XML 기반의 대용량 유사 문서 편집기/변환기 구현

Implementation of an XML-Based Editor/Transformer for Large Volume of Similar Documents

황인준(Een-Jun Hwang)*

초 록

최근 웹이 보편화되면서 웹은 이제 거대한 정보의 보고로서 중요한 의미를 가지게 되었다. 현재 웹 상에 존재하는 많은 문서들은 HTML로 작성되어 있다. HTML은 간단하고 배우기가 쉬운 반면, 고정된 태그 등으로 정보 검색에 있어서 비효율적이다. 이러한 단점을 보완하기 위해 XML이 제안되어 현재 다양한 응용에 활용되고 있다. XML은 HTML에 비해 구조적이고 또한 정보의 의미를 적절하게 표현할 수 있어 정보 검색에 있어서 훨씬 효과적이다. 이러한 추세에 맞추어 본 논문에서는 XML 문서를 다양하고 효과적으로 생성하고 관리하는 XML 문서 관리기를 제안한다. 시스템의 특징으로는 첫째, 문서의 양식을 반영하는 폼을 제공하여 XML을 잘 모르는 일반 사용자도 쉽게 새로운 문서를 작성할 수 있게 하였으며 둘째, 이미 존재하는 비슷한 구조를 가지는 HTML 문서에 대해서는 자동적인 XML 문서로의 변환을 제공하여 기존의 HTML 문서도 XML를 통해 관리와 검색을 가능하게 한다. 마지막으로 GUI 상에서 문서의 DTD를 편집할 수 있게 하여 DTD 작성을 용이하게 하였다.

ABSTRACT

With its recent popularity, Web is now considered as a huge repository of information. Most documents on the web have been created using HTML(Hyper Text Markup Language). Even though HTML is simple and easy to learn, it has several features that are obstacles to the efficient information retrieval. XML(eXtensible Markup Language) can provide a solution to such problems and in fact, has already been used in many applications. XML is a standard markup language for exchanging data on the web. It can describe a document structure freely by defining its DTD, which enables efficient integration and retrieval of data on the web. In this paper, we propose a versatile and efficient XML document manager. Its features include (i) form-based XML editor that enables easy creation of new XML documents, (ii) automatic document converter that can transform HTML documents with similar structure into XML documents automatically, and (iii) GUI-based DTD editor.

키워드 : XML 변환기, 공통 패턴, 폼, HTML, DTD, 편집기

XML Converter, Common Pattern, Form, HTML, DTD, Editor

* 아주대학교 정보통신 전문대학원

1. 서 론

인터넷과 정보 처리 기술의 발달은 컴퓨터를 통한 조직 간의 정보 교환을 활성화시켰고 많은 조직들이 더욱 효과적인 정보 교환을 위해 비용과 노력을 기울이고 있다. 조직간의 정보 교환에 있어서 커다란 장애의 하나는 조직들이 사용하는 문서의 형태가 서로 다르다는 것이다. 이러한 문제점을 해결하기 위해 최근에 웹상에서 통합된 문서의 전송이 가능하도록 설계된 언어인 XML(eXtensible Markup Language)을 이용한 여러 가지 시도가 진행되고 있다.

XML[1,2,3]은 SGML과 HTML(HyperText Markup Language)의 단점을 극복하고자 1997년 W3C에 의해 표준으로 제정된 언어이다. 기존의 웹 응용에서 일반적으로 사용되는 HTML은 가능상의 제약과 고정적인 태그의 사용 때문에 효과적인 정보의 검색이나 복잡한 응용에 적합하지 못하고, SGML의 경우 문서 기술에 필요한 태그의 생성이나, 문서 내용, 내용 구조의 정의 및 데이터의 교환과 같은 면에서 장점이 있지만, 구조가 복잡하고 사용이 어려우며 SGML 전체를 지원하는 소프트웨어의 개발이 용이하지 않다. XML은 SGML의 범용적인 기능성과 웹 문서의 구조적 표현성을 지원할 뿐 아니라 SGML의 단점이던 사용의 복잡성을 최소화하여 사용자로 하여금 문서상에 사용될 태그 세트와 속성을 필요에 따라 자유롭게 정의할 수 있게 한다. 또한 문서의 내용과 형식을 분리하여 문서에는 그 구조와 의미에 관한 정보만 들어가며, 요소들을 꾸미는 부분은 별도의 스타일시트로 정의하여 결과적으로 하나의 문서를 다양한 형태로 표현할 수 있게 한다.

XML 문서는 웹 브라우저 상에서의 데이터 표시 기능 이외에도 다양한 종류의 응용 프로그램과도 통합될 수 있는 범용적인 데이터베이스라고도 할 수 있다. XML이 가진 확장성과 편리함 때문에 현재 많은 웹 문서들이 XML로 작성되고 있다. 하지만 대개 문서의 작성은 트리 구조의 사용자 인터페이스를 기반으로 하며 XML에 대한 사전 지식을 필요로 한다. 더욱이, 이미 다수의 문서들이 HTML로 만들어져 있기 때문에 웹 데이터를 통합하고 효과적으로 관리하기 위해서는 HTML 문서를 XML 문서로 변환하여 통합 처리할 필요가 있다.

이에 본 논문에서는 문서의 양식을 반영하는 폼을 통해 새로운 XML 문서의 작성을 용이하게 하며 동시에 유사한 구조를 가지는 대량의 기존 HTML 문서를 XML 문서로 자동 변환할 수 있는 XML 문서 관리기를 개발하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 XML 문서의 생성 및 변환을 위한 관련 연구들을 살펴보고, 3장에서는 폼을 기반으로 한 XML 문서 작성, 유사성에 기반한 HTML 문서의 XML 자동변환, 그리고 DTD 편집기에 대해서 기술하며 4장에서는 결론 및 향후 연구 과제에 대해 언급하겠다.

2. 관련 연구

XML은 데이터의 교환을 고려한 언어로서 HTML과는 달리 문서의 내용에 관련된 태그들을 사용자가 직접 정의할 수 있으며, 그 태그들을 다른 사람들도 사용할 수 있다. 이러한 데이터로서의 XML의 특성은 이기종간(heterogeneous) 시스템 환경에서 문서의 상호 교환 뿐 만 아니라 데이터의 통합 및 처리 자동화 등에 많은 장점을 가진다. 또한 XML은 문서의 구조와 표현을 분리하여 XML 문서는 문서의 구조와 의미에 관한 정보만을 기술하며, 그 요소들의 표현은 스타일시트 언어인 XSL을 통해 XML 문서의 내용을 다양하고 세련된 형태로 표현한다. 또한 XSLT(eXtensible Stylesheet Language for Transformations)를 사용하여 HTML로 변환하면 일반 웹 브라우저를 통해 XML 문서를 검색할 수 있다. 현재 이러한 XML의 목표에 부합되는 많은 XML 응용들이 개발되고 있으며 그와 연동하여 효과적인 문서 작성을 위한 다양한 형태의 XML 편집기가 개발되고 있다.

