XSL-fo를 적용한 XML 문서표현 시스템의 설계 및 구현

(Design and Implementation of XML Document Presentation System applying XSL-fo)

김 진 수 † 강 치 원 †† 류 근 호 ††† 정 회 경 ††††
(Jinsoo Kim) (Chiwon Kang) (Keunho Ryu) (Hoekyung Jung)

요 약 본 논문은 XML 문서의 내용 및 구조 정보를 XSL 스타일시트(stylesheet)의 포맷팅(formatting) 정보를 적용하여 표현하는 포맷팅 시스템의 설계 및 구현에 관한 것이다. 본 시스템은 XML 문서를 XSLT(XSL Transformations) 및 Xpath(XML Path Language)를 이용하여 문서를 변환하고, XSL-fo(XSL Formatting Objects)를 적용하여 포맷팅을 지정하는 XML 문서 표현 시스템을 설계 및 구현하였다. 이 XML 문서 표현 시스템은 웹 표준화 기구인 W3C에서 제안하는 XSL 포맷팅 처리에 대한 구성을 기반으로 구현함으로써 표준화에 입각한 처리시스템으로써 변화에 능동적으로 대처 가능하고 모듈화 되어 있어 부분적인 수정 및 대체가 가능하도록 설계하였다. 본 시스템은 IBM 호환 PC에서 동작하며, 운영체제는 Windows 2000 환경에서 Visual C++6.0을 사용하여 개발하였다

Abstract This paper is on the design and implementation of formatting system to display content and structured data of XML document using formatting information of XSL. This XML document presentation system transformed XML document using XSLT and Xpath, applied XSL-fo to formatting process. This System is based on XSL formatting process recommended by W3C.

This System developed using Visual C++6.0 on Windows 2000 operating system and IBM PC.

1. 서 론

기존의 문서 편집시스템으로 작성된 문서는 각기 독자적인 문서구조와 다양한 내용 정보를 갖고 있으므로 서로 다른 시스템 및 장치를 갖는 문서처리 환경에서 이들 문서의 교환 및 공유를 위한 표준 문서구조로서 SGML(Standard Generalized Markup Language, ISO 8879 : 1986)을 국제표준으로 제정하였으며, 최근 인터넷의 등장으로 SGML과 같은 구조적인 문서의 교환에 관한 필요성이 급증하게 되자 W3C(World Wide Web Consortium)에서는 XML(eXtensible Markup Language)을 새로운 인터넷 표준으로 권고하게 되었다[1,2].

XML은 SGML과 마찬가지로 임의 형태 문서, 임의 응용에 대해 일반화 마크업(Markup)을 정의하기 위한 방법을 표준화하는 메타 언어(Meta Language)로서 기술적 문서 마크업을 생성하는 기법과 문서의 지능적 내용이나 체계를 정의하기 위한 언어를 제공하며, 다양한 응용들 사이에 구조화된 데이타를 상호 교환하기 위한 도구와 다양한 입력으로부터 출력의 내용, 구조, 조직을 표준화하기 위한 방법을 제공한다.

그러나, XML 문서는 문서의 논리구조만을 가지므로 여기에 문서 스타일에 대해 기술하는 요소간의 관계를 나타내고 논리 요소와의 대응관계를 기술하기 위해 W3C에서는 XSL(eXtensible Stylesheet Language)을 권고하였다[4]. 이는 인터넷에서 문서교환을 위해 만들어진 XML에 적합하도록 설계되었으며, XML 문서를 사용자에게 표현하기 위해 포맷 처리를 하고, 이렇게

• 이 논문은 과학재단 98특정기초연구과제 연구비에 의하여 연구되었음.
† 정 회 원 : 배재대학교 컴퓨터공학과 교수
jskim@mail.paichai.ac.kr
†† 정 회 원 : 배재대학교 컴퓨터공학과
cwkang@kdn.com
††† 종신회원 : 충북대학교 컴퓨터과학과 교수
khryu@dblab.chungbuk.ac.kr
†††† 종신회원 : 배재대학교 컴퓨터공학과 교수
hkjung@mail.pcu.ac.kr
논문접수 : 2001년 2월 21일
심사완료 : 2001년 4월 5일

포맷된 문서는 장치에 의존하지 않게 기술되어 인쇄장치, 표시 장치에 표현된다.

본 논문은 XML 문서의 내용 및 구조 정보를 XSL 스타일시트의 포맷팅 정보를 적용해 표현하는 포맷팅 시스템의 설계 및 구현에 관한 것이다. 구조화된 XML 문서를 XSLT 및 XPath를 이용하여 문서를 변환하고, XSL-fo를 적용하여 포맷팅을 지정하는 XSL 스타일시트를 입력으로 포맷팅 처리하기 위한 XML 문서 표현 시스템을 설계하고, XML 문서를 사용자에게 표현하거나 포맷팅 된 XML 문서를 상호 교환할 수 있도록 하여 XML 문서처리 환경 구축에 기여하고 멀티미디어 문서처리 시스템 기술 개발 촉진에 기여하리라 본다.

2. XML / XSL 개요

2.1 XML

XML은 인터넷 상에서 데이타 교환을 위한 목적으로 모든 문서 및 응용에 대한 범용 마크업을 정의하는 방법을 표준화한 메타 언어이다[2]. 이는 OSI 환경에서 이 기종간의 효율적인 문서정보 교환을 위한 국제 표준인 SGML을 인터넷 상에서 활용 가능하도록 적용한 것이다. XML은 SGML에 기반을 두어 매우 유연하며, 인터넷 언어인 HTML(HyperText Markup Language, W3C)이 가지는 단순함을 포함한다. 또한, XML은 1996년 W3C의 XML 워킹 그룹(Working Group)에 의해 표준화된 텍스트 형식으로 설계된 이래 체계적이고 일관적 접근방식을 정의하며, 많은 기술을 포괄할 수 있는 기본 틀(Framework)을 제공하고 있다.

