Form 기반의 XML 문서 편집기 구현

고 탁 현[†] · 황 인 준[††]

요 약

트리 구조를 기반으로 하는 기존의 XML 문서 편집은 사용자로 하여금 XML에 대한 사전 지식을 필요로 한다. 그러나 XML 문서의 작성과 활용이 보편화되기 위해서는 이러한 문서 편집 방식에서 벗어나 누구나 쉽게 작성할 수 있는 환경이 제공되어야 한다. 본 논문에서는 트리 방식의 문서 편집뿐만 아니라 문서의 양식에 근거한 form을 기반으로 하는 편집 환경도 지원하는 새로운 XML 문서 편집기를 제안한다. 특히, form을 이용한 문서 편집은 기업이나 관공서 등과 같이 정형화된 양식을 이용하여 다량의 XML 문서를 작성하는 경우에 아주 효과적이다. Form 자체는 HTML 문서로 표현이 되고, 이러한 HTML form은 template XML 문서와 XSL 문서를 이용한 XSLT 적용을 통해 자동적으로 생성된다. 생성된 HTML form은 내장된 브라우저를 통해 사용자에게 보여지며, form에 입력된 사용자 데이터는 XML 문서로 변환되어 XML 저장소에 저장된다.

Implementation of Form-based XML Document Editor

Tak Hyun Ko[†] · Eenjun Hwang[††]

ABSTRACT

Existing XML editors, which are usually tree-based, require knowledge on the XML from users. But this requirement should be removed in order for any user to create XML documents easily. In this paper, we developed a new XML editor which provides both the usual tree-based interface and the form-based interface derived from the original document. Editing XML documents through forms will be especially effective in the places such as enterprise or municipal office where a large amount of documents of same format need to be generated. Forms, which are HTML documents, are generated automatically through the XSLT using both template XML document and XSL document, and displayed on the built-in HTML browser. When a form is filled out by user, it will be transformed into its corresponding XML document and stored into the XML repository.

키워드 : XML 에디터(XML Editor), 폼(Form-based), 템플릿 XML(template XML), XSLT

1. 서 론

최근 인터넷의 발전이 가속화되고 사용이 보편화됨에 따라, 지금까지 웹 상에서 문서를 공유하고 전송하기 위한 대중적인 형식이었던 HTML은 많은 한계점을 드러내게 되었다. 이에 따라 XML(eXtensible Markup Language)이 인터넷상의 새로운 정보 교환의 표준으로 채택되었다. 내용의 효과적인 표시(display)를 위한 제한적인 용용 분야라고 할 수 있는 HTML은 특정한 애플리케이션(웹 브라우저)을 위해 설계되어진 언어이다. 즉, 웹 브라우저를 통해 문서 내용을 표현하는 데 있어서는 많은 장점을 가지고 있는 반면 브라우저에 종속적인 태그만을 사용해야 되고, 그로 인해 다양한 문서의 내용(contents)을 기술하는 데 있어서는 많은 제약을 가지게 되었다.

이와는 달리 XML은 데이터의 교환을 고려해서 설계되어진 언어이다. 즉, 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 문서들의 상호교환 뿐만 아니라 데이터 통합과 데이터 처리의 자동화 등에 많은 장점을 가질 수 있다. 또한 XML은 문서의 구조와 표현을 분리시켰다. XML 문서는 문서의 구조와 의미에 관한 정보만을 기술하며, 그 요소들을 표현하는 부분은 스타일시트를 사용한다. XSL(eXtensible Stylesheet Language)은 바로 이런 XML 데이터와 문서에 사용될 목적으로 설계된 스타일시트 언어이다. 이렇듯 XML은 그의 스타일시트 언어인 XSL과 함께 작동하여 XML 문서의 내용을 다양하고 세련된 형태로 표현할 수 있다. 또한 XSLT(eXtensible Stylesheet Language for Transformations)

† 준 회 원 : 아주대학교 대학원 정보통신 전문대학원
†† 종신회원 : 아주대학교 정보통신 전문대학원 교수
논문접수 : 2001년 6월 18일, 심사완료 : 2002년 2월 5일

[3]를 사용하여 HTML로 변환하면 일반 웹 상에 XML 문서를 표현할 수도 있다. XSLT는 좀 더 큰 범위의 언어인 XSL의 하위 구성 요소이다. XSLT를 사용하면 XML 문서를 다른 형태의 XML 문서뿐만 아니라 HTML과 같은 다른 형식의 문서로도 변환이 가능하다. 이때 변환된 HTML 문서의 내용에 해당하는 것은 XML 문서로부터 생성되고, 화면 표시에 관계된 것들은 XSL 스타일시트로부터 생성된다. 현재 이러한 XML의 목표에 부합되도록 많은 XML 애플리케이션이 개발되고 있으며 그와 연동하여 효율적인 문서 작성을 위한 다양한 형태의 XML 편집기가 연구되고 있다.

본 논문에서는 기존의 XML 문서 편집기가 가진 문제점들을 살펴보고, XML 패밀리(family)중의 하나인 XSLT를 이용하여 새로운 개념의 XML 문서 편집기를 개발한다. 본 논문에서 제안하는 XML 문서 편집기는 기존의 트리 구조를 기반으로 하는 문서 제작 방식에서 탈피하여, 원래 문서의 양식을 반영하는 form을 기반으로 사용자가 쉽게 XML 문서를 제작할 수 있는 환경을 제공한다. 본 XML 문서 편집기는 Java 1.3을 기반으로 개발되었고, 자체 HTML 브라우저를 내장하고 있어서 실시간에 XML 문서를 HTML문서로 변환 브라우징 할 수 있도록 제작되었다. 아울러 브라우저에 표현된 form으로부터 사용자 입력을 받아 해당 XML 문서로 변환하여 데이터베이스에 저장하는 기능을 수행한다.