XML 기반의 저작 도구는 WWW을 위한 XML 편집기, XML 문서의 분석과 검증을 위한 XML 파서, 작성된 XML 문서의 디스플레이를 위한 XML 브라우저, XML 문서의 출력 관리를 위한 스타일시트 관리기 등으로 구성된다[13]. 편집기는 크게 XML 편집, DTD 편집, XSL 편집 등의 기능을 지원하며 기본적으로 문서의 수정, 삭제, 저장 기능과 추가적으로 문서에 대한 유효성 검사 기능을 가진다. 기존의 편집기는 앞에서 말한 세 가지 편집 기능을 통합적으로 지원하는 환경이 아니라 편집하고자 하는 문서에 따라 편집기를 별도로 구현하였다. 이렇게 구현된 편집기는 XML과 DTD, XSL 사이의 연관성을 고려하지 않고 단순히 XML 문서에 대한 구조화된 트리 정보를 사용자에게 보여줌으로써 텍스트 형태의 편집 기능을 제공한다. 즉, XML 문서에 대한 내부 모델링 처리만 할뿐 편집한 문서를 스타일에 적용하여 화면에 표시해 주는 엔진이 없어, 실제로 브라우저에서 어떻게 보이는지 확인하기 위해서는 사용자가 웹 브라우저를 통해서 해당 문서를 확인해 보아야 한다.

W4F(WysiWyg Web Wrapper Factory)[6]는 특정 웹에 있는 자원을 쉽고 간편하게 래퍼(wrapper)로 만들 수 있도록 해주는 도구로서 반자동(semi-automatic)적으로 웹 문서의 데이터에서 자바 클래스로 생성하며 사용자는 이 클래스 파일을 변형하여 다른 응용에 재사용 할 수 있다. 하지만 W4F는 HTML 문서를 파싱해서 그 태그를 자바 객체로 매핑시키는 정도에만 그치며 사용자가 필요한 용도에 따라 가공해야 한다는 단점이 있다.

XWRAP[7,8]은 웹 문서를 위한 래퍼 프로그램을 반자동적으로 생성해주는 시스템이며 비 구조적인 HTML 문서를 입력 받아 구조적인 XML 문서를 생성한다. 또한 XML로 제작되지 않은 웹 문서를 XML 문서로 표현할 때 불필요한 부분에 대한 제거를 통한 내용 선택 기능을 제공한다. 자바로 구현된 이 프로그램의 장점은 사용자 인터페이스가 제공되고 래퍼의 생성이 용이하다는 것이다. 그러나 XML이 아닌 문서의 구조를 분석하는데 있어서 미흡한 면이 있고, 사용자가 개입

하여 변환을 할 경우 다수의 문서를 위한 공통적인 DTD 생성이 어렵다는 단점이 있다.

Taniar와 Jiang[9]은 효율적인 웹 정보의 검색을 위해 평면적인 구조의 HTML 문서에서 구조적인 XML 문서를 생성하는 시스템을 제안하였다. 이들이 제안한 방법은 웹 페이지의 분석을 통해 구조 정보를 얻어내고 그 구조 정보를 이용해 스키마를 생성한 후 매핑 도구를 사용하여 XML 문서를 생성하는 것이다. 하지만 변환해야 하는 HTML 문서마다 사용자가 개입을 해야 하는 단점이 있다.

특정한 프로그램에서 필요한 정보를 웹 문서로부터 추출하기 위한 다른 방법으로 Huck[10]은 여러 개의 독립적인 웹 문서로부터 정보를 다룰 수 있는 기법을 제안했다. 내부적으로 이질적인 문서 모델을 규칙적인 객체 모델로 변형시킬 수 있으며 여러 개의 문서를 통합하는 뷰를 정의하여 사용자는 데이터를 다룰 수 있다. 이질적인 문서 형태를 규칙적인 객체 모델로 변형시키기 때문에, 특정 문서에만 있는 정보를 추출하고 싶을 때는 문서 모델을 고려하여 추출 규칙을 정해야 하는 단점을 지닌다.

위에서 설명한 시스템들은 비구조적인 문서의 전체나 일부를 XML 문서로 변환하는 방법을 제안하고 있다. 변환 과정이나 방법은 다를 수 있으나 변환의 기본 골격은 비구조적인 문서의 구조를 분석하고 분석한 구조 중에 데이터를 중심으로 사용자 개입을 통해 XML 문서로의 매핑을 이루는 접근법을 취하고 있다.

본 논문에서는 기업이나 관공서와 같이 정형화된 폼을 따르는 다량의 문서를 다루는 환경에서 XML 기반의 환경으로 전환할 새로운 XML 문서의 생성이나 기존 문서의 XML 변환을 효과적으로 수행할 수 있는 시스템을 개발하고자 한다. XML 문서의 변환에서는 공통적인 패턴을 갖는 HTML 문서의 구조를 분석하고 그에 관련된 경로 정보를 인식하여 자동적인 변환을 수행한다. 새로운 문서를 생성하는 경우 문서의 양식에 따르는 form을 제공하여 기존 문서 작성 방법과 유사하게 XML 문서를 작성할 수 있게 한다.

3. 시스템 구현

3.1 폼/트리 기반의 XML 문서 생성

XML 문서 편집기의 주요 기능은 XML이 정의하는 문법에 맞는(well-formed) 문서나 DTD에 맞는 유효한 문서를 오류 없이 사용자가 쉽게 제작할 수 있게 하는 것이다. 지금까지 다수의 XML 문서 편집기가 개발되었지만 아직 많은 문제점을 보여주고 있다. 구체적으로 기존의 XML 문서 편집기는 효율적인 검색 기능을 제공하지 못하며, XML 문서의 구조적 정보를 표현하기 위해 단순한 트리 구조 형태의 인터페이스를 바탕으로 XML 문서를 제작한다. 이것은 사용자가 어느 정도 XML에 대한 지식이 있어야 문서를 작성하고 편집할 수 있다는 것을 의미한다. 그러나, XML 문서의 사용이 보편화되고 일반화되는 현실에 비추어 보면, 사용자가 보다 쉽게 사용할 수 있고, XML에 대한 지식이 없이도 문서를 쉽게 작성할 수 있는 환경을 제공하여야 한다.

기존의 XML 문서 편집기가 트리 구조를 기반으로 하는 것은 XML 문서의 구조적인 특성과 관계가 있다. 하지만 그러한 문서 편집 환경은 XML에 능숙하지 못한 사용자에게는 적지 않은 부담이 된다. 이러한 문제점을 해결하기 위해 본 논문에서는 XML 문서를 작성하기 위해 기존의 트리 구조를 기반으로 하는 인터페이스 환경뿐 만 아니라 문서의 폼을 기반으로 하는 인터페이스를 함께 제공한다.

트리 기반의 XML 문서 작성 모듈은 DTD가 존재하지 않는 문법에 맞는 XML 문서나 입력 DTD 구조에 유효한 XML 문서를 제작하게 하며, 폼 기반의 XML 문서 작성 모듈은 템플릿(template) XML 문서와 그 문서의 디스플레이 정보를 가진 XSL 문서를 이용하여 새로운 XML 문서를 제작할 수 있게 한다. 폼을 이용하여 XML 문서를 작성하기 위해서는 XSLT 처리기, HTML 브라우저, XML 생성기와 같은 구성 요소가 필요하다. XSLT 처리기는 템플릿 XML 문서와 XSL 문서를 바탕으로 사용자의 입력 폼에 해당하는 HTML 문서를 생성하여 HTML 브라우저에 전달하며, 브라우저는 입력 폼을 사용자에게 보여주게 된다. 끝으로 XML 생성기는 입력 폼을 통해 얻은 데이터를 바탕으로 해당 XML 문서를 생성하게 된다.