이는 기존의 HTML의 한계와 SGML의 복잡함을 해결하며, 구조화된 정보를 사용자들 간에 효율적인 정보교환과 범용적인 시스템으로 활용 가능한 효과적인 수단이 되고 있다.

XML은 크게 문서의 논리적 구조를 정의하는 문서형 정의부(Document Type Definition : DTD)와 문서 형 정의에 따라 작성하는 문서 실례부(Document Instance : DI)로 구성된다.

XML 문서는 문서 실례부의 내용 구성에 따라 잘 구성된 문서(Well-formed Document)와 유효한 문서(Valid Document)로 나뉜다. 잘 구성된 문서는 XML의 기본 규약을 준수하여 만들어지며, 문서형 정의부의 구조에 따라 작성되었는지 정확히 검증하지는 않는다. 유효한 문서는 반드시 문서형 정의부에 기술된 구조에 따라 작성된 문서이며, 문서형 정의부에 따라 정확히 작성되었는지 검증되는 문서를 말한다[2].

이와 같은 XML 문서는 엘리먼트(Element), 엔터티(Entity), 애트리뷰트(Attribute)로 이루어진다. 엘리먼트는 문서의 논리적 구조를 구성하는 기본 단위로서 계층적인 관계로 문서를 표현하며, 애트리뷰트는 엘리먼트의 추가적인 정보를 기술하기 위해 사용한다. 엔터티는 문서에서 처리되는 최소 처리 단위이며, 저장 단위로서 XML 처리기에 의해 해석된다.

2.2 스타일시트와 XSL

2.2.1 스타일시트 개념

스타일시트는 문서를 화면상이나 출력물로 표현 할 수 있도록 방법을 기술한다. 즉, XML은 논리적인 구조의 요소들로 기술되어 데이타의 의미를 가지는 문서로 이러한 데이타 중심적인 문서를 스타일시트라는 포맷팅 방법 및 스타일링 방법이 기술된 문서를 적용하여 원하는 결과물을 얻을 수 있다.

사용자나 저자가 데이타 문서에 새로운 추가 사항 없이 장치 독립적인 기술로 표현이 가능하다는 스타일시트가 갖는 장점으로 인해 수 많은 스타일시트들이 생겨나게 되어 서로 다른 기술 방법과 표현 정보 등이 중복되거나 특수성을 지니게 되었다[3].

이에, 1996년 SGML의 논리적인 구조 기술 요소에 대응하는 문서 스타일 요소간의 관계를 나타내고 논리 요소와의 대응 관계를 기술하는 DSSSL[4]을 스타일에 관한 국제 표준으로 제정하였으며, 또한 인터넷 언어인 HTML의 추가적인 스타일 정보와 스타일시트로서 적용 가능한 CSS(Cascading Style Sheets, W3C)를 인터넷 표준으로 제정하였다[5,6,7].

2.2.2 XSL 구성

XML은 문서의 논리적인 구조 정보와 내용만을 가지므로 이를 표현하기 위한 스타일시트가 필요하게 되었다. 이에 DSSSL의 자유로운 표현과 CSS의 간결한 사용구문을 반영하여 XSL을 XML의 구조적인 정보를 표현하기 위해 제정하였다[8].

XSL은 크게 XSLT와 XPath 그리고 XSL-fo로 구성되어 있다. XSLT는 문서 변환 언어로서 XML 문서를 다른 문서 형태로 변환하거나 재구성 하는 역할을 담당하고, XPath는 XML 문서의 요소와 대응관계를 표현하기 위한 부분으로 XML 문서 구조에 접근하기 위한 언어이다[9,10]. 마지막으로 XSL-fo는 XML 문서의 논리적 요소에 포맷팅 구문(Formatting Semantics)을 지정하여 원하는 형태의 결과를 얻기 위한 방법을 제공한다. XSL-fo는 온라인(On-line) 상에서 출판자 중심의 포맷팅 처리를 위한 DSSSL의 부분 집합인

DSSSL-O(DSSSL Online Application Profile)의 핵심 흐름 객체들(Core Flow Objects)과 HTML/CSS의 핵심 흐름 객체들을 포함하는 포맷팅 모델을 제시한다[5,6,7,8,11,12].

XSL은 생성 규칙(Construction Rules)으로 이루어지며, XSLT의 요소로 표현한다. 또한 생성 규칙은 패턴(Pattern)과 액션(Action)으로 구성된다. 패턴은 XPath와 같은 패턴 언어로 표현하고 액션은 생성 규칙 내부를 의미하며, XSL-fo나 HTML/CSS의 흐름 객체(Flow Objects) 등의 포맷팅에 관련된 흐름 객체를 포함할 수 있도록 설계되었다[13,14,15,16].

XSL-fo에서는 포맷팅에 관련된 객체들을 흐름 객체와 보조 객체(또는 레이아웃 객체 : Auxiliary Objects, Layout Objects)로 나누고 포맷팅 객체(Formatting Objects)에 포함하도록 한다[11].

XSL-fo의 포맷팅 모델은 크게 블록 수준(Block level)과 인라인 수준(Inline level)으로 나뉘고 테이블, 리스트 등의 포맷팅 객체 들은 이에 종속되는 모델링을 제시한다. 블록 수준 포맷팅 객체는 영역이라는 개념아래 처리되는 특정한 공간이며, 인라인 수준 포맷팅 객체는 영역을 채우기 위한 공간의 연속이다[11].