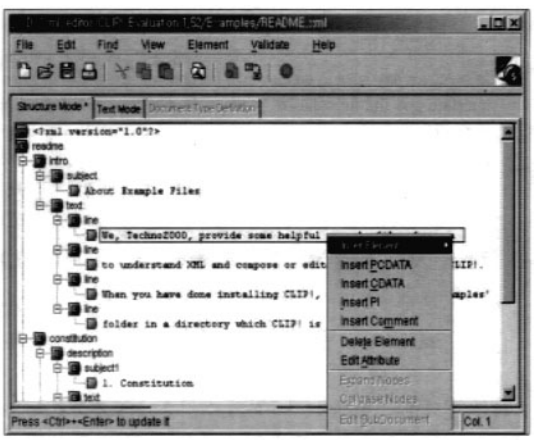
2. 관련 연구

2.1 XML 관련 저작도구 분석

XML 기반의 저작 도구의 구성은 WWW을 위한 XML 편집기, XML 문서 분석과 검증을 위한 XML 파서, 작성된 XML 문서의 디스플레이를 위한 XML 브라우저, XML 문서의 출력 관리를 위한 시타일시트 관리기로 구성한다[13]. 편집기는 크게 XML 편집기, DTD 편집기, XSL 편집기로 나누어 볼 수 있다. 기존의 편집기는 기본적으로 문서의 수정, 삭제, 저장 기능과 추가적으로 문서에 대한 validity를 검증하는 기능을 제공한다. 또한 기존의 편집기는 위에서 말한 세 가지 형태의 문서 편집을 통합적으로 지원하는 환경이 아닌 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 확인하는 방법을 사용해야 한다.

2.2 기존의 XML 문서 편집기 분석

XML 문서 편집기의 주요한 기능은 크게 XML 문서의 작성과 검증(validation)에 있다고 할 수 있다. 즉, XML이 제공하는 syntax에 적합한 well-formed XML 문서나 DTD (Document Type Definition)에 부합되는 유효한(valid) XML 문서를 어떠한 오류 없이 사용자가 보다 쉽게 제작할 수 있도록 인터페이스를 제공해야 한다. 이를 실현하기 위해 현재까지 많은 XML 문서 편집기가 제안되고 개발되어 왔지만 대부분 아직까지 많은 문제점을 나타내고 있다. 구체적으로 기존의 XML 문서 편집기는 효율적인 브라우징 기능을 제공해 주지 못하고 있으며, XML 문서의 구조적 정보를 표현하기 위해 (그림 1)과 같이 단순한 트리 구조를 기반으로 하는 인터페이스 상에서 XML 문서를 제작할 수 있는 기능만을 제공한다. 이것은 XML 문서를 제작하기 위해서는 사용자가 어느 정도 XML에 대한 사전 지식을 가지고 있어야 한다는 것을 의미한다. 그러나, XML 문서의 사용이 보편화되고 일반화되는 현실에 비추어 보면, 사용자가 보다 쉽게 사용할 수 있고, XML에 대한 지식이 없이도 XML 문서를 쉽게 작성할 수 있는 환경을 제공하는 XML 문서 편집 도구의 개발이 절실하다.



(그림 1) 트리 기반 XML 문서 편집기

또한, XML 문서가 전자 상거래에서 문서 교환의 중요한 수단이 된다는 점과 XML 문서 자체의 내용이 데이터베이스와 밀접한 관계를 가진다는 것을 고려해 볼 때, XML 문서와 데이터베이스와의 상호 연계에 대한 부분도 고려되어야 한다. 단순한 XML 문서의 편집 기능뿐만 아니라 실제 XML 문서를 파싱하여 데이터베이스에 저장하고 사용자가 원하는 질의에 대해 그 결과를 XML 문서로 변환하여 보여주는 기능이 지원되어야 한다[11].

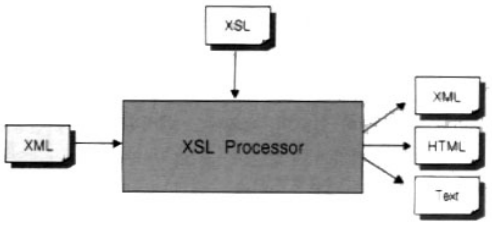
본 시스템에서는 문서의 양식을 반영하는 form을 기반으로 하는 XML 문서 제작 환경를 지원함으로써 XML에 대한 사전 지식이 없이도 일반 사용자가 쉽게 XML 문서를

작성할 수 있게 해주며 form을 통해 작성된 문서는 자동적으로 XML 문서로 변환시키는 새로운 XML 문서 편집기를 개발하고자 한다.

2.2 XSLT 특성 분석

XSL[3]은 스타일시트를 만들기 위해 사용되는 XML 기반의 언어이다. XML 엔진은 이러한 스타일시트 언어를 사용하여 XML 문서를 다른 문서 타입으로 변환할 수 있고, 또한 출력물의 형식을 지정할 수 있다. XSL은 XSLT와 XSL FO(Formatting Objects)[10]로 나누어진다. XSLT는 데이터를 다른 데이터로 변환시킬 때 사용되고, XSL FO는 마치 캐스케이딩(cascading) 스타일시트처럼 디스플레이를 위해 데이터를 좀 더 조작할 때 사용된다. 본 시스템에서는 이러한 XSL 중에서 이미 W3C의 권고안(Recommendation)으로 확정된 XSLT를 사용하였다.

XSLT는 (그림 2)와 같이 XML 문서를 입력 문서로 하여 well-formed XML 뿐만 아니라 HTML이나 일반 텍스트 형태의 다양한 출력 문서를 만들어 낼 수 있다. 이 중에서 브라우저 상의 디스플레이를 위해서는 XML 문서를 HTML 문서로 변환하는 작업이 필요하다. 본 시스템에서도 XML 문서의 브라우징을 위해 XSL 문서를 사용하였으며, XSLT를 적용한 후 생성된 HTML 파일을 내장된 HTML 브라우저를 통해 사용자에게 보여준다.



(그림 2) XSLT 문서 처리 흐름

〈표 1〉 XSLT Processor

Xalan	http : //www.apache.org
XT	http : //www.jclark.com
Saxon	http : //users.iclway.co.uk/mhkay/saxon

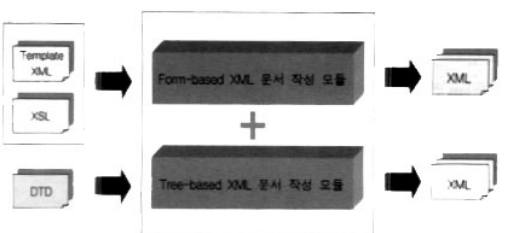
현재 개발된 XSLT 처리기에서 사용되고 있는 XSLT 엔진으로는 Xalan, XT, Saxon 등이 있으며 자세한 내용은 <표 1>에 나타난 사이트를 참고하기 바란다.