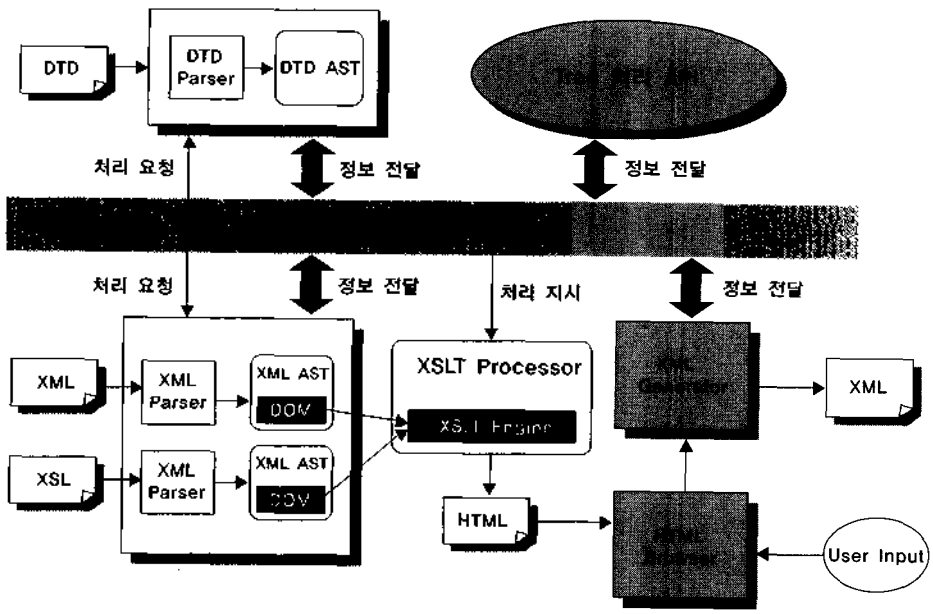
입력 폼에서 사용자의 입력을 받기 위한 입력 양식으로는 버튼, 체크 박스, 패스워드, 라디오, 텍스트, 선택 등이 있으며, 입력된 정보의 전송과 다시 입력을 지원하기 위한 전송(submit) 과 재설정(reset)이 포함된다. 구현에 있어서 Java1.3을 사용하였고 사용자 인터페이스는 Swing을 사용하였다. 또한 XML 문서의 유효성 검사를 위한 파서는 JAXP1.1을 사용하였고, XML 문서를 검색하기 위한 XSLT 엔진으로는 Xalan-java v1.0을 사용하였다.

본 절에서 제안하는 XML 문서 생성 시스템은 크게 DTD 파서, XML 파서, XSLT 처리기, HTML 브라우저, XML 생성기, 트리 처리 API 모듈로 구성되어 있다. 그림 1은 시스템을 구성하는 각 요소와 실제 XML 문서 처리를 위한 요소들 간의 상호 관계 및 전체 흐름을 보여주고 있다.

트리 처리를 위한 API는 트리의 생성, 저장, 추가, 삭제, 확대, 축소, 선택 등을 제공하며, XML 파서를 통해 생성된 DOM은 트리 처리 API를 통해 응용상에서 사용되는 XML AST(Abstract Syntax Tree)로 만들어진다.

DTD AST의 생성 : 기존의 XML 파서가 DTD를 이용한 XML 문서의 검증 기능을 제공하지만, DTD 문서 자체는 XML 문서가 아니기 때문에 XML 파서를 통해 DTD에 대한 실제적인 정보를 제공받는 데 한계가 있다. 하지만, DTD를 가지는 XML 문서의 효율적인 생성을 위해서는 DTD에 대한 구조적 정보가 필요하다. 따라서 본 논문에서는 별도의 DTD 파서를 통해 DTD 문서를 파싱하여 DTD AST라는 트리를 생성한다. DTD 문서를 통해 DTD 트리를 생성하여 구조적 정보를 가지는 XML 문서 상에서 엘리먼트를 추가하거나 삭제할 때 DTD의 구조 정보를 얻을 수 있도록 하였다.

XML 파서 : 파서는 XML 문서의 유효성을 검증하거나 XSL 문서를 파싱할 때 사용된다. 응용 상에서 접근되는 XML AST와 XML



〈그림 1〉 XML 문서 처리 전체 흐름도

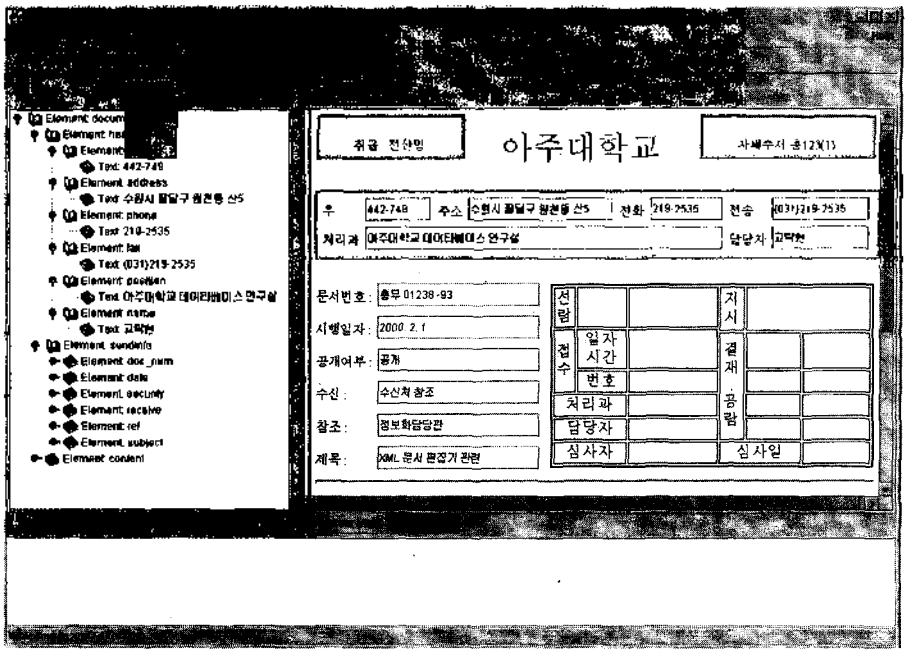
파서 사이의 중간 객체 모델로는 DOM을 사용하였고 XML AST는 Java의 Swing 컴포넌트인 JTree를 사용하여 구현하였다. 즉, XML 파서를 통해 파싱된 XML 문서는 DOM으로 만들어지고, 이러한 DOM은 사용자 응용 상에서 브라우징 하기 위해 사용되는 JTree를 생성하는 데 사용되어진다.

XSLT 처리기 : XML 파서를 통해 로딩된 XML 문서와 XSL 문서는 XSLT 처리기를 통해 사용자 입력 폼에 해당되는 HTML 문서로 변환된다. 사용자는 다양한 XSL 문서나 템플릿 XML 문서를 받아들여 XSLT 처리를 수행하며 그 결과로 다양한 형태의 입력 폼이나 표시 결과를 자체 브라우저를 통해 확인해볼 수 있다.