3. XSL 포맷팅 엔진 설계

본 장에서는 W3C에서 제안한 XSL-fo를 기반으로 작성된 스타일시트와 XML 문서의 입력으로 변환될 결과 트리(Result FO Tree)를 생성하는 XSLT/XPath

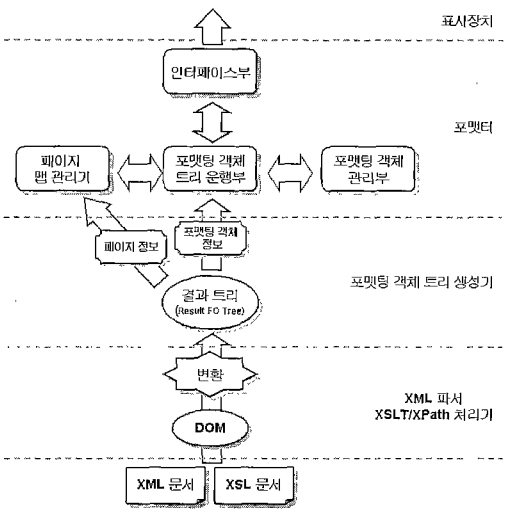


그림 1 전체 시스템 구성도

처리기와 결과 트리를 포맷팅 객체 트리(Formatting Object Tree : FOT)로 변환 하기 위한 포맷팅 객체 트리 생성기, 포맷팅 객체 트리를 포맷팅 처리하는 포맷터로 구성되며, 그림 1은 전체 시스템 구성도를 나타낸다.

본 시스템은 XML 문서와 XSL 스타일시트를 입력으로 변환하여 DOM(Document Object Model) 형태의 결과 트리를 생성하고 변환 결과물은 포맷팅을 위해 객체별로 분류하며, 페이지 정보는 페이지 맵 관리기에서 템플리트(Template)로 저장 후 관리하고, 결과 객체와 속성들은 포맷팅 객체로 변형되어 새로운 트리를 구성한다. 페이지 정보와 포맷팅 객체 트리는 포맷터로 전달되고, 포맷팅 모듈에서 각각의 정보들은 포맷팅 모듈 객체로 저장 관리되어 화면에 디스플레이 되는 과정을 거친다.

포맷팅 시스템은 본 논문에서 제시한 브라우저(Browser) 뿐만 아니라 타 시스템에서도 사용할 수 있도록 DLL(Dynamic Linking Library) 형태로 설계하였으며, 포맷팅 시스템은 클래스(Class) 인터페이스(Interface)를 제공하며, 타 시스템에서도 이를 이용해 트리 형태의 연결 구조나 선형 연결 구조, 또는 두 형태가 혼합된 연결 구조 등의 자료구조를 만들어 포맷팅 시스템에서 제공하는 API(Application Programming Interface)를 통해 포맷팅이 가능하도록 설계하였다.

3.1 포맷팅 객체 트리 생성기 설계

XML 문서와 XSL 스타일시트를 파서 API를 통해 각각 파싱한 결과를 메모리에 저장하고, 저장된 두 DOM 객체는 파서의 변환 API를 통해 새로운 DOM 객체로 생성된다. 새로운 DOM 객체는 변환된 결과로써 포맷팅 객체의 데이타로 입력되는 소스 트리가 된다. 소스 트리는 페이지 정보를 관리하는 페이지 맵 관리기와 포맷팅을 위한 객체를 트리로 만들어주는 포맷팅 객체 트리 생성기로 나뉘어서 처리한다.

3.1.1 템플리트 페이지 맵 관리기

XSL의 포맷팅 객체들 중에서 페이지 관련 객체는 계층적 구성을 가지며, XSL 스타일시트의 최상위 포맷팅 객체로 존재 가능한 fo:root의 하위에 페이지 관련 객체와 실제 포맷팅에 관련된 객체들이 존재한다. 페이지 관련 포맷팅 객체들의 계층적 구성도는 그림 2에 보인다.

페이지 관련 객체들의 최상위 객체인 fo:layout-master-set은 성격상 크게 두 가지로 나뉘는 객체를 포함하며, 실제 화면의 구성과 배치에 관련된 객체로 독립적으로 존재하는 fo:simple-page-master와 하위 객체가 있고, 이를 참조하여 여러 개의 화면 구성을 조합할 수

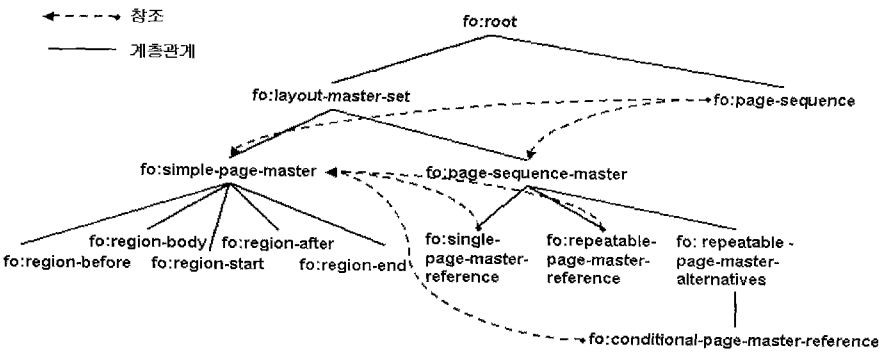


그림 2 페이지 포맷팅 객체들의 계층적 구성도

있는 참조관련 객체 fo:page-sequence-master와 하위 객체가 있다.