2.3 기타 관련 연구

최근에 출시된 다산의 XML Builder는 XML 데이터를 웹 브라우저 상에서 사용할 수 있도록 하기 위해 HTML Form을 제작해 주는 도구이다. DTD 정보를 이용한 Drag & Drop 방식을 통해 HTML Form을 생성해 주며 생성된 해당 Form에 대해서는 특정 XSLT tag를 매핑 시켜 놓아서 자동적으로 XSLT 문서를 생성해 주는 기능도 지원한다. 다시 말해서, XML Bulider는 XML 데이터를 웹 상에서 보여주기 위한 HTML Form과 XSLT 문서를 제작하는 도구이다. 본 논문에서 제안한 XML 문서 편집기는 XSLT 문서 제작을 위한 것이 아니라 Form 양식의 입력 인터페이스를 통해 XML 문서 자체를 제작할 수 있게 해주는 시스템이다. 또 다른 연구로서, W3C에서 제안된 XForms[14]는 2001년 8월 28일에 working draft로 제안되어 있으며 컴퓨터나 PDA, 핸드폰, 가전 제품과 같은 다양한 플랫폼을 지원하기 위한 확장된 Form을 제공한다[14].

3. 편집기의 전체적인 구성

본 편집기의 특징은 XML 문서를 작성하기 위해 기존의 트리 구조를 기반으로 하는 인터페이스 환경뿐만 아니라 문서의 양식에 따르는 form을 기반으로 하는 인터페이스를 함께 제공한다는 것이다. XML 문서의 구조적인 특성이 트리 구조와 잘 호환된다는 점은 대부분의 XML 문서 편집기가 트리 구조를 기반으로 하는 사용자 환경에서 문서를 제작하도록 설계된 이유를 잘 설명해준다. 하지만 그와 같이 설계된 XML 문서 편집기를 사용하여 문서를 작성하는 것은 XML에 능숙하지 못한 사용자에게는 적지 않은 부담을 주게 된다. 이러한 문제점을 완화하기 위하여 본 시스템은 form 양식을 지원하는 XML 문서 편집기를 설계하였으며 그 구성 요소는 (그림 3)에 나타나 있다.



(그림 3) XML 문서 편집기 구성요소

본 편집기는 크게 트리 구조를 기반으로 하는 XML 문서 작성 모듈과 form을 기반으로 하는 문서 작성 모듈로 나누어진다. 전자는 기존의 문서 편집기가 지원하는 방식을 취하여 구현되었고, 후자는 문서의 양식에 따르는 form 상에서 필요한 XML 문서를 작성하는 환경을 제공한다. 트리 기반의 XML 문서 작성 모듈을 통해서는 DTD가 존재하지 않는 well-formed XML 문서와 입력 DTD 구조에 유효한 XML 문서를 제작할 수 있으며, form 기반 XML 문서 작성 모듈에서는 template XML 문서와 그 XML 문서의 디

스플레이 정보를 가진 XSL 문서를 입력 문서로 하여 새로운 XML 문서를 제작할 수 있도록 설계되었다.

본 시스템은 Java1.3을 사용하여 구현되었고, 사용자 인터페이스는 Swing을 사용하였다. 또한 XML 문서의 유효성 검사를 위한 파서는 JAXP1.1[4]을 사용하였고, XML 문서의 브라우징을 위한 XSLT 엔진은 Xalan을 사용하였다.

3.1 트리 기반 인터페이스를 사용한 문서 생성

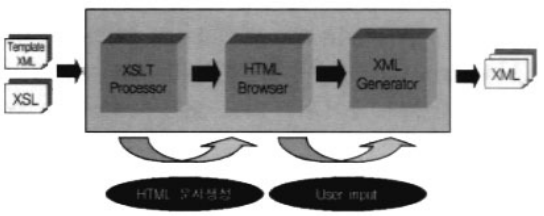
XML 문서의 작성을 위한 트리 기반 편집 모듈은 크게 트리 생성 및 저장 모듈, 트리 조작 모듈, 트리 편집 모듈, 트리 엘리먼트 filtering 모듈 등으로 구성되어 있다.

(그림 4) 트리 기반 XML 문서 작성을 위한 구성 요소

트리 생성 및 저장 모듈은 작성될 XML 문서의 DTD 사용 여부를 판단하여 well-formed XML 문서나 DTD에 유효한 XML 문서를 생성하고 저장하는 기능을 수행한다. 트리 조작 모듈은 트리의 확장(expansion)이나 축소(reduction), 트리 엘리먼트의 선택 여부 판별과 같은 기능을 수행한다. 마지막으로 트리 편집 모듈에서는 트리 조작 모듈에서 선택된 엘리먼트를 기준으로 새로운 엘리먼트의 추가나 기존의 엘리먼트 삭제가 이루어진다. 이 과정에서 트리 엘리먼트 filtering 모듈은 파싱된 DTD 문서의 구조 정보를 유지하면서 엘리먼트를 추가하고자 할 때 유효한 엘리먼트들의 리스트를 사용자에게 보여주어 원하는 것을 선택하게 하는 기능을 제공한다.

3.2 Form 기반 인터페이스를 이용한 문서 생성

Form을 기반으로 하는 인터페이스 상에서 XML 문서를 작성하기 위해서는 template XML 문서와 XSL 문서가 필요하다. 즉 사용자의 입력 form 양식을 생성하기 위해서 XML



(그림 5) Form-based를 지원하기 위한 시스템 주요 구성도

문서의 표현을 담당하는 XSL 문서를 중간 매개체로 사용하였으며, template XML 문서와 함께 입력 form 생성에 사용되어진다. (그림 5)는 사용자 입력 form 생성에서 만들어진 입력 form을 통한 새로운 XML 문서의 작성까지 전체적인 흐름을 보여준다.

그림에서 form을 기반으로 하는 XML 문서 작성을 위해서는 XSLT 처리기(Processor), HTML 브라우저(Broswer), XML 생성기(Generator)와 같은 시스템 구성 요소가 필요하다. XSLT 처리기는 template XML 문서와 XSL 문서를 입력으로 받아서 사용자의 입력 form에 해당하는 HTML 문서를 생성한다. 생성된 HTML 문서가 HTML 브라우저에게 전달되면, 브라우저는 이 HTML 문서를 브라우징하여 해당 입력 form을 사용자에게 보여주게 된다. 끝으로 XML 생성기는 브라우저 상에서 입력 form을 통해 입력받은 데이터를 바탕으로 새로운 XML 문서를 생성하게 된다. 여기서 XSLT 처리기는 해당 입력 HTML form을 생성할 때 XML 엘리먼트의 텍스트 값도 같이 읽어들여 브라우저 상에서 보여지게 하며, XML 생성기를 통해 만들어 진 XML 문서는 새로운 XML 문서를 생성하기 위한 template XML 문서로 재사용 가능하다.