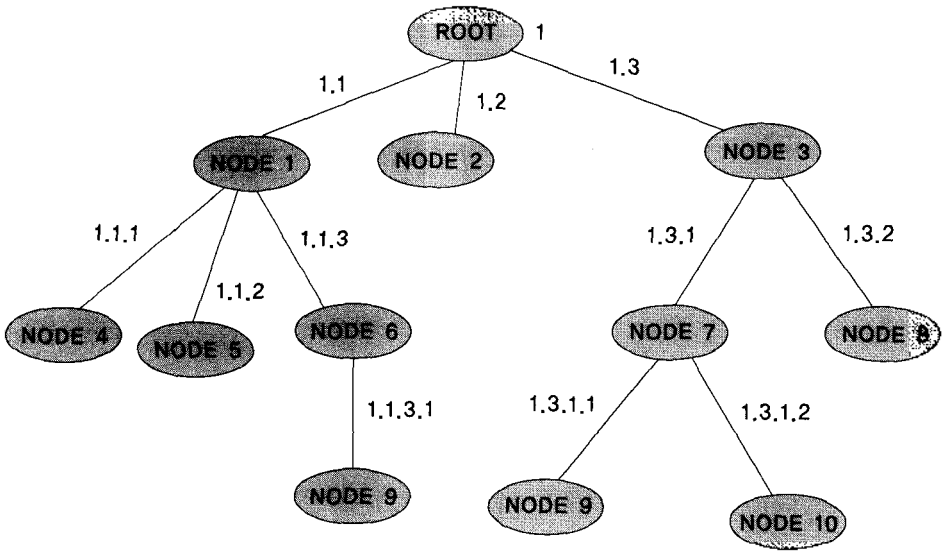
HTML 브라우저 : HTML 브라우저는 XSLT 처리기를 통해 생성된 HTML 문서를 보여주며 입력 폼에서 다양한 입력 양식의 정의와 폼 이벤트의 처리를 통하여 사용자 입력을 XML 생성기에 넘겨준다.

〈그림 2〉는 구현된 XML 문서 편집기의 화면 구성을 보여준다. 화면의 왼쪽 창은 DTD나 XML, XSL 문서를 파싱하여 그 결과를 JTree 형태로 보여주며, 오른쪽 창은 HTML 브라우저를 통하여 HTML 문서를 폼 형태로 나타내주고 있다.

폼을 기반으로 하는 XML 문서의 작성을 위해 본 논문에서는 템플릿 XML 문서의 각 엘리먼트에 이름 대신 고유 식별자를 부여하고 그것에 기초한 매핑 방식을 이용하였으며, 발생되는 폼 이벤트는 (*element_id = value*)



〈그림 2〉 XML 문서 편집기 화면 구성



〈그림 3〉 고유 경로에 따른 XML 엘리먼트의 키 부여

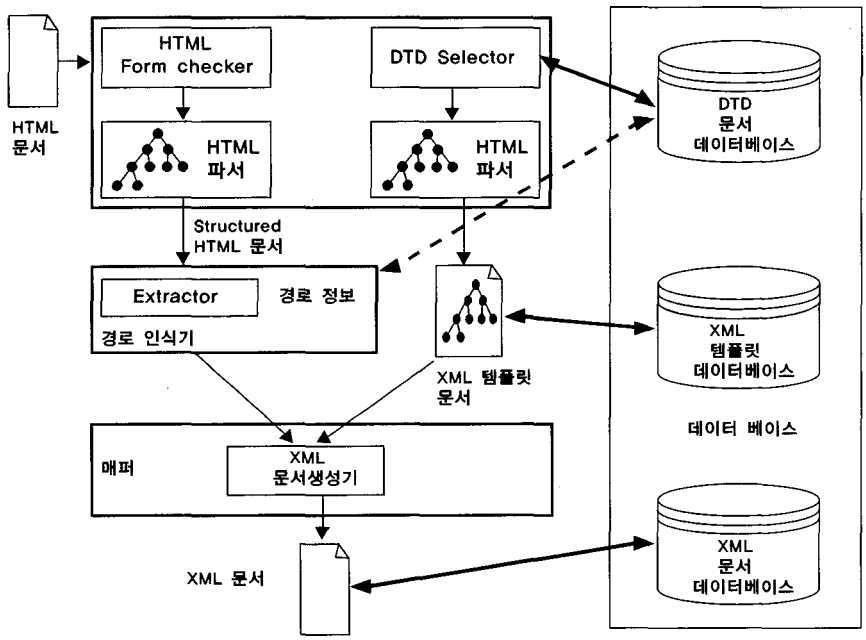
의 형태를 가진다. 〈그림 3〉은 XML 엘리먼트에 고유 식별자를 부여한 예를 보여주는데, 식별자는 각 엘리먼트의 고유 경로를 근거로 하여 부여된다. 그림에서 루트 노드에 대한 키 값을 1로 하고, 자식 노드들의 키 값을 부여하기 위해 깊이 우선 탐색(depth-first search) 방식으로 모든 노드들을 순회한다.

이와 같이 고유한 경로를 기초로 하여 템플릿 XML 문서의 각 엘리먼트에 대한 고유 식별자를 부여함으로써, 폼 양식을 통해 얻어진 엘리먼트와 그 데이터를 매핑시킬 때 엘리먼트 탐색 비용을 최소화할 수 있다. 즉, 엘리먼트의 키 값 속에는 루트 노드에서 해당 엘리먼트로의 경로 정보가 포함되어 있고, 그런 키 값을 통한 엘리먼트의 탐색은 루트 노드에서부터의 최소 경로가 된다.

3.2 원본 HTML 문서의 변환

XML 기반의 문서 관리를 위해서 기존의 HTML 문서에 대한 XML 변환이 필요하다. 특히 관공서나 기업의 경우 같거나 유사한 양식을 가지는 방대한 분량의 HTML 문서를 보관하고 있다. 여기서는 문서의 유사성을 기반으로 첫 변환을 통해 얻어진 문서의 경로 정보를 이용하여 유사하거나 같은 구조를 가지는 문서에 대해서는 사용자의 개입 없이 자동적인 변환을 수행할 수 있는 자동 변환 기법을 제안한다.

일반적으로 비구조적 형상에서 구조적 형상으로 변환하는 데 있어 필수적인 과정은 분석과 추출이다. 그림 4에 나타난 문서 변환 모듈의 구조에서 구조 분석기(Structure



〈그림 4〉 문서 변환 모듈의 전체 구조

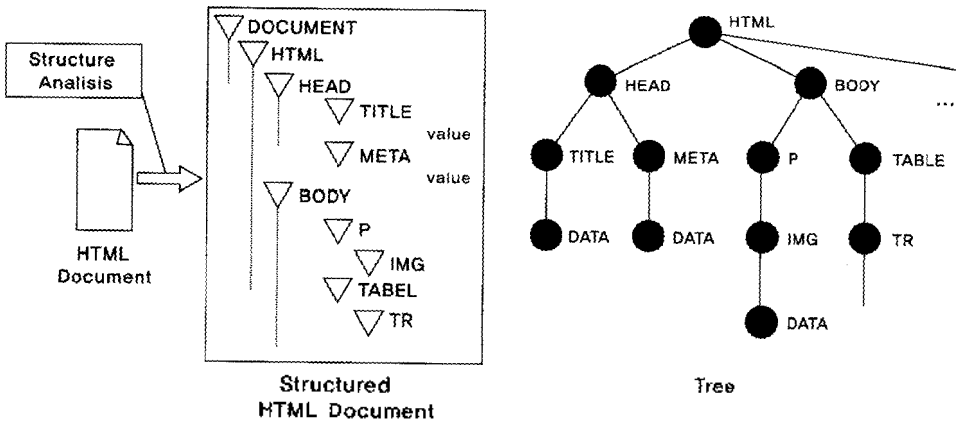
Analyzer)는 HTML 문서의 구조를 분석하며 입력되는 HTML 문서에 대한 구조 분석 모듈과 문서가 사용할 DTD의 엘리먼트들의 구조 관계 분석 모듈로 이루어져 있다. HTML 문서에 대한 구조 분석은 문서 내의 데이터에 대한 위치 파악을 수월하게 하며 데이터 경로를 중심으로 데이터에 의미를 부여한다. 또한 DTD 분석을 통해서 데이터가 매핑되어 들어갈 실제 영역을 지정할 수 있다. 여기서 실제 영역이란 XML 문서를 이루는 기본 골격을 의미하며 DTD에서 정의한 엘리먼트들의 조합을 통해 구체화된다.