실제 포맷팅 관련 객체들의 최상위 객체 fo:page-sequence는 fo:simple-page-master 포맷팅 객체와 fo:page-sequence-master 객체를 참조하여 하위에 객체들이 그려질 페이지 설정을 한다.

fo:simple-page-master 포맷팅 객체는 페이지 설정 이외에 fo:page-sequence-master 객체를 이루는 각각의 특성을 갖는 객체들이 참조하게 된다.

fo:page-sequence는 하나 이상 존재하고, fo:simple-page-master 혹은 fo:page-sequence-master 중 하나를 선택하여 하위 포맷팅 객체들을 처리하기 때문에 이를 기준으로 메모리에 맵으로 관리하도록 제공한다. 즉, fo:simple-page-master와 fo:page-sequence-master의 속성값인 master-name을 키 값으로 하여 메모리에 매핑 후 관리한다.

메모리 매핑을 위한 페이지 관련 포맷팅 객체 클래스 계층도는 그림 3과 같으며, 페이지 관련 포맷팅 객체들

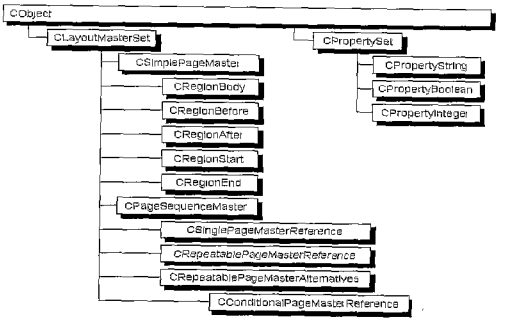


그림 3 메모리 매핑을 위한 페이지 관련 포맷팅 객체 클래스 계층도

의 계층 구조와 발생 빈도 및 메모리 매핑을 고려하여 그림 3처럼 계층 구조의 클래스를 설계하고, 또한 각각의 포맷팅 객체 즉, 클래스들의 특성을 고려하여 속성값을 저장하는 클래스들을 포함하도록 설계한다.

그림 4는 설계한 페이지 포맷팅 클래스가 사용되는 위치와 관리형태에 대한 것으로 페이지 맵 관리기는 master-name의 키 값으로 매핑되는 클래스인 CSimple PageMaster와 CPageSequenceMaster의 객체를 관리하고, 포맷팅 시스템의 접근 API를 통해 전달된다.

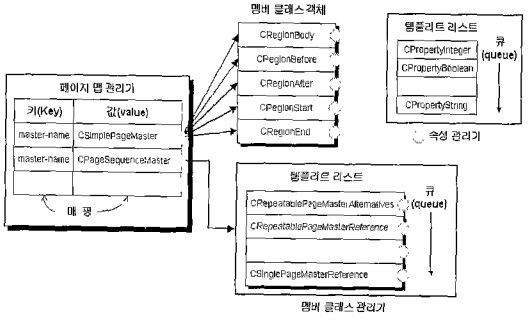


그림 4 페이지 포맷팅 클래스 관리 설계도

3.1.2 포맷팅 객체 트리 생성기

페이지에 연계되는 실제 포맷팅 객체는 region-body와 이를 제외한 4개의 외부 영역으로 나뉘어 존재하게 된다. XSL-fo에서는 fo:static-content와 fo:flow 객체를 두어 이를 구분한다.

fo:flow는 region-body 영역을 의미하며, fo:static-content는 region-before 등의 외부 영역을 의미한다. 그림 5는 페이지를 5개의 논리적 영역으로 구분한 것이다.

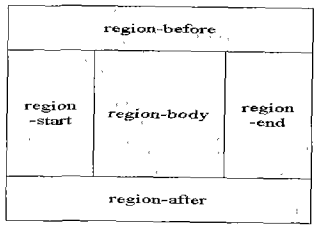


그림 5 페이지내의 영역 분할

각 영역에 존재하는 객체들은 하나의 트리로 구성되며 영역의 특성은 속성값으로 저장한다.

가장 기본 클래스인 CFOT는 관계 클래스 포인터와 클래스의 종류, 속성값을 위한 템플리트 리스트, 그리고 페이지 객체를 참조하기 위한 키 값 등을 기본적으로 저장하도록 설계하였으며, 포맷팅 객체 트리의 각 객체는 CFOT를 상속 받아 설계한 클래스 객체로 구성한다.

관계 클래스 포인터인 CFOT 포인터 형의 객체를 멤버로 포함하는 것은 트리 구조를 이루기 위한 필수 요소이기 때문이다. 또한 각 포맷팅 객체의 속성들이 가질 수 있거나 가지는 특징을 저장하기 위해 적합한 구조인 템플리트 리스트를 멤버로 포함한다. 이는 모든 속성들을 멤버로 구성하는 경우 메모리 적재량의 크기가 늘어나는 점에 착안하여 설계하였다.

포맷팅 객체 트리 생성기에서는 변환된 결과 트리의 포맷팅 객체를 여러 수준으로 나누어 설계한 클래스에 매핑하여 트리로 구성한다. 클래스들은 특징에 따라 CBlock, CInline, CExternalGraphic, CTable, CTable Row, CTableCell, CList 등으로 나뉘며, 모두 CFOT를 상속하여 나타낸다. 이들 클래스의 객체들은 결과 트리의 포맷팅 객체를 탐색하며 분류된 포맷팅 객체와 속성, 데이타 값들과 적절한 매핑 과정을 거친 후 트리로 구성되는 것을 그림 6에 보여준다.