앞에서 언급되었듯이 HTML 형태의 form을 생성하기 위해서는 먼저 template XML 문서와 XSL 문서를 제작하여야 한다. XSL 문서를 통해 생성된 form은 해당 template XML 문서의 디스플레이 정보를 담고 있으며, 사용자 입력을 받기 위한 입력 양식을 포함하고 있어야 한다. 사용자 입력 양식으로는 버튼 입력 양식, 체크박스 입력 양식, 패스워드 입력 양식, 라디오 입력 양식, 텍스트 입력 양식, 선택 입력 양식 등이 있으며, 이외에도 입력된 정보의 전송과 다시 입력을 지원하기 위한 submit 입력 양식과 reset 입력 양식이 포함된다. 다음은 다양한 입력 양식을 지원하기 위한 XSL 문서의 작성 예를 보여주고 있다.

- **텍스트 입력 양식**

```
< xsl : template match = "XML_TAG_NAME" >
        < input type = "text" name = "TXT1" size = "30%"
            value = "{text()}"/ >
< /xsl : template >
```

- **라디오 입력 양식**

```
< xsl : template match = "XML_TAG_NAME" >
      < input type = "radio" name = "RAD1" value = "M"/ > MAN
      < input type = "radio" name = "RAD1" value = "F"/ >
          WOMAN
< /xsl : template >
```

- **선택 입력 양식**

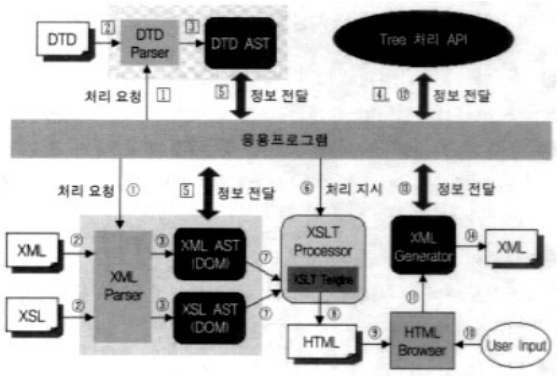
```
< xsl : template match = "XML_TAG_NAME" >
    <select name = "SEL" >
```

```
        <option value = "A"> DATABASE </option>
        <option value = "B"> SYSTEM SOFTWARE </option>
        <option value = "C"> ETC. </option>
    </select>
</xsl : template>
```

위의 예제에서 나타나는 HTML 입력 태그를 포함한 모든 입력 태그들은 HTML <FORM> 태그 속에 포함되어야 하며, <FORM> 태그 속의 마지막 부분에는 사용자 입력 내용의 전송과 재 입력을 위한 버튼 입력 양식이 정의되어야 한다.

3.3 XML 문서 처리 전체 흐름도 및 구현 화면

본 논문에서 제안한 트리/form 기반 XML 문서 편집 시스템은 크게 DTD 파서, XML 파서, XSLT 처리기, HTML 브라우저, XML 생성기, 트리 처리 API 모듈로 구성되어 있다. (그림 6)은 시스템을 구성하는 각 요소와 실제 XML 문서 처리를 위한 요소들 간의 상호 관계 및 전체 흐름을 보여주고 있다.

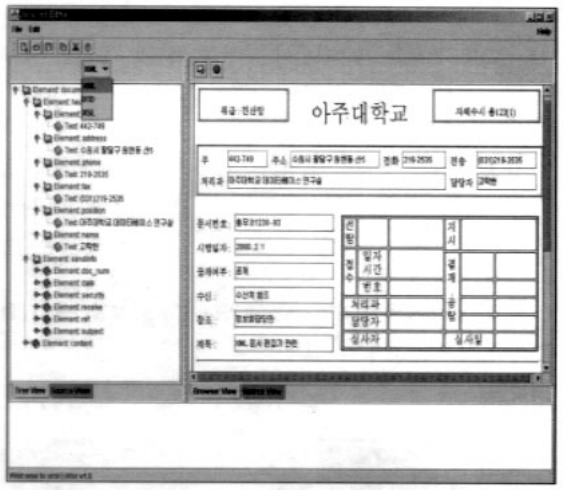


DOM : Document Object Model
AST : Abstract Syntax Tree

(그림 6) XML 문서 처리 전체 흐름도

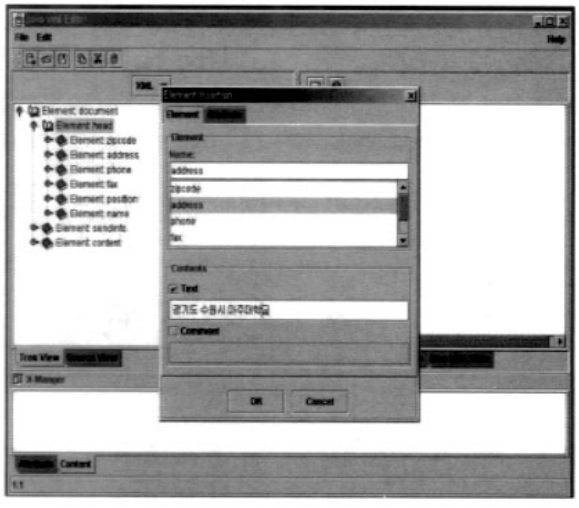
(그림 6)에서 화살표를 따라 표시된 숫자([1], …, ①, …)는 form을 기반으로 하는 XML 문서의 작성시 흐름 순서를 나타내고 있다. 트리 처리를 위한 API에는 트리의 생성, 저장, 추가, 삭제, 확대, 축소, 선택 등을 위한 기능이 제공되며, XML 파서를 통해 생성된 DOM은 트리 처리 API를 통해 애플리케이션 상에서 사용되는 XML AST(Abstract Syntax Tree)로 만들어진다. 또한 트리 방식에서 XML 문서를 작성할 경우, well-formed XML 문서의 작성은 (그림 6)에 나타난 처리 순서 중 [4], [5]에 의해 이루어지며, DTD 문서에 유효한 XML 문서는 [1], [2], [3], [4], [5]의 순서로 작성되어진다.