유사 패턴을 가진 문서들의 변환 과정은 크게 사용자 개입에 의한 초기 문서 변환과 초기 문서 변환에서 나온 정보를 이용하여 유사한 다른 문서들을 자동으로 변환하는 반복 변환으로 나뉜다

유사한 HTML 문서들의 집합 D가 있다고 가정하자. 집합 내 문서의 자동 변환을 위해서는 최초 문서에 대해 (i) 문서의 구조 분석, (ii) 문서와 연관된 DTD 문서의 분석, 그리고 (iii) 데이터 및 경로 정보의 추출 등을 수행하여 해당 XML 문서를 생성한다. 여기서 경로 정보의 추출은 사용자의 개입을 통해서 이루어지며 유사한 다른 문서들에 대한 자동 변환에 사용된다. 다음은 집합내의 임의의 문서 d_i에 대한 변환 과정을 설명한다.

HTML 문서의 분석 : 우선 집합 내의 문서 d를 파싱하여 트리를 구성한다. 트리에서 각각의 태그는 계층적으로 표현되며 트리의 말단에는 추출될 실제적 데이터들이 위치하고 있다. 이때 데이터들과 그들의 경로 정보는 분석 과정을 마친 후 경로 인식기로 전달된다.

〈그림 5〉는 문서의 분석을 통해 생성되는 트리의 예를 보여준다. 트리의 각 노드는 해석된 HTML 태그로 표현되는 데 루트 노드는 HTML 문서를 대표하며, 내부 노드는



〈그림 5〉 구조화된 HTML 문서

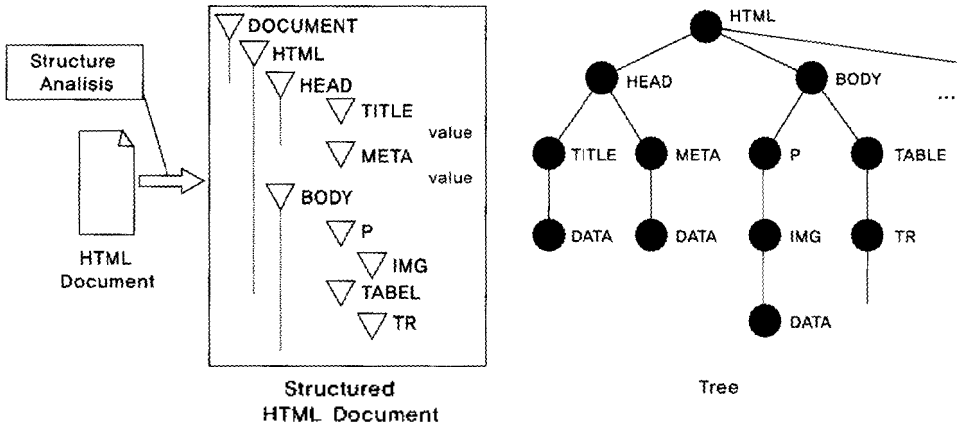
HEAD나 BODY를 나타내는 노드에서부터 HTML의 가장 작은 단위 태그를 나타내는 노드들로 구성되며 태그가 가지는 이름, 속성, 값의 정보를 포함하게 된다. 말단 노드에는 추출될 데이터가 저장된다. 태그에서 사용되는 URI나 URL은 태그를 나타내는 노드의 자식 노드로 표현된다.

DTD 문서의 분석 : 사용자가 해당 문서에 연관된 DTD를 선택하면 구조 분석기는 저장소에서 해당 DTD 문서를 검색하여 파싱하고, DTD에 정의되어 있는 엘리먼트들을 추출한다. 추출된 엘리먼트들은 DTD에 명시된 그들 사이의 관계에 따라 다시 조립된다. 이 과정에서는 DTD에 정의된 엘리먼트만 고려하며 결과적으로 해당 XML 문서의 골격에 해당되는 템플릿 XML 문서를 만들게 된다. 템플릿 XML 문서는 별도로 저장되어 유사한 다른 문서를 변환할 때 재사용된다.

템플릿 XML 문서를 만드는 과정에서 엘리먼트들은 정확히 한번씩 사용된다. 그러나 많은 XML 문서에서 동일한 엘리먼트가 여러 번 나타날 수 있다. 예를 들어, 엘리먼트를 정의할 때 출현 빈도를 나타내기 위해 "*"나 "+", "?" 등의 기호가 사용되는 데, "*"의 경우 엘리먼트가 문서 내에 없어도 되거나 여러 번 출현함을 의미하며, "+"의 경우는 한번 혹은 여러 번의 출현을 의미한다. "?"의 경우 한 번 혹은 나타나지 않음을 의미한다. 만약 동일한 엘리먼트가 문서 내에 여러 번 나타나야 할 경우, 템플릿 XML 문서에 반복되는 엘리먼트를 삽입하는 추가적인 과정이 필요하게 된다. 이 과정은 데이터 삽입 과정 전에 이미 사용된 동일 엘리먼트가 데이터를 가지고 있는 경우에만 엘리먼트를 생성하고 데이터를 삽입함으로써 해결할 수 있다.

경로 정보 저장과 데이터 추출 : 이 단계에서는 데이터를 추출하고 그에 관한 경로 정보를 저장하며, 데이터가 위치한 경로와 템플릿 XML 문서 안의 해당 엘리먼트와의 관계를 결정한다. 이때 사용자는 정확한 관계 설정을 위해 제공되는 인터페이스 상에서 데이터의 내용과 해당 엘리먼트의 연결에 관여할 수 있다. 이러한 과정은 각각의 데이터에 대해 반복적으로 수행된다. 모든 데이터에 대한 경로 정보의 저장이 끝나면, 다음 문서의 변환에 재사용되어 나타나는 경로는 저장된 경로 정보와의 비교를 통해 해당 엘리먼트로 치환된다. 이때 더 이상의 사용자 개입이 없이 자동적인 변환을 시도할 수 있다. 그림 6은 데이터의 경로와 템플릿 XML 문서에 있는 엘리먼트와의 관계를 생성하는 과정을 보여주고 있다.

예를 들어, 공공기관에서 사용하는 기안문에 해당하는 HTML 문서를 트리로 구조화할 경우 발신자의 데이터가 위치하는 경로가 *html→body→table[5]→tr[1]→td[1]→p[1]* 이라고 가정하자. 변환기는 사용자에게 데이터가 위치한 경로와 그 내용을 알려주고, 사용자는 내용을 고려하여 인터페이스에서 〈발신인〉 엘리먼트를 선택하면 템플릿 XML 문서에서 〈발신인〉의 실질적 노드를 찾아 데이터를 삽입하게 된다. 이때 경로 인식기는 위의 경로를 대신하여 엘리먼트 〈발신인〉을 기억하게 되며 이러한 관계는 앞으로 입력될 HTML 문서의 변환에 반복적으로 사용된다.