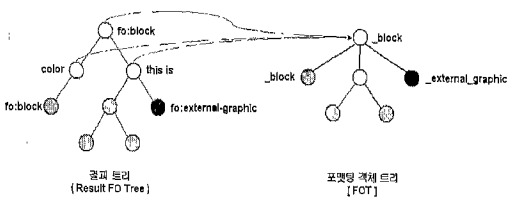


그림 6 결과 트리에서 포맷팅 객체 트리 생성과정

결과 트리는 포맷팅 객체와 속성값, 데이타가 함께 DOM으로 구성된다. DOM 정보는 일반적이고 처리가 복잡하기 때문에 형식적인 트리인 포맷팅 객체 트리 정보로 변환하기 위한 과정을 거치게 된다. 그림 6은 결과 트리의 객체인 "fo:block"과 관련된 속성값인 "color", 데이타인 "this is..."를 하나의 클래스 객체인 "_block"으로 변환하여 새로운 트리를 생성하는 과정이다.

3.2 포맷터 설계

XSL에서는 포맷팅 객체 트리를 포맷터의 입력 값으로 제공하여 포맷팅 결과와 영역들의 계층적인 배치를 생성하도록 규정하고 있다[11]. 또한 XSL의 보편적인 영역 모델은 각 포맷팅 객체들의 의미를 기술 하는데 추상적인 골격을 제시한다.

본 논문에서는 XSL의 기본 취지를 응용하여 블록 영역과 인라인 영역을 처리하는 모듈을 기본으로 하고 특별한 처리는 이를 상속하도록 설계하였다.

그림 7은 포맷팅 객체들을 포맷팅 객체 트리 운행부를 통해 포맷팅 객체 관리부에서 포맷팅 및 영역을 생성하도록 선별하며, 인터페이스부를 통해 화면에 포맷팅된 결과를 보여주는 포맷팅 시스템 설계 구성도이다.

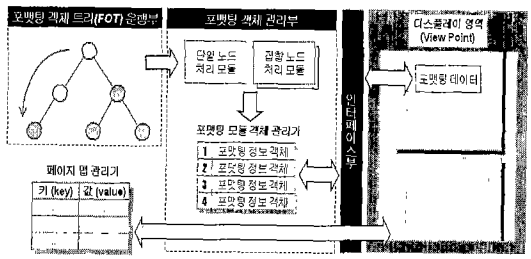


그림 7 포맷팅 시스템 설계 구성도

포맷팅 객체 트리 운행은 포맷터에서 가장 기본적인 모듈로써 입력된 포맷팅 객체 트리에 대해 일반적 트리의 탐색방법에 주로 사용되며, 연속되는 영역을 표현하기 적합한 알고리즘(Algorithm)인 깊이 우선 탐색(Depth First Search : DFS) 방법을 사용하며, 각 포맷팅 객체의 특징에 따라 분류 후 포맷팅 객체 관리부를 호출한다.

포맷팅 객체 관리부는 객체 트리 운행부에서 호출되어, 실제 화면에 표현될 영역과 포맷팅 데이타 객체를 생성 한다. 처리될 포맷팅 객체의 노드 구성에 따라 단일노드 처리 모듈과 집합노드 처리 모듈로 나뉘고, 생성된 포맷팅 정보 객체를 관리하는 역할을 수행한다.

그림 8은 노드 처리 과정을 거쳐 생성된 객체들은 포맷팅 모듈 객체 관리기를 통해 메모리 배열로 관리하는 포맷팅 모듈 객체 관리도 이다. 단일노드 처리 모듈에서는 내부적으로 영역 처리, 인라인 처리, 단락 처리, 폰트

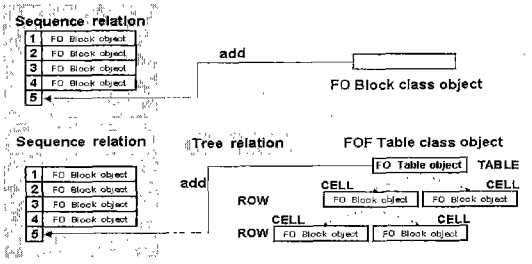


그림 8 포맷팅 모듈 객체 관리도

및 기타 포맷팅 처리를 거쳐 하나의 포맷팅 모듈 객체를 생성하고, 집합노드 처리 모듈에서는 각 노드의 단일 노드 처리 후, 관계 생성과정을 거친 객체군을 메모리 배열로 관리한다. 인터페이스부는 크게 디스플레이 관리 모듈과 뷰포인트(View Point) 영역 관리기로 구성된다. 디스플레이 관리 모듈은 포맷팅 객체 관리부에서 포맷팅 모듈 객체 생성시 영역 등의 포맷팅 정보를 이용하여 블록 및 인라인 영역 정보를 관리 하고, 뷰 영역에 연속적으로 그린다. 또한 페이지 맵 관리기와 연계하여 해당 포맷팅 정보 객체가 표현될 페이지를 관리한다. 뷰포인트 영역 관리기는 뷰 영역의 사용자 이벤트를 감지하여 작동하며, 포맷팅 모듈 객체 관리기에서 해당 포맷팅 정보 객체를 추출하여 뷰 영역에 다시 그려준다.

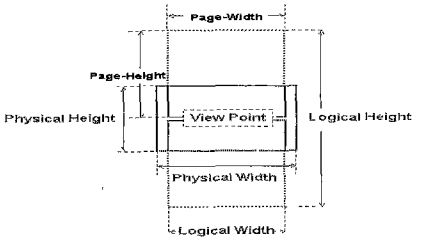


그림 9 영역 처리 개념도

그림 9는 디스플레이 관리 모듈에서 관리하는 영역 정보(Logical Height, Width)와 페이지 정보(Page Height, Width) 그리고, 뷰포인트 영역 관리기에서 처리되는 영역인 뷰포인트의 위치를 보인다.

