티가 외부 엔티티로 선언되었을 경우 그 해당 엔티티가 SYSTEM 또는 PUBLIC 중 하나로 선언되었냐 하는 것에 따라 달라지게 된다. 엔티티와 노테이션이 함께 사용될 경우 노테이션이 노테이션 이름에 저장된다. 엔티티 선언부는 엔티티를 테이블로 표현하기 위한 정보가 저장된다.

<표 2> 엔티티 클래스의 자료구조

<Table 2> Data Structure of the Entity Class

순 번	항 목
1	엔티티 이름
2	엔티티 인자 / 일반 객체 선언
3	엔티티 내부 / 외부 객체 선언
4	내부 엔티티 값
5	외부 시스템 엔티티 값
6	외부 퍼블릭 엔티티 값
7	노테이션 이름
8	엔티티 선언부

노테이션은 <표 3>과 같은 자료 구조를 가진다. 노테이션 객체는 DTD 문서의 노테이션 자료에 대한 정보를 저장하게 된다. 노테이션 이름에는 노테이션 이름이 저장된다. 노테이션 타입은 해당 노테이션이 SYSTEM 또는 PUBLIC으로 선언되었냐는 구별해 준다. 노테이션 값에는 해당 노테이션이 가질 수 있는 값이 저장된다.

<표 3> 노테이션의 자료구조

<Table 3> Data Structure of the Notation Class

번 호	항 목
1	노테이션 이름
2	노테이션 타입
3	노테이션 값

3.4 동작 원리

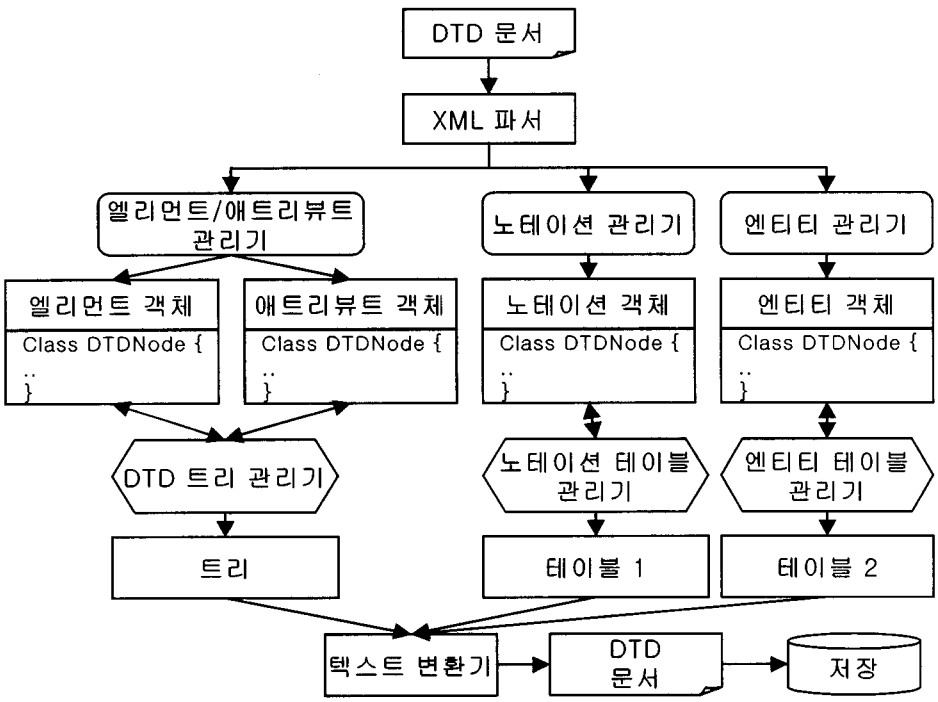
[그림 2]는 DTD 문서 편집기의 동작 원리를 나타낸다. DTD 문서는 XML 파서를 통해 문서의 구조를 분석하여 엘리먼트, 애트리뷰트, 노테이션, 엔티티로 나누어진다. 엘리먼트와 애트리뷰트는 각각 엘리먼트/애트리뷰트 관리기에 의해 DTDNode 객체에 필요한 정보들이 저장되고 DTD 트리 관리기가 이를 각각의 노드로 하여 트리를 만들어낸다.

새로운 엘리먼트 추가시에는 새로운 DTDNode 객체가 생성되어지고 필요한 정보를 입력한 후에 트리의 한 노드로서 현재 트리에 삽입되게 된다.