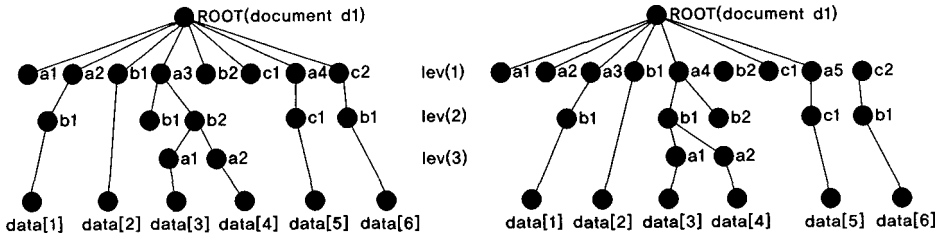
〈그림 6〉 데이터 경로와 템플릿 XML 문서의 관계

XML 문서의 생성 : 추출기는 트리의 말단에서 추출한 데이터를 매퍼에게 전달하며 매퍼는 템플릿 XML 문서에 데이터를 삽입하는 과정을 수행하게 된다. 모든 데이터의 삽입이 끝나면서 해당 XML 문서가 완성되게 된다.

3.3 유사 문서의 변환

문서의 최초 변환 과정에서 XML 문서와 더불어 경로 정보가 생성되었다. 생성된 경로 정보를 이용하면 유사한 문서의 변환에 있어 사용자 개입을 최소화하거나 배제할 수 있고 DTD 분석 과정도 생략할 수 있다. 변환에 있어 집합내의 문서가 서로 동일한 구조를 가졌을 경우 변환에 아무런 문제없이 진행되지만, 동일하지 않은 유사한 구조를 지닌 문서라면 약간의 수정 절차가 필요하게 된다. 여기서 변환할 문서는 공통 패턴을 지니고 있는 문서들 중에서도 문서의 데이터 유실이 없는 경우를 전제로 설명하고자 한다. 전체적으로 유사 문서의 반복 변환은 HTML 문서의 분석, 데이터 추출 및 XML 문서의 생성 등으로 이루어진다. HTML 문서의 분석 과정에서는 입력되는 문서를 트리 형태로 변환하여 경로 정보를 기반으로 데이터를 추출한다. 변환기는 최초 변환 후 저장되어 있는 경로 정보를 불러와 경로 정보에 저장된 규칙에 따라 데이터를 추출하게 된다. 모든 데이터의 삽입이 완료될 때까지 추출된 데이터가 삽입될 해당 템플릿 XML 문서의 엘리먼트를 찾는 과정을 반복하여 새로운 XML 문서를 생성한다.

HTML 문서의 분석 : 유사한 HTML 문서의 분석 과정에서는 최초 문서의 분석과 동일한 과정을 거쳐 트리 형태로 표현한다. 〈그림 7〉은 유사한 HTML 문서 d_1과 d_2를 트리의 형태로 보여주고 있다.



〈그림 7〉 유사한 문서의 트리 구조

〈그림 7〉에서 두 트리는 유사한 구조를 보이지만 세부적으로 레벨 1에서 노드 〈a2〉가 추가되었고, 레벨 3에서 〈a1〉과 〈a2〉의 부모 노드가 레벨 2의 〈b1〉으로 바뀌어 있다.

데이터 추출과 경로 정보 수정 : 유사 문서의 트리 표현에서 레벨 i에 위치한 임의의 HTML 태그 〈a〉의 *k*번째 태그를 *N[ak]*라고 하면 최초 문서의 변환에서 얻어진 경로 정보를 바탕으로 데이터를 추출하는 데 그 경로 정보는 〈표 1〉과 같다.

유사한 문서 d_2에 대해서 첫 데이터가 위치한 곳의 경로를 저장된 경로 정보 파일로부터 읽어오게 된다. 경로 인식기는 data[1]의 실제 위치가 N[a3] N[b1] data[1]임을 발견하고 기억된 경로 정보와 상이하므로 data[1]에 관계되는 경로를 수정하게 된다. d_1을 변환할 때 기록되었던 경로 정보에서 레벨 1에 있는 〈ak〉 태그에 대해서 일률적으로 k에서 k+1만큼의 수정이 일어나게 된다. 수정된 경로 정보는 〈표 2〉와 같으며 〈표 1〉의 $\overline{P_1}$, $\overline{P_3}$, $\overline{P_4}$, 그리고 $\overline{P_5}$가 수정되었다.

다음 데이터인 data[2]에 대해서는 특별한 경로변환 없이 데이터 추출이 가능하다. 다음은 d_2문서변환에서 data[3]에 대한 확인 과정이다. d_1문서의 경로 정보에는 data[3]의 위치가 N[a3]→N[b2]→N[a1]→data[3]으로 기록되어 있지만 이 경로 정보로는 실제 데이터를 찾을 수 없다. 왜냐하면, 실제 데이터는 N[a4] → N[b1]→N[a1]→data[3]에 위치하기 때문이다. 경로 인식기는 위의 정보를 통해 다른 경로 정보를 수정하게 된다. 즉 〈표 2〉에서 $\overline{P_3'}$의 정보와 레벨 2에서 〈bk〉를 갖는 정보 $\overline{P_1'}$를 k에서 k+1만큼 수정하게 된다. 위의 예제에서는 $\overline{P_3'}$와 $\overline{P_4'}$의 정보가 〈표 3〉과 같이 수

〈표 1〉 d_1의 변환과정에서 생성된 경로 정보

$\overline{P_1} = N_{a2} \cdot N_{b1}$: Root→N[a2]→N[b1]→data[1]
$\overline{P_2} = N_{b1}$: Root→N[b1]→data[2]
$\overline{P_3} = N_{a3} \cdot N_{b2} \cdot N_{a1}$: Root→N[a3]→N[b2]→N[a1]→data[3]
$\overline{P_4} = N_{a3} \cdot N_{b2} \cdot N_{a2}$: Root→N[a3]→N[b2]→N[a2]→data[4]
$\overline{P_5} = N_{a4} \cdot N_{c1} \cdot N_{b1}$: Root→N[a4]→N[c1]→data[5]
$\overline{P_6} = N_{c2} \cdot N_{b1}$: Root→N[c2]→N[b1]→data[6]

〈표 2〉 d_2의 변환과정 중에 수정된 경로 정보

$\overline{P_1'} = N_{a3} \cdot N_{b1}$: Root→N[a3]→N[b1]→data[1]
$\overline{P_2'} = N_{b1}$: Root→N[b1]→data[2]
$\overline{P_3'} = N_{a4} \cdot N_{b2} \cdot N_{a1}$: Root→N[a4]→N[b2]→N[a1]→data[3]
$\overline{P_4'} = N_{a4} \cdot N_{b2} \cdot N_{a2}$: Root→N[a4]→N[b2]→N[a2]→data[4]
$\overline{P_5'} = N_{a5} \cdot N_{c1} \cdot N_{b1}$: Root→N[a5]→N[c1]→data[5]
$\overline{P_6'} = N_{c2} \cdot N_{b1}$: Root→N[c2]→N[b1]→data[6]

〈표 3〉 d_2의 변환 후 갱신되어 저장된 경로 정보

$\overline{P_1''} = N_{a3} \cdot N_{b1}$: Root→N[a3]→N[b1]→data[1]
$\overline{P_2''} = N_{b1}$: Root→N[b1]→data[2]
$\overline{P_3''} = N_{a4} \cdot N_{b1} \cdot N_{a1}$: Root→N[a4]→N[b1]→N[a1]→data[3]
$\overline{P_4''} = N_{a4} \cdot N_{b1} \cdot N_{a2}$: Root→N[a4]→N[b1]→N[a2]→data[4]
$\overline{P_5''} = N_{a5} \cdot N_{c1} \cdot N_{b1}$: Root→N[a5]→N[c1]→data[5]
$\overline{P_6''} = N_{c2} \cdot N_{b1}$: Root→N[c2]→N[b1]→data[6]

정되었다. 이후 나머지 데이터에 대해서는 자동적인 추출이 가능해 지고 $\overline{P_1''}$를 대신해서 템플릿 XML 문서에 있는 알맞은 엘리먼트로 치환되어 진다. 〈표 3〉의 정보는 다른 유사한 문서의 변환에서도 비슷한 과정을 거치며 사용되어 진다.