4. 구현 및 고찰

4.1 구현

본 논문에서 개발한 시스템의 구현환경은 IBM PC 호환 컴퓨터에서 동작하며, Windows 2000의 운영체제 환경에서 구현언어는 Microsoft visual C++ 6.0을 사용하였고, 파서 및 XSLT/XPath 처리기는 Microsoft사의 COM으로 구성된 MSXML을 사용하였다[17]. 본 시스템은 W3C에서 제안한 XSL 1.0 명세서에 기본하여 포맷팅 처리를 하도록 구현하였다[11].

4.1.1 포맷팅 객체 트리의 설계

DOM 형태의 결과 트리를 포맷팅 객체 트리로 변환하기 위한 포맷팅 객체 트리 생성기는 브라우저에 포함된 모듈로서 구현하였다. 페이지 정보를 담고 있는 페이지 맵 관리기는 포맷터로 직접 저장하도록 하였고, 생성될 포맷팅 객체 트리는 포맷터의 API를 통해 최상위 노드만을 전달 하도록 하였다. 스택(Stack)을 구성하여 페이지 정보와 포맷팅 객체 트리 생성시에 상속 값을 처리하도록 설계하였다.

그림 10은 결과 트리에서 페이지 정보는 페이지 맵 관리기에 저장하고, 객체 정보는 포맷팅 객체 트리로 변환하는 과정을 보이며, 상속해야 할 속성값은 스택을 통해 관리 저장된다.

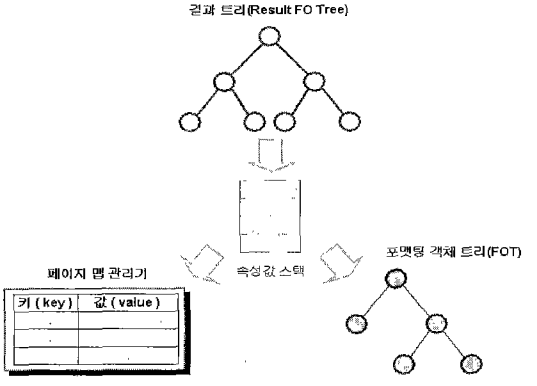


그림 10 포맷팅 객체 트리 생성 과정 구현도

4.1.2 포맷팅 시스템 구현

포맷팅 시스템의 렌더링 엔진(Rendering Engine)은 크게 3개의 클래스로 구성되어 있다. 포맷터의 메인 클래스인 CFormatter는 포맷팅 객체 트리 운행부, 포맷팅 객체 관리부, 디스플레이 인터페이스부를 모두 포함한다. 또한 페이지 맵을 관리하는 템플리트 맵 관리기를 포함한다. 포맷팅 객체 관리부에서는 포맷팅 모듈 객체를 생성하기 위해 CFO_Block과CFO_Table 클래스를 사용한다. 그림 11은 구현된 포맷팅 시스템의 구성 모듈이다.

페이지 정보를 관리하는 페이지 맵 관리기는 MFC에서 제공하는 템플리트 클래스인 CMap을 사용하였으며, 브라우저에서 사용자 이벤트의 윈도우 메시지가 발생하

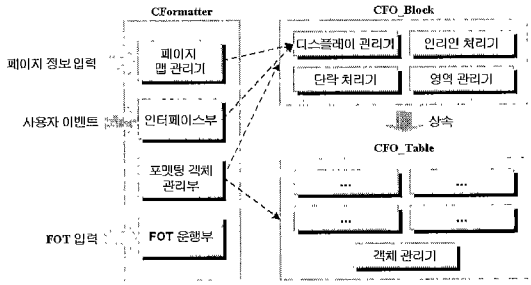


그림 11 포맷팅 시스템의 모듈 구현 구성도

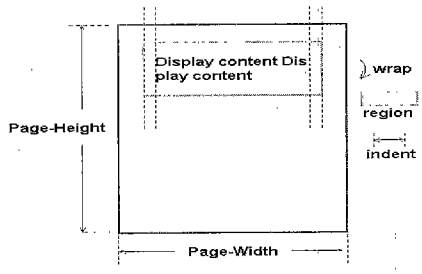


그림 12 블록 처리

면 호출되는 CView의 OnDraw 메소드(Method)를 재정의하여 포맷터의 인터페이스부를 호출하도록 하였다. 인터페이스부에서는 현재 뷰포인트를 찾아 해당되는 포맷팅 모듈 객체를 추출하고, 각 객체의 디스플레이 관리기를 호출하여 화면 뷰 영역에 재 포맷팅 하는 기능을 수행하도록 하였다.

포맷팅 객체 트리 생성부에서 포맷터 API를 통해 생성된 포맷팅 객체 트리의 최상위 노드를 인자 값으로 하여 포맷팅 객체 트리 운행부를 호출하게 된다. 운행부는 탐색을 수행하며, 포맷팅 객체 트리 생성시 저장된 타입에 따라 분류하여 해당되는 포맷팅 객체를 포맷팅 모듈 객체로 변환한다.

포맷팅 객체 관리부에서는 운행부에서 타입에 따라 분류된 포맷팅 객체를 단일 노드 처리 모듈이나 집합 노드 처리 모듈을 거친후 포맷팅 모듈 객체를 생성하여 메모리 배열에 저장한다.