예를 들어 다음의 DTD가 있다고 가정하자.

<!ELEMENT book (title, author+)>

엘리먼트/애트리뷰트 관리기는 이 DTD에 대하여 하나의 DTDNode 객체를 만들고, 이를 <표 4>와 같은 구조로 저정한다. 그리고 이 객체를 DTD 트리 관리기로 넘기면 DTD 트리 관리기는 DTDNode 개체로부터 DTD 트리를 생성한다.



[그림 2] DTD 에디터의 내부 구조도

[Fig. 2] Internal Structure for DTD Editor

<표 4> DTD 노드 객체 생성과정(1)

<Table 4> Creation Process a DTDNode Object(1)

노드 이름	book	title	author
노드 타입	엘리먼트	내용모델	내용모델
내용 모델	내용모델		
연결자	,		
발생 지시자	1		
애트리뷰트 타입			
애트리뷰트 값			
애트리뷰트 기본 플래그			
애트리뷰트 기본값			
그룹 연결자			
그룹 발생 지시자			
내용 모델 발생 지시자		1	+

계속해서 title이란 새로운 엘리먼트를 추가하고 이 엘리먼트에 KOR이라는 기본값을 가지는 language라는 애트리뷰트를 추가하고자 한다고 가정하자. 이를 DTD로 나타내면 다음과 같다.

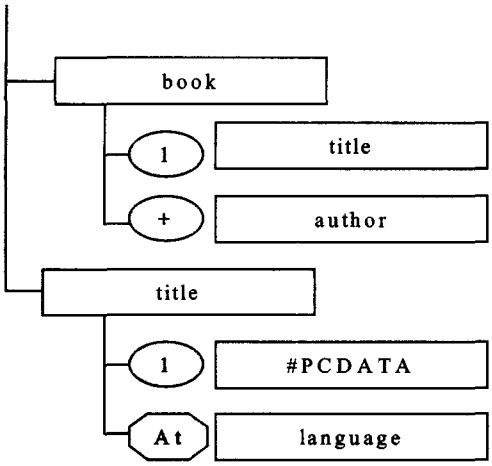
```
<!ELEMENT title (#PCDATA)>
<!ATTLIST language CDATA #FIXED KOR>
```

엘리먼트 관리기는 새로운 DTDNode 객체를 만들고 <표 5>와 같은 구조로 만든다.(<표 5>는 <표 4>의 열의 연속이다.) DTD 트리 관리기는 <표 5>를 바탕으로 [그림 3]과 같은 트리를 생성한다.

<표 5> DTD 노드 객체 생성과정(2)

<Table 5> Creation Process of a DTDNode Object(2)

노드 이름	…	title	#PCDATA	language
노드 타입	…	엘리먼트	내용모델	애트리뷰트
내용 모델	…	내용모델		
연결자	…			
발생 지시자	…	1		
애트리뷰트 타입	…			CDATA
애트리뷰트 값	…			
애트리뷰트 기본 플래그	…			#FIXED
애트리뷰트 기본값	…			KOR
그룹 연결자	…			
그룹 발생 지시자	…			
내용 모델 발생 지시자	…		1	



[그림 3] DTD 노드 객체로부터의 트리 생성 예제
[Fig. 3] Building a Tree from DTDNode Objects

[그림 3]은 <표 5>로부터 생성되어진 트리이다. [그림 3]에서 볼 수 있듯이 DTDNode 객체로부터 노드 이름은 각 트리의 노드 이름이 되고 노드의 타입과 발생지시자가 필요에 따라 출력되어 DTD 문서의 구조를 알 수 있는 트리를 만든다.

문서의 구조와 상관없는 노테이션과 엔티티의 경우는 노테이션 관리기와 엔티티 관리기가 각각 StrNotaion 객체와 StrEntity 객체에 정보를 저장하고 이를 각각의 테이블 관리기가 테이블 형태로 표현한다.

예를 들어 다음의 엔티티 경우를 살펴보자.

<!ENTITY % dash "-">
<!ENTITY amp "&">

엔티티 관리기는 StrEntity 객체에 <표 6>과 같은 모습으로 각 값들을 저장한다.

노테이션의 경우는 엔티티와 거의 유사한 과정을 거치게 된다..