XML 문서의 생성 : 최초 문서의 생성 과정과 동일하게 XML 문서를 생성한다. 즉, 추출된 데이터가 삽입될 해당 템플릿 XML 문서의 엘리먼트를 찾아 모든 데이터의 삽입이 완료될 때까지 삽입 과정을 반복하게 된다.

3.4 DTD 편집기

DTD는 유효한 XML 문서를 저작하기 위한 규칙들의 집합이며, 문서의 클래스에 문법을 제공하는 마크업 선언을 포함하거나 지정한다. DTD는 문서 구조를 정의하고 엘리먼트 타입과 속성, 엔터티를 선언한다. DTD를 갖는 문서는 DTD에 기술된 정의에 따라 문서의 유효성이 결정된다. 문서내의 모든 엘리먼트나 엔터티, 속성들은 DTD에 반드시 선언되어야 하며 모든 데이터 타입들은 그것에 대한 정의가 요구하는 규약들을 반드시 충족해야 한다.

XML 문서를 위한 DTD의 설계는 복잡하고 시간이 많이 소모되는 작업이다. DTD 편집기는 DTD를 개발하는 데 따르는 단조로운 작업의 반복을 피하면서 XML의 모든 규정에 따르는 적합한 DTD를 쉽게 개발할 수 있는 환경을 제공한다. 특히 본 논문에서는 WYSIWYG 기능에 중점을 두어 DTD 전문가에서 초보자까지 모든 계층의 이용자들이 손쉽게 DTD를 개발할 수 있게 하였다.

본 논문에서 구현하는 DTD 편집기는 일반적인 텍스트 기반의 편집 환경 외에도

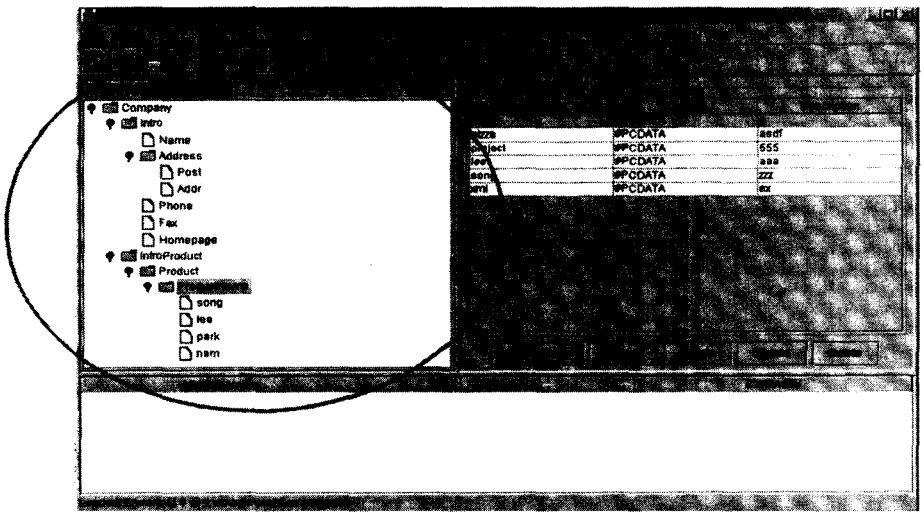
DTD 트리 편집 창, 텍스트 문서 편집 창, 엘리먼트 리스트, 속성 편집 창으로 구성되는 GUI 기반의 편집 환경을 제공하며 상호 전환이 가능하다. DTD 문서 작성자는 텍스트 기반 편집 환경 하에서 GUI 기반 환경이 미처 지원하지 못하는 상세한 편집을 수행할 수 있다. DTD 편집기를 구성하고 있는 모듈과 그 기능은 다음과 같다.

DTD 파서 : *com.wutka.dtd* 라는 패키지에서 제공되는 DTD 파서를 사용하였으며 http://www.wutka.com/dtdparser.html에서 상세 정보 및 파서를 내려 받을 수 있다. Java로 구현된 인터페이스를 제공하며 DTD를 파싱한 후 DOM 트리 형태로 변환하는 기능을 제공한다. 또한 트리 내부의 노드들에 접근해서 엘리먼트 및 애트리뷰트의 내용을 가져올 수 있는 함수를 지원한다.

DTD tree maker : 기존 DTD를 읽었을 때 그 문서를 트리 형태로 표현해 주는 모듈이다. 내부적으로는 DOM 트리 형태를 하며 읽으면서 메모리에 올라오기 때문에 빠른 속도로 Java Swing 패키지를 통해서 트리 GUI를 제공한다.

DTD manipulation : DTD 저작에 관련된 모듈로써 DTD를 구성하는 엘리먼트나 애트리뷰트를 생성하고 삭제할 수 있게 해준다. 엘리먼트, 애트리뷰트의 이름, 종류, 속성 등에 대한 설정을 바꿔줌으로써 내부적인 속성을 수정하는 기능을 제공한다. 이 모듈은 DTD tree maker 모듈과 연계되어서 DTD 엘리먼트가 생성되고 삭제될 때 GUI 상에 보여지는 트리를 갱신하게 된다.

Element List : 분야에 따라 필요한 DTD를 미리 데이터베이스에 저장하여 사용할 수 있다. Element List는 이 데이터베이스에 접속하여 DTD 엘리먼트 테이블에 있는 정보를 가져온다. 현재 Mysql을 사용하고 JDBC를 이용해서 정보를 가져온다. JDBC 드라이버만 바꿔주면 어떤 데이터베이스와도 연동이 가능하기 때문에 이질 데이터베이스와 쉽



〈그림 8〉 DTD 편집기 실행 모습

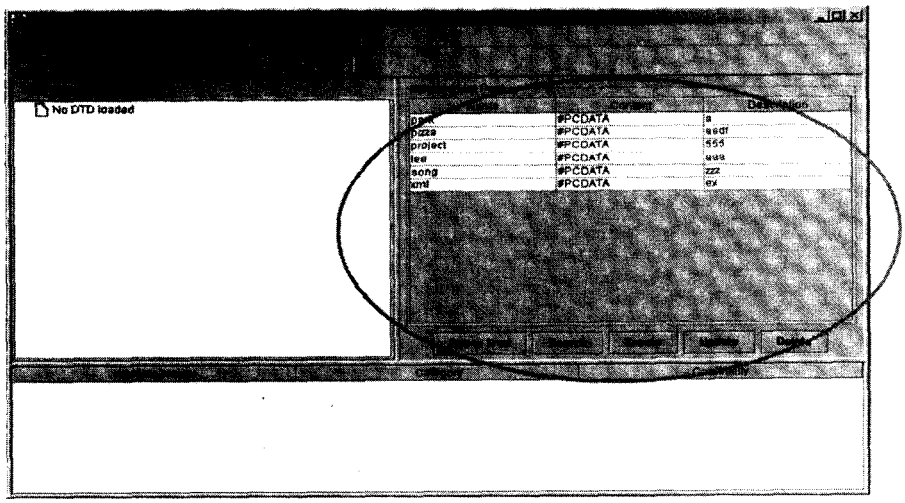
게 연동될 수 있다. 또한 권한에 따라 List에 있는 DTD 엘리먼트의 검색, 생성, 수정, 삭제가 가능하며 일반 사용자들은 필요한 DTD 엘리먼트를 List에서 검색하고 그 엘리먼트를 문서에 삽입하여 DTD를 작성하게 된다.