단순 노드 처리 모듈에서는 포맷팅 모듈 객체를 생성시 영역 정보, 속성 정보, 텍스트 데이타 등을 저장하며, 집합 노드 처리 모듈에서는 단순 노드 모듈에서의 처리 모듈과 생성된 포맷팅 모듈 객체들에 관계 형성을 추가하는 모듈과 추가 처리 모듈을 포함하여 구현하였다.

기본 포맷팅 클래스인 CFO_Block은 내부 모듈인 인라인 처리모듈, 단락 처리 모듈, 영역 처리 모듈, 디스플레이 처리 모듈을 포함하며, 블록을 처리한다. 디스플레이 처리 모듈은 다른 모듈들을 호출하여 저장된 정보를 포맷팅 처리하고 화면에 표현한다. 또한 사용자 이벤트에 따른 포맷터의 인터페이스와 연계 처리된다.

그림 12는 CFO_Block 객체 내의 블록 처리를 보여주고 있다. 디스플레이 처리 모듈에서 포맷팅 모듈 객체의 해당 페이지를 표현 후 각 처리 모듈에서 생성된 포맷팅 데이타를 표현한다. 영역 처리 모듈에서는 여백 처리와 영역처리 등을 계산하고, 단락 처리 모듈에서는 속성값을 계산하여 인라인 처리를 한다.

4.1.3 포맷팅 결과

그림 13의 입력 창은 XML 문서의 검증 기능까지도 갖춘다. XML 문서를 선택하면 하단의 오류 보고 창에서는 문서에 오류가 있는지를 보여주고 오류가 없다면 XML document 라는 키워드(Keyword)를 보여준다. 만일 오류가 있다면, 오류 원인, 오류 위치 등의 오류 정보를 보여준다. 또한 반복해서 입력 창을 호출하지 않고 XML 문서와 XSL 스타일시트를 같이 선택할 수 있도록 구현하였다. 마지막으로 선택한 XML 문서나 XSL 스타일시트가 입력 표시 창에 출력되며 모두 입력이 되었다면 곧바로 포맷팅을 시작하게 된다.

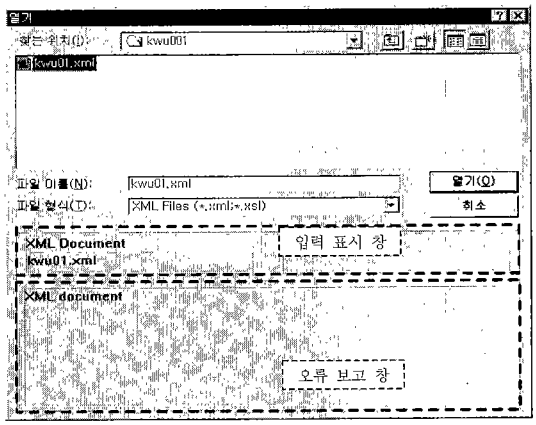


그림 13 입력 및 검증 화면

그림 14는 전체 시스템의 화면 구성을 보이고 있다. 좌측에 보이는 트리 구조 창은 XML 문서와 XSL 스타일시트 및 두 문서를 변환한 결과를 트리 형태로 보여주며 토글(Toggle) 선택으로 숨김 기능을 두었다. 하단의 오류 보고 창은 XML 문서나 XSL 스타일시트가 변환 시에 XSL-fo의 문법에 맞게 작성되지 않았다면 오

류 보고 창으로 오류 내용을 출력한다.

오류 보고 창의 오류를 선택하면 좌측 트리 구조 창에서 위치를 자동적으로 선택하여 원인을 쉽게 알 수 있다. 우측 디스플레이 영역은 실제 포맷팅 되는 부분이다. 문서 입력 후에 디스플레이 영역에 모든 데이타가 출력된다.

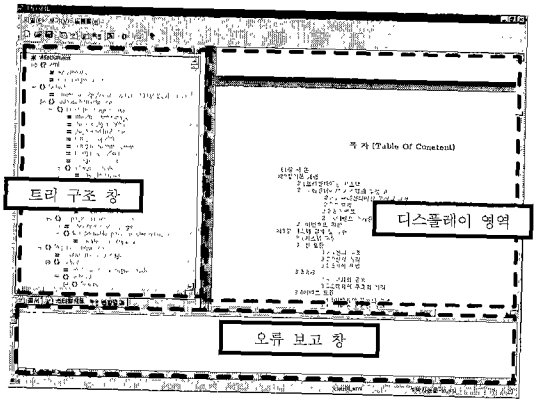


그림 14 시스템 화면 구성

그림 15는 블록 단위의 메모리 포맷팅 결과를 보이고 있다. 블록 주위의 선들은 영역의 범위를 알기 위한 디버깅(Debugging)용으로 사용하도록 토글 버튼을 두었다. 데이타나 그림의 위치정보는 블록 포맷팅 모듈을 거쳐 하나의 포맷팅 단위로 관리되는 것을 보인다.

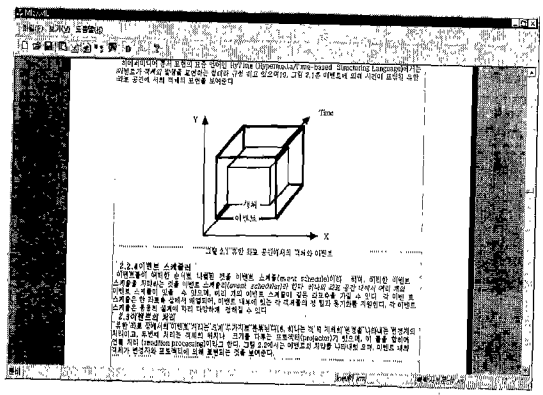


그림 15 블록 단위 데이타 및 그림 출력 결과

그림 16은 문자 데이타를 출력하기 위해 스페이스 및 화이트 스페이스 처리 그리고 라인처리 등을 거친 결과로 실제 데이타 문자들과 스페이스 문자 등의 특수 문자 등을 관리하며 문자단위로 표현해 준 결과를 보이고 있다.