현재 작업 중인 DTD 문서의 소스를 보거나 저장하기 위해서는 텍스트 변환기를 통해 DTD 문서로 변환해야 한다. 먼저 노테이션 테이블과 엔티티 테이블부터 텍스트로 변환이 된다. 엔티티의 경우 <표 6>에 나타난 것처럼 StrEntity 객체의 엔티티 이름부터 노테이션 이름까지를 순회하면서 각각의 필드로부터 값을 추출하며 DTD 문법에 맞는 엔티티 문장을 생성해낸다. 노테이션도 엔티티와 동일한 방법으로 StrNotation 객체를 순회하면서 DTD 문법에 맞는 노테이션 문장을 생성해낸다. 엔티티와 노테이션이 끝난 다음에는 엘리먼트와 애트리뷰트를 만들어 낸다. 엘리먼트와 애트리뷰트의 경우에는 DTDNode 객체로부터 만들어진 DTD 트리를 전위 운행(Preorder Traverse)하면서 각 노드의 정보를 Enumeration class형의 객체에 저장한 이를 다시 하나하나 토큰(token)하여 DTD 문법에 맞는 문장으로 변환한다.

<표 6> 엔티티 객체 생성과정
<Table 6> Creation Process of Entity Object

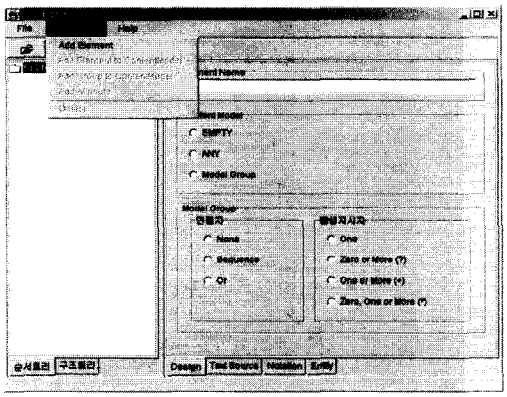
엔티티 이름	dash	amp
엔티티 인자 / 일반 객체 선언	인자 객체	일반 객체
엔티티 내부 / 외부 객체 선언	내부 객체	내부 객체
내부 엔티티 값	-	&
외부 시스템 엔티티 값		
외부 퍼블릭 엔티티 값		
노테이션 이름		
엔티티 선언부	% dash "-"	amp "&"

4. 구현

XML 문서 저작 시스템 구현에 사용된 OS는 MS-Windows 98이며 언어는 Java 1.2를 이용하였다. 구현을 위한 프로그래밍 툴은 JBuilder 3.5를 사용하였다. DTD 문서와 XML 문서의 구문 분석을 위해서 XML4J v1.1.9 Paser를 CSS 문서의 구문 분석을 위해 CSS2Parser가 사용하였다. 그리고 DOM은 DOM API v1.0을 사용하였다.

4.1 DTD 문서 편집기 구현 화면

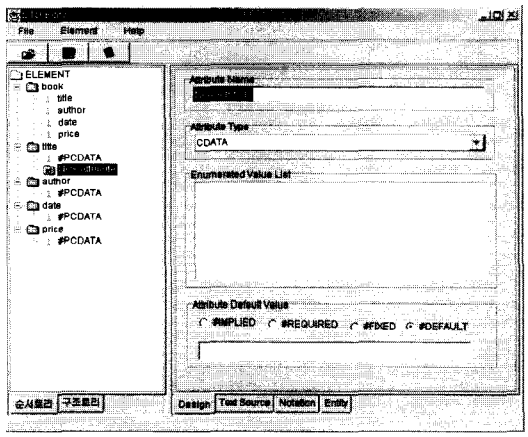
다음은 구현된 DTD 문서 편집기의 모습이다.



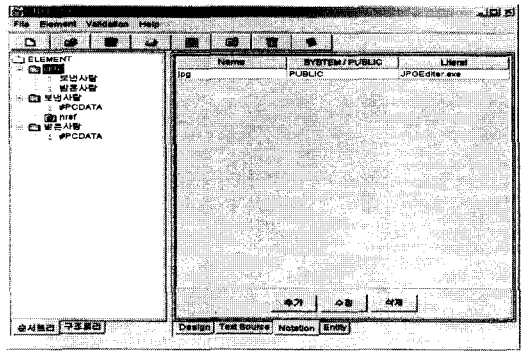
[그림 4] 엘리먼터 추가
[Fig. 4] Inserting an Element

DTD 문서 편집기에서 엘리먼트를 삽입하는 화면이다. 사용자는 메뉴로부터 Add Element를 선택하고 삽입하고자 하는 엘리먼트의 이름만을 입력하고 나머지는 선택만 함으로서 엘리먼트를 삽입할 수 있는 편리성을 가진다. 선택하는 것에 따라 다음 선택 가능한 메뉴가 활성화 되거나 비활성화 됨으로써 사용자에게 해야할 일을 손쉽게 알려주며, 입력 내용은 왼쪽의 순서 트리창에 그대로 반영되게 함으로써 사용자가 자신이 하는 작업의 결과를 바로 알 수 있는 가시성을 가진다.