(그림 7), (그림 8)은 제안된 XML 문서 편집기의 화면 구성을 나타내고 있다. (그림 7)은 form 양식을 지원하고 있는 에디터 화면을 나타내고 있으며, 화면의 왼쪽 창은 DTD, XML, XSL 문서를 파싱하여 그 결과를 JTree 형태로 보여주며, 오른쪽 창은 HTML 브라우저를 통하여 HTML 문서를 form 형태로 나타내주고 있다. 이때 브라우저를 통해 보여지는 HTML form은 template XML 문서의 엘리먼트가 가지는 텍스트 값을 같이 읽어와서 브라우징하고 있다.



(그림 7) Form 양식을 지원하는 XML 문서 편집기 화면구성

(그림 8)은 트리 구조를 기반으로 하여 XML 문서를 제작하는 화면을 보여주고 있다. 특정 엘리먼트를 선택한 후 엘리먼트를 추가, 삭제, 수정하는 기능을 지원하고 있으며, 엘리먼트의 추가 및 수정 시 편집 가능한 엘리먼트 속성으로는 엘리먼트 이름, 텍스트 값, 애트리뷰트 값 등이 있다. 또한 DTD에 유효한 XML 문서를 제작하고자 할 때, 올바른 엘리먼트와 애트리뷰트 이름을 선택하게 하기 위해서 본 시스템은 DTD AST의 정보를 참조하게 되며 유효한 엘리먼트와 애트리뷰트만을 사용자에게 보여준다.

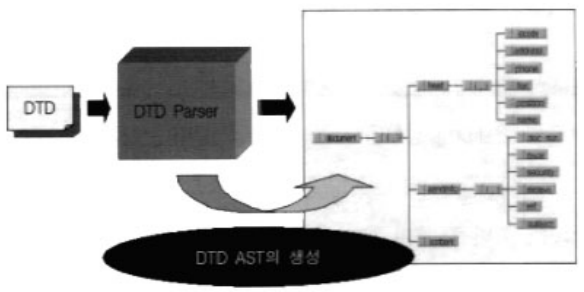


(그림 8) 트리 기반의 XML 문서 엘리먼트 생성 화면

4. 시스템의 주요 기능

4.1 DTD AST의 생성

기존의 XML 파서가 DTD를 이용한 XML 문서의 validation 기능은 제공하고 있지만, DTD 문서의 비 XML이라는 특성은 XML 파서를 통해 DTD에 대한 실제적인 정보를 제공받는 데 많은 한계점을 드러내고 있다. 즉 XML 파서는 XML 문서를 개조하거나 수정할 때에만 중요한 것이기 때문이다. 하지만, DTD를 가지는 XML 문서의 효율적인 생성을 위해서는 DTD에 대한 구조적 정보를 필요로 하게 된다. 따라서 본 시스템에서는 별도의 DTD 파서를 통해 DTD 문서를 파싱하여 DTD AST(Abstract Syntax Tree)라는 트리를 생성한다. DTD 문서를 통해 (그림 9)와 같은 DTD 트리를 생성함으로써, 구조적 정보를 가지는 XML 문서 상에서 엘리먼트를 추가하거나 삭제할 때 DTD의 구조 정보를 얻을 수 있도록 하였다.

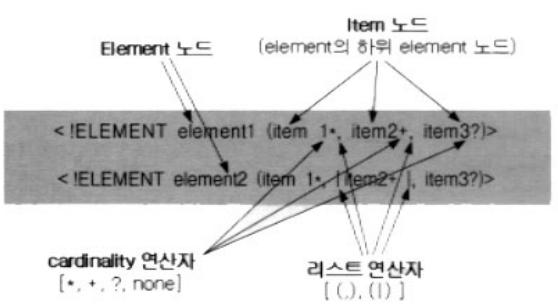


(그림 9) DTD AST의 생성

DTD 구조 정보를 얻기 위한 DTD AST는 다음과 같은 방법을 이용하여 구현하였다.

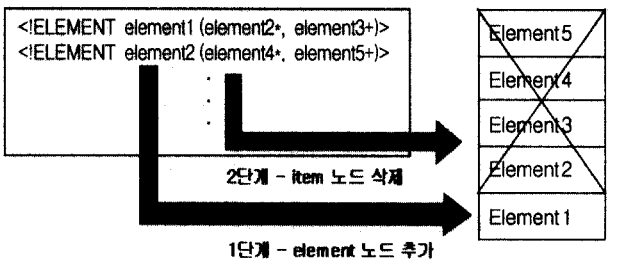
• DTD AST 생성 알고리즘

DTD AST 생성 알고리즘의 설명을 위하여 (그림 10)과 같이 DTD 엘리먼트 태그의 요소들을 정의하였다. 엘리먼트 노드는 정의하고자 하는 엘리먼트를 나타내며, 해당 엘리먼트의 하위 엘리먼트인 item1, item2, item3는 item 노드라고 정의하였다. 리스트 연산자에는 (,), (|)이 있으며, (,)는 and 연산을 나타내고 (|)는 or 연산을 나타낸다. 끝으로 아이템 노드의 반복회수를 정의하는 카디널리티 연산자에는 *, +, ?, none이 있다.



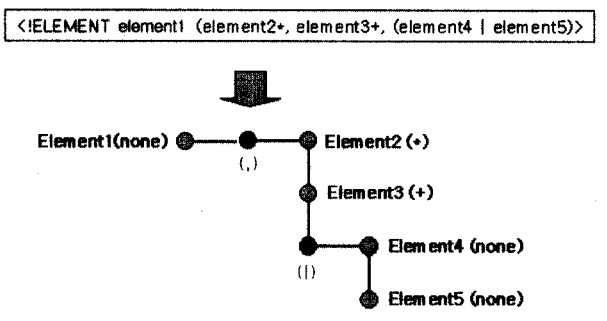
(그림 10) DTD element의 정의

● 최상위 루트 노드를 찾아 부모 엘리먼트 노드에 삽입한다. 최상위 루트 노드를 찾는 과정은 크게 2가지 단계로 이루어진다, 1단계에서는 DTD 문서가 가지는 모든 엘리먼트 노드를 해쉬 테이블에 추가하게 되고, 2단계에서는 추가된 모든 엘리먼트 노드들의 아이템 노드들을 삭제하게 된다. 이러한 과정을 거쳐 해쉬 테이블에 남게 되는 마지막 노드가 루트 노드가 된다. (그림 11)