〈그림 8〉은 DTD 편집기의 초기 화면을 보여준다. 왼쪽 창이 DTD가 보여지는 부분이고 오른쪽 창은 엘리먼트 리스트 테이블이다. 새로운 DTD를 만들기 위해서는 먼저 DTD 문서의 루트 이름을 지정한다. 루트를 생성한 다음 자식 엘리먼트를 생성하면서 트리를 구성하게 되는 데 GUI 상에 보여지는 트리 구조는 곧 DTD 문서의 구조를 나타낸다. DTD 엘리먼트를 추가하는 위해서는 트리 상에서 자식의 이름을 삽입하거나 오른쪽 창의 엘리먼트 리스트에서 필요한 DTD 엘리먼트를 찾아 트리에 추가하면 된다.

〈그림 9〉는 예제 DTD 파일을 읽었을 경우 화면이다. 왼쪽 화면에 DTD 엘리먼트가 트리 형태로 표현된 것을 볼 수 있다. 문서의 내용은 소스 형태로도 볼 수 있는 있으며 왼쪽 화면 상단의 메뉴를 통해 선택한다. 물론 소스 보기 모드에서도 소스 코드를 수정하는 작업이 가능하다

4. 결론 및 향후 연구 과제

본 논문에서는 XML 기반의 통합 문서 관리 시스템을 구현하였다. 시스템은 크게 HTML로 작성되어 있는 유사한 문서에 대해서 공통 패턴을 기반으로 XML 문서로의 자동 변환이 가능한 문서 변환기와 새로운 XML 문서의 편집을 위해서 트리를 기반으로 하는 기존의 XML 문서 작성 외에도 문서의 양식을 반영하는 폼을 기반으로 하여 XML 문서 의 작성이 가능한 문서 편집기, 그리고 DTD 작성을 쉽게 하기 위한 DTD 편집기로 구성되어 있다. 결과적으로 XML에 익숙하지 않은 사용자의 경우에도 원하는



〈그림 9〉 트리 형태의 DTD 문서 내용

XML 문서를 쉽게 작성할 수 있으며 기존의 HTML 문서를 XML로 자동 변환하여 유사한 대용량 문서의 효과적인 문서의 관리가 가능하게 하였다.

입력 폼의 생성을 위해서는 템플릿 XML 문서와 XSL 문서를 필요로 하기 때문에, 초기 템플릿 XML과 XSL 문서의 제작이 부담으로 작용할 수 있다. 하지만 기업이나 관공서 등과 같이 이미 정형화된 양식의 문서를 빈번하게 사용하는 경우 폼을 이용한 대용량의 문서 작성과 유사한 폼을 가지는 HTML 문서의 자동 변환 등이 업무의 효율을 높이는데 크게 기여할 것이다. 또한 고정된 폼 생성의 관점에서 접근하기보다는 다양한 양식의 폼을 지원할 수 있도록 XSLT 기술을 가지고 설계되었으므로 기업체의 문서 양식 변화가 응용에 영향을 주지 않으며, XSL 문서의 수정을 통해 다양한 입력 폼 생성이 가능하다는 이점이 있다.

참 고 문 헌

[1] David Hunter, Beginning XML, 2000

[2] Frank Boumphrey, Propessional XML Application, 1999

[3] Alexander, Tom, Professional Java XML Programming, 2000

[4] James Clark, XSLTransformations (XSLT) Version 1.0, W3C Recommendation, 16 November 1999. http://www.w3.org/TR/xslt

[5] Kevin Williams Professional XML Databasees, 2000

[6] Arnaud Sahuget, Fabien Azavant, "WysiWyg Web Wrapper Factory (W4F)", Available athttp://db.cis.upenn.edu/~sahuguet/WAPI/wapi/

[7] Ling Liu, Calton Pu, Wei Han, "XWRAP: An XML-enabled Wrapper Construction System for Web Information Sources", in Proceedings of the 16th International Conference on Data Engineering, pp.611-621, 1998

[8] Ling Liu, Wei Han, David Buttler, Calton Pu, "An XML-Enabled Wrapper Construction System for Web Information Sources", SIGMOD Conference, pp.540-543, 1999

[9] Taniar, Y. Jiang, J.W. Rahayu, L. Bishay, "Structured Web Pages Management for Efficient Data Retrieval," Proceedings of the First International Conference on Web Information Systems Engineering

(WISE'00), 2000

[10] Gerald Huck, Peter Fankhauser, Karl Aberer, Erich Neuhold, "Jedi: Extracting and Synthesizing Information from the Web", in Proceeding of CoopIS, pp.32-43, 1998

[11] Anders Kristensen, "Formsheets and the XML Forms Language," in Proceedings of WWW9, Toronto, Canada, May 1999.

[12] Vidur Apparaoet al., Document Object Model(DOM) Level 1 Specification, W3C Recommendation, 1 October, 1998. http://www.w3.org/TR/REC-DOM-Level-1/

[13] Eric Armstrong, The Java API for Xml Parsing (JAXP) Tutorial, Version 1.1, 16 May 2001. http://java.sun.com/xml/jaxp-1.1/docs/tutorial/

[14] Java API for XML Processing 1.1 Specification, May 2001. http://java.sun.com/ xml/ xml_jaxp.html

[15] N. Ashish, C.A. Knoblock, "Wrapper Generation for Semi-structured Internet Sources", in Proceeding of the Workshop on Management of Semi-structured Data, pp.8-15, Tucson, Arizona, 1997

[16] Dan Connolly, Rohit Khare, Adam Rifkin, "The Evolution of Web Documents: The Ascent of XML", World Wide Web Journal, Vol. 2, No. 4, pp.119-128, 1997

[17] Jean Robert Gruser, Louiqa Raschid, Maria Esther Vidal, Laura Bright, "Wrapper Generation for Web Accessible Data Sources", Proceedings of the Third International Conference of Cooperative Information Systems, pp.14-23, 1998

[18] N. Kushmerick, D. Weil, R. Doorenbos, "Wrapper induction for information extraction", Proceeding of Int. Joint Conference on Artifical Intelligence (IJCAI), pp.729-737, 1997

[19] S. Lee, E. Hwang, K. Byeon, "Template-based XML Data Integration System", International Conference on Electronic Commerce, Seoul, Korea, pp.17-23, 2000

[20] K. Oh and E. Hwang, "Automatically Generating XML Documents from Web Data with Similar Pattern", Journal of Computer and Information Science, Sep. 2002.

저 자 소 개

황인준	(E-mail : ehwang@ajou.ac.kr)
1988.	서울대학교 컴퓨터공학과(학사)
1990.	서울대학교 컴퓨터공학과(석사)
1998.	Univ. of Maryland at College Park 전산학과(박사)
1998. 8 ~ 1999. 8	Bowie State Univ., 조교수
1999. 6 ~ 1999. 8	Hughes Research Lab. 연구교수
1999. 9 ~ 2003. 2	아주대학교 정보통신전문대학원 조교수
2003. 3 ~ 현재	아주대학교 정보통신전문대학원 부교수
관심 분야	데이터베이스, 멀티미디어 검색, 웹 응용, XML 응용 등