[그림 5]는 애트리뷰트를 삽입하는 모습이다. 사용자는 엘리먼트 삽입 때와 같이 필요한 부분만을 입력하고 나머지는 선택을 통해서 필요한 값들을 입력하게 된다. 선택하는 항목에 따라 다음 입력 가능한 메뉴가 역시 활성화 되거나 비활성화 되어 나타나게 된다. 애트리뷰트 입력시 애트리뷰트의 타입에 따라 애트리뷰트가 가질 수 있는 값을 입력 할 필요가 있을 경우 Enumerated Value List창을 제시하여 필요한 값을 입력한다. 입력 내용은 마찬가지로 왼쪽 트리 메뉴에 바로 적용되어 나타나게 된다.

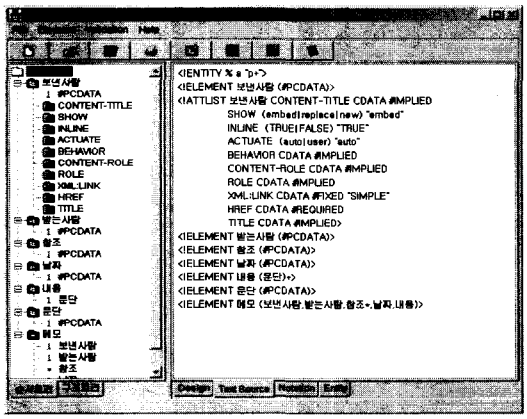


[그림 5] 애트리뷰트 추가
[Fig. 5] Inserting an Attribute



[그림 6] 노테이션 리스트 보기
[Fig. 6] Viewing the Notation List

[그림 6]은 노테이션 목록을 보는 화면이다. 노테이션과 엔티티의 삽입시에도 엘리먼트나 애트리뷰트와 같이 사용자는 최소한의 입력 내용만 입력하고 나머지는 선택을 통해 편집을 하게 된다. 노테이션과 엔티티는 문서의 구조와 상관이 없기 때문에 트리 구조로 표현하지 않고 [그림 5]와 같이 테이블 형태로 나타나게 된다. 이를 통해 DTD 문서내에서 문서의 구조와 비구조 부분을 분리하여 사용자에게 문서 구조를 명확히 보여준다.



[그림 7] 텍스트 소스 보기

[Fig. 7] Viewing the Text Source

[그림 7]은 작성한 DTD 문서의 소스를 확인하는 모습이다. 사용자는 이전의 [그림 5]에서 [그림 7]까지의 일련의 과정에 따라 트리 구조를 통해 DTD 문서를 작성한다. 작성된 DTD 문서의 내용을 확인하기 위해서는 그림 하단의 Text Source 탭을 누르면 오른쪽 창을 통해 생성된 DTD 문서를 확인할 수 있다. 텍스트에 대해서는 직접적으로 편집을 할 수 없고, 단지 트리를 통해서만 DTD 문서를 작성하고, 텍스트 모드는 문서를 확인하는 기능만을 가진다. 이러한 설계 방향은 간단한 선택을 통해 오류없이 완전한 DTD 문서의 생성을 유도하기 위해서이다.

5. 비교분석 및 평가

본 장에서는 개발된 시스템을 국내외 문서 편집기와 비교한 결과를 제시한다. 다음의 <표 7>는 국내외 DTD 문서 편집기와 본 시스템을 비교한 표이다.

본 논문에서 설계하고 개발한 XML DTD 저작 시스템은 사용자에게 보다 쉽고 간편하게 문서를 저작 할 수 있도록 하는데 중점을 두었다. XML 문서 저작시에 필요한 DTD 문서 편집기를 패키지 형태로 제공한다. 각각의 편집기들은 문서의 구조를 명확히 보여 주며 문서의 저작 방법을 시각화하여 제시하여 사용자들이 각각의 문법을 정확히 모르더라도 신뢰성 있는 문서를 저작 할 수 있는 방법을 제시한다. 그러나 기존의 여러 제품들과 비교해 볼 때 한글 태그를 지원하지 않아 이 점을 개선한 필요가 있다.