(그림 11) DTD AST 생성을 위한 루트 노드 찾기

● 선택된 부모 엘리먼트가 가지는 모든 아이템 노드들을 추출한다(그림 12).
● 추출된 각각의 아이템 노드들에 대하여 그들의 카디널리티 연산자 정보를 가지고 새로운 자식 노드를 생성한다(그림 12).
● 새로이 생성된 자식 노드들은 그들의 리스트 연산자의 자식노드에 추가된다. 이때 재귀적인 호출을 통해 리스트 연산자의 중복을 허용한다(그림 12).
● 리스트 연산자를 부모 노드에 추가한다(그림 12).
● 새로이 생성된 모든 자식들을 부모 엘리먼트 노드로 치환하면서 과정 ●-●를 반복한다.

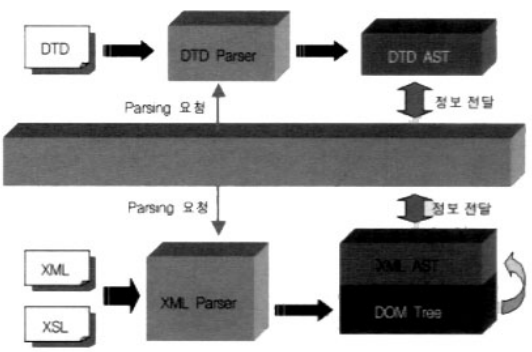


(그림 12) DTD 문서를 통한 엘리먼트 노드의 트리 생성

4.2 XML 파서

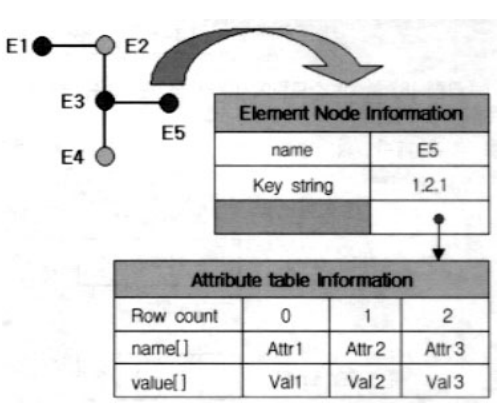
본 시스템에서는 XML 문서의 유효성 검증을 위하여 JAXP1.1을 사용하여 XML 문서의 파싱을 구현하였고, XML 문서외에도 XSL 문서를 파싱할 때에도 사용된다.

(그림 13)은 DTD와 XML 그리고 XSL 문서의 파싱에 수반되는 처리 단계들을 보여주고 있다. 애플리케이션 상에서 접근되는 XML AST와 XML 파서 사이의 중간 객체 모델로는 DOM을 사용하였으며 XML AST는 Java의 Swing 컴포넌트 중의 하나인 JTree를 사용하여 구현되었다. 즉, XML 파서를 통해 파싱된 XML 문서는 DOM 트리로 만들어지며, 이러한 DOM 트리는 사용자 애플리케이션 상에서 효율적으로 처리되기 위해 JTree로 변환되어진다.



(그림 13) DTD 및 XML 파서 구성도

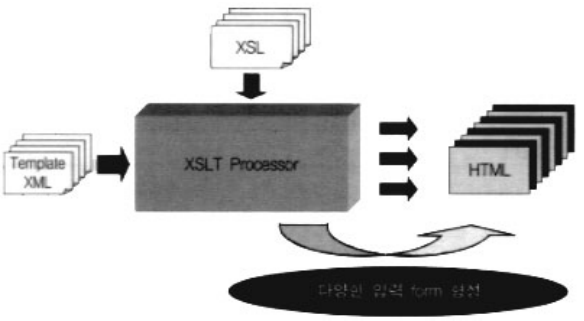
JTree로 표현되어진 XML AST의 각 노드는 해당 노드의 이름과 노드의 고유한 경로 정보를 가지는 노드의 키 값, 애트리뷰트 테이블의 참조 값 등의 정보를 포함하고 있다. 여기서 노드의 키 값은 새로운 XML 문서를 생성할 때 DOM 트리를 탐색하기 위해 사용되어진다. 또한 각 노드가 가진 애트리뷰트들의 정보를 저장하기 위한 테이블은 애트리뷰트의 이름 배열과 애트리뷰트의 값 배열, 애트리뷰트의 개수 등으로 구성되어진다. (그림 14)는 XML AST를 구성하기 위한 엘리먼트 노드와 애트리뷰트 테이블의 자료 구조를 나타내고 있다.



(그림 14) XML AST를 구성하기 위한 노드의 자료구조

4.3 XSLT 처리기

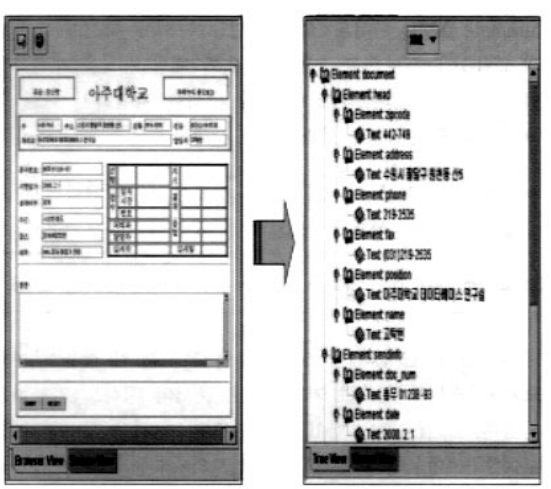
XML 파서를 통해 로딩된 XML 문서와 XSL 문서는 XSLT 처리기를 통해 사용자 입력 form에 해당되는 HTML 문서로 변환된다. (그림 15)와 같이 사용자는 다양한 XSL 문서나 template XML 문서를 받아들여 XSLT 처리를 수행하며 그 결과로 다양한 형태의 입력 form이나 display 결과를 자체 브라우저를 통해 확인해볼 수 있다.



(그림 15) XSLT Processor를 통한 다양한 입력 form 생성

4.4 HTML 브라우저

본 시스템에서 사용하는 HTML 브라우저는 크게 두 가지 기능을 수행하도록 설계되어 있다. 그 중 하나는 HTML 문서의 디스플레이 기능으로 XSLT 처리기를 통해 생성된 HTML 문서를 디스플레이 하는 데 사용된다. 브라우저의 두 번째 기능은 form 이벤트 처리 기능이다. 다양한 입력 양식의 정의와 form 이벤트의 처리를 통하여 사용자 입력을 XML 생성기에 넘겨줄 수 있다. (그림 16)은 브라우징을 통한 사용자 입력 form의 브라우징과 form 상에서 사용자가 입력한 데이터를 이용하여 작성된 XML 문서를 나타낸다.



(그림 16) 브라우저에서 입력 form을 통한 XML 문서 생성

<표 2>는 브라우저가 처리할 수 있는 다양한 form 이벤트 생성을 위해 사용되는 HTML 태그와 애트리뷰트 타입을 보여준다. 본 문서 편집기에서 구현되어진 HTML 브라우저는 HTML 스펙에서 제안된 입력 양식들을 대부분 지

원할 수 있도록 설계되어졌다.

〈표 2〉 브라우저가 처리하는 form event 생성 HTML 태그

Tag name	Tag Type	Tag name	Tag Type
Input	button	Input	reset
Input	checkbox	Input	submit
Input	Image	Input	text
Input	password	Select	
Input	radio	TextArea	

4.5 XML 문서 생성

Form을 기반으로 하는 XML 문서의 작성을 위해 다음과 같은 두 가지 매핑 방안을 고려해 볼 수 있다. 우선 기본적인 매핑 방안으로 template XML 문서의 엘리먼트 명(name)에 기초한 매핑이다. 이러한 매핑을 지원하기 위해서 브라우저 상에서 발생되는 form event는 아래 (1)에 나타난 형태로 XML 생성기에 전달된다.

• 브라우저 발생 이벤트(엘리먼트 이름을 사용한 매핑) - (1)

```
element_name1 = value1&element_name2
              = value2&element_name3 = value3
```

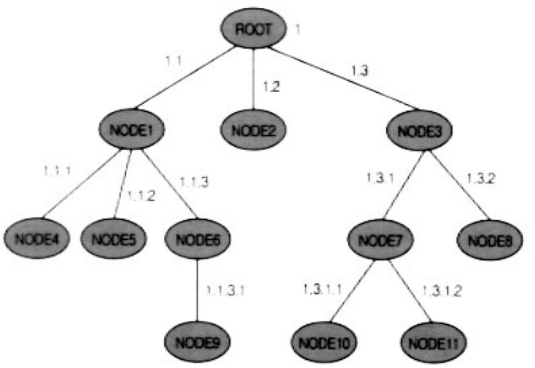
(1)과 같이 발생된 form event에 근거하여 각 엘리먼트 명에 대한 value 매핑으로 새로운 XML 문서가 생성된다. 하지만 이러한 방법을 이용하는 XML 문서 생성기는 중복된 태그 명을 가지는 XML 문서의 생성 시 많은 문제점을 내포하게 된다. 이에 대한 해결 방안으로 본 시스템에서는 template XML 문서의 각 엘리먼트에 이름 대신에 고유한 키 값(ID)을 부여하고 그것에 기초한 매핑 방식을 채택하였다. 고유한 키 값에 기초한 매핑 방식을 지원하기 위해 발생되는 form event는 아래 (2)와 같다.

• 브라우저 발생 이벤트 (고유 경로에 따른 엘리먼트 키 값을 사용한 매핑) - (2)

```
element_id1 = value1&element_id2 = value2&element_id3 = value3
```

Template XML의 각 엘리먼트에 고유한 키 값을 부여함으로써 중복된 태그를 가진 XML 문서의 생성도 가능하게 하였다. (그림 17)은 XML 엘리먼트에 키 값을 부여한 예를 보여주는 데, 키 값은 각 엘리먼트의 고유 경로를 근거로 하여 부여된다.

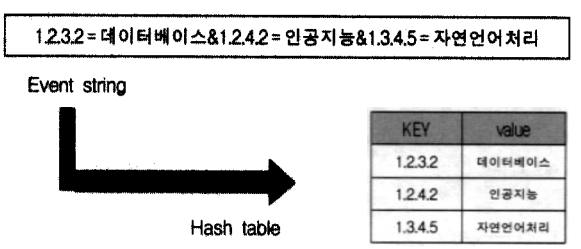
그림에서 루트(root)노드에 대한 키 값을 1로 정의하였고, 자식(child) 노드들의 키 값을 부여하기 위해 재귀 호출을 통한 깊이 우선 탐색(depth-first search) 방식으로 모든 노드들을 순회한다. 이를 통해 탐색된 모든 자식 노드들의 키 값은 루트 노드로부터의 고유한 경로에 기초하여 부여된다. 위와 같이 고유한 경로를 기초로 하여 template XML 문서의 각 엘리먼트들에 대한 키 값을 부여함으로써, form 양식을 통해 얻어진 엘리먼트와 엘리먼트가 가지는 데이터를 매핑시킬 때 엘리먼트의 탐색 비용을 최소화시킬 수 있다. 즉, 엘리먼트의 키 값 속에는 루트 노드에서부터 해당 엘리먼트로의 경로 정보가 포함되어 있고, 그런 키 값을 통한 엘리먼트의 탐색은 루트 노드에서부터의 최소 경로가 된다.



(그림 17) 고유한 경로에 근거한 XML 엘리먼트의 키 값 부여

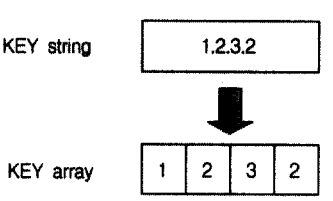
• XML 문서 생성 알고리즘

● HTML 브라우저에서 발생된 Event 스트링에서 키 값과 테스트 값을 추출하여 해쉬 테이블을 생성한다(그림 18).



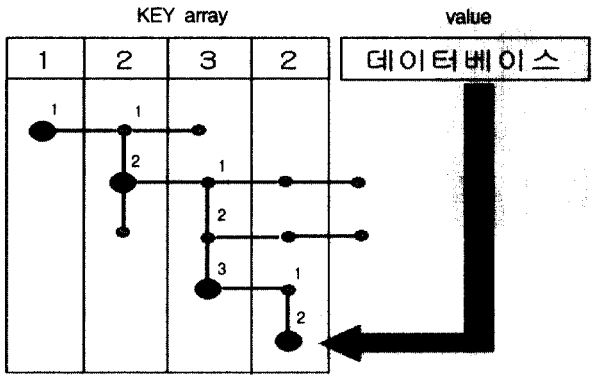
(그림 18) event 스트링에서부터 생성된 해쉬 테이블

● 해쉬 테이블의 키 스트링 값을 추출하여 키 배열을 생성한다(그림 19).