<표 7> DTD 에디터 비교분석

<Table 7> Comparison of DTD Editors

구 분		장 점	단 점
국외	Visual DTD	· XML 스키마 지원 여러개의 XML 문서로부터 병합된 유효한 DTD 문서 자동 생성	· 한글 태그 지원 불가
국내	DQuilter	· 한글 태그 지원 · DTD 문서 구조 트리 지원	· 엔터터 처리 미흡 · 인터페이스의 편리성 부족
	본 시스템	· DTD 문서 구조 트리 지원 · 향상된 사용자 인터페이스	· 한글 태그 지원 불가

6. 결론 및 향후 과제

앞으로 XML은 더욱 많은 분야에서 활발하게 사용될 것이 예상되고 있다. 따라서 이런 XML 문서를 손쉽게 제작할 수 있는 응용 프로그램이 필요하다.

본 논문에서는 효율적인 XML 문서 저작 시스템 중의 일부분인 DTD 편집기를 패키지 형태로 설계, 구현하였다. 향후 연구과제로는 이러한 기술을 기반으로 XML 문서 편집기, CSS 편집기 등을 추가로 개발하여 하나로 묶는 패키지 형태의 XML 저작 시스템에 대해 연구 개발할 계획이다. 다만 본 논문에서는 문서의 구분 분석을 위해 기존의 파서를 그대로 사용함으로써 한글 처리를 위한 완벽한 방법이 제공되지 못한다는 단점이 있다.

또한 완벽한 한글 태그 지원과 다양한 인코딩 지원을 위한 파서의 개발이 병행되어야 하며 개발된 XML 문서 저작 시스템을 기반으로 연계되어 작동할 수 있는 XML 문서 저장 시스템의 설계와 구현에 대한 연구가 필요하다.

※ 참고문헌

[1] 이경호, 고승규, 백승욱, 변창원, 장은영, 최윤철, "DSSSL에 기반한 SGML 문서 편집기의 설계 및 구현", 한국정보과학회 논문지(C) 제 4권 제 6호, 1998.

[2] ISO 8879, Information Processing -- Text and Office Information Systems -- Standard Generalized Markup Language(SGML), ISO (http://www.iso.ch), 1986.

[3] W3C Group, "eXtensible Markup Language (XML)", 6 October 2000, http://www.w3.org/TR/2000/REC-xml-20001006.

[4] 정보통신부, "창조적 지식 기반 국가 건설을 위한 정보화 Vision", 1999, http://web.mic.go.kr/ck21/kor/part2/index211.html.

[5] 조영환, 강지훈, "SAX를 이용한 XML 문서 편집기", 한국정보과학회 학술발표 논문집 (B) 제 25권 1호, 1999.

[6] Megginson Technologies, "The Simple API for XML(SAX 1.0)", 11 May 1998, http://www.megginson.com/SAX/SAX1/index.html.

[7] 김규평, 이종학, "문서병합 기능을 갖는 XML 문서 편집기의 설계 및 구현", 한국정보과학회 학술발표논문집(B) 제 27권 1호, 2000.

[8] 채진석, 나홍석, 백두권, "데이터 레지스트리를 이용한 자동화된 XML DTD 생성 방법", 한국정보과학회 학술발표논문집(B) 제 26권 1호, 1999.

[9] IBM, Visual DTD, 1999, http://www.alphaworks.ibm.com.

[10] Microstar Software Ltd, "Near & Far Designer", Version 3.0, 1998, http://www.microstar.com.

[11] (주)언어기술, "Quilter DTD Editor", Version 1.0, 1999, http://www.ibase.co.kr.

[12] 다산기술, "TagFree 2000", 2000, http://www.dasannet.com.

[13] W3C Group, "Documnet Object Model(DOM) Leve 1 Specification", 1 October 1998, http://www.w3.org/TR/REC-DOM-Level-1.

김 현 주

1988년 경상대학교
전산통계학과(이학사)
1990년 숭실대학교
전자계산학과(공학석사)
2000년 경상대학교
전자계산학과(공학박사)
1994년 ~ 1997년
제일정밀공업(주) 연구원
2000년 ~ 2002년
경남정보대학
컴퓨터정보계열 교수
2002년 ~ 현재
진주산업대학교
컴퓨터공학과 전임강사
관심분야 : 정보검색, XML,
디지털 도서관,
웹프로그래밍
E-mail : khj@jinju.ac.kr