(그림 19) 키 스트링 값의 키 배열 매핑

● 탐색 노드의 최소 경로 정보를 가진 키 배열 값을 이용하여 template XML 문서 트리를 탐색하며 그 노드의 텍스트 값을 변환한다(그림 20).



(그림 20) 키 배열을 이용한 트리 탐색 및 텍스트 값 매핑

5. 결론 및 향후 연구 과제

본 논문에서 제안된 XML 문서 편집기는 기존의 트리 방식에만 한정되었던 XML 문서 작성에서 벗어나 form 방식에서도 XML 문서의 작성이 가능하도록 설계되었다. 개발된 XML 문서 편집기는 XML에 익숙하지 않은 사용자의 경우에도 더욱 쉽고 편리하게 XML 문서를 작성하게 해 줄 것으로 기대된다. 지금까지 기술한 본 문서 편집기의 특징을 요약하면 다음과 같다.

- 새로운 XML 문서의 생성을 위하여 기존의 트리 구조 방식뿐만 아니라 form 양식을 통한 입력 및 수정을 가능하게 하였다.
- 사용자 입력 폼의 생성을 위하여 template XML 문서와 XSL 문서를 사용하였다.
- 최종적으로 생성되는 사용자 입력 폼 문서는 HTML 문서이며, 이는 내장된 브라우저를 통하여 사용자에게 보여지게 된다.
- form 인터페이스를 통해 생성된 새로운 XML 문서는 다시 새로운 XML 문서 생성을 위한 template XML 문서로 사용되어질 수 있다.
- 임의의 template XML 문서에 다양한 형태의 XSL 문서 적용을 지원하며, 이를 통해 다양한 입력 폼을 사용자에게 제공해 줄 수 있다.

위와 같이 설계된 문서 편집기는 기업이나 관공서 등과 같이 이미 정형화된 양식의 문서를 사용하는 곳에서 이러한 문서 form을 통해 다량의 XML 문서를 작성할 때에 많은 장점을 발휘할 수 있을 것으로 기대된다. 또한 고정된 form 생성의 관점에서 접근하기보다는 다양한 양식의 form을 지원할 수 있도록 XSLT 기술을 가지고 설계되었다. 이것은 곧 기업체의 문서 양식의 변화가 애플리케이션에 영향을 주지 않으며, XSL 문서의 수정을 통해 다양한 입력 form 생성이 가능하다는 잇점이 있다. 하지만 입력 form의 생성을 위해 template XML 문서와 XSL 문서를 필요로 하기 때문에, 초기 template XML과 XSL 문서의 제작이라는 부담이 남는다. 현재 본 문서 편집기에서는 이러한 초기 template XML 문서와 XSL 문서의 생성 기능을 트리 구조에 기반 한 인터페이스를 통하여 제공하고 있다. 하지만 좀 더 효율적인 문서 편집 기능을 제공하기 위해서 다음과 같은 기능의 보완이 이루어져야 하겠다.

- DTD 문서의 분석을 통한 자동적인 template XML 문서 생성 기능
- HTML form의 효율적인 생성을 위하여, GUI(Graphic User Interface) 상에서의 drag & drop 방식을 통한 XSL 문서의 자동 생성 기능

이러한 기능의 보완과 함께 향후 연구 과제로는 데이터베이스와의 상호 연계를 위해 DTD-to-Schema와 XML-to-Database에 대한 연구가 계속 진행되어야 할 것이다.

참 고 문 헌

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

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

[3] James Clark, XSL Transformations(XSLT) Version 1.0, W3C Recommendation, November, 1999. http : //www.w3.org/TR/xslt.

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

[5] Java API for XML Processing 1.1 Specification, 08-May-2001. http : //java.sun.com/xml/xml_jaxp.html.

[6] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Extensible Markup Language (XML) 1.0, W3C Recommendation, 10 February, 1998, http : //www.w3.org/TR/REC-xml.

[7] Benoit Marchal, Guidelines for using XML for Electronic Data Interchange, Version 0.05, 25th January, 1998. http : //www.eccnet.com/xmledi/guidelines-styled.xml/.

[8] David Hunter, Beginning XML, 2000.

[9] Alexander, Tom, Professional Java XML Programming, 2000.

[10] Kevin Williams Professional XML Databasees, 2000.

[11] 고윤희, 김현철, 이원규, 고려대학교 컴퓨터교육과, "데이터베이스와의 연동 기반 XML 에디터의 설계 및 구현", 한국정보과학회, 2001봄 학술발표논문집(B) 제28권 제1호, 2001.

[12] 김동욱, 최한석, 목포대학교 컴퓨터과학과, "XSL을 지원하는 XML문서 편집기 설계 및 구현", 목포대학교 정보산업연구지, 제7권, pp.35-44, 1999.

[13] 김길준, 정용득, 숭실대학교 컴퓨터학부, "XML 저작도구의

설계", 정보산업기술연구(제3집), 1998.
[14] XForms-The Next Generation of Web Forms, http : // www.w3c.org/MarkUp/Forms/, October, 2001.
[15] Frank Boumphrey, Propessional XML Application, 1999.

고 탁 현

e-mail : sanha@madang.ajou.ac.kr
2000년 영남대학교 컴퓨터공학과(학사)
2000년~2002년 아주대학교 정보통신 전문대학원(석사)
관심분야 : 데이터베이스, 멀티미디어 시스템, XML 응용

황 인 준

e-mail : ehwang@madang.ajou.ac.kr
1988년 서울대학교 컴퓨터공학과(학사)
1990년 서울대학교 컴퓨터공학과(석사)
1988년 Univ. of Maryland at College Park 전산학과(박사)
1998년~1998년 Hughes Research Lab. 연구교수
1998년~1999년 Bowie State Univ., Assistant Professor
1999년~현재 아주대학교 정보통신전문대학원 조교수
관심분야 : 데이터베이스, 멀티미디어 시스템, 정보 통합, 전자상거래, XML 응용