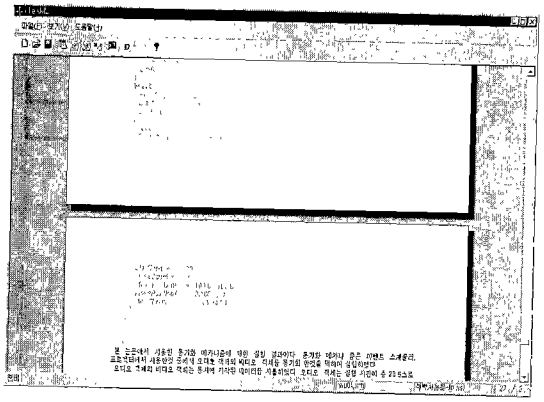


그림 16 문자 데이타 출력 결과

그림 17은 테이블 출력 결과를 보이고 있다. 테이블의 각 셀은 하나의 블록 단위 포맷팅 데이타를 포함하고 첫번째 행의 첫번째 열 즉, 첫번째 셀이 대표로 메모리 배열에 관리되며 나머지 셀들은 트리 형태의 구조를 이루며 포인팅되어 관리된다.

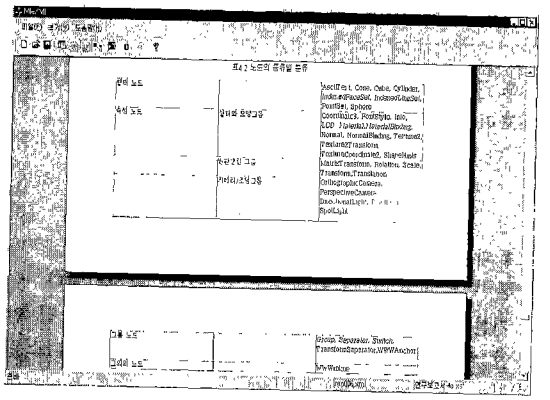


그림 17 테이블 출력 결과

4.2 고찰

본 논문은 XML 문서를 XSL의 변환 및 표현 정보를 이용하여 포맷팅 객체 트리를 생성하고 포맷팅 처리하는 XML 문서 표현 시스템의 설계 및 구현에 관한 것이다.

본 논문에서 개발한 문서 표현 시스템은 XML/XSL

문서의 검증 및 정보 추출을 위해 XSLT/XPath 처리기가 내장된 MSXML 파서를 사용하여 포맷팅 객체 트리를 생성하였으며, 포맷팅 객체 트리를 표현하기 위한 포맷터를 구현하였다.

포맷팅 객체 트리는 표준화된 DOM 인터페이스 사용이 파서에 따라 약간은 상이하므로 이를 위해 C++ 의 네임스페이스(Namespace)와 컴파일러의 선행 처리기를 사용하여 구현하였다.

포맷팅 객체 트리를 포맷팅 처리하는 포맷터는 다른 시스템에 적용할 수 있도록 동적 연결 라이브러리 형태로 구성하였고, 포맷팅 모듈 처리과정을 최소화하여 빠른 속도로 처리 가능하게 구현하였다.

본 시스템의 장점은 첫째, 웹 표준화 기구인 W3C에서 제안하는 XSL 의 포맷팅 처리에 대한 구성을 기반으로 구현 함으로써 표준화에 입각한 처리 시스템으로써 변화에 능동적으로 대처 가능하다는 점이고, 둘째 시스템을 모듈화하여 부분적인 수정 및 대체가 가능하다는 점이다.

현재의 웹 브라우저들은 HTML 언어를 기반으로 하여 XML을 추가 처리하는 시스템으로 발전하고 있고, 기존의 HTML/CSS 포맷팅 처리를 이용하므로 HTML/CSS 포맷팅의 한계를 넘어서는 처리는 할 수 없다. 즉, XSL을 완전히 수용할 수 없기 때문에 본 시스템의 구현은 의미가 있는 것이다.

미흡한 점으로는 XLink(XML Linking Language)와 XPointer(XML Pointer Language) 등의 XML 부가기술 처리에 대한 설계가 미흡하고 링크와 같은 기존에 사용되는 기술의 처리는 보완해야 한다. 또한 본 시스템은 XSL 명세서 1.0에 따라 구현하였으나 아직 개발 중이므로 지속적인 관심과 연구가 요구된다.

본 논문에 관련된 연구는 기존 브라우저 시스템의 확장 연구, 그리고 순수 XML 문서와 XSL-fo를 이용한 연구로 구분되어 진다. 기존 브라우저 시스템에서 이루어지는 연구는 XML 문서를 XSLT/XPath를 이용하여 변환하고 HTML/CSS 포맷팅 처리 엔진을 그대로 사용한다. MS사의 Internet Explorer와 Netscape사의 Gecko 엔진을 이용한 브라우저, 그리고 Opera 소프트사의 Opera가 대표적인 XML / HTML 브라우저이다.

반면에 XML 문서를 XSLT/XPath/XSL-fo를 사용해 포맷팅 처리하는 연구는 InDelv사의 InDelv XML/XSL browser와 본 시스템을 들 수 있다[18].

InDelv는 본 연구와 같은 의미를 갖지만 한글과 같은 2바이트 코드 처리에 문제가 있기 때문에 한글코드 처리를 지원하는 본 연구의 의미는 크다고 할 수 있다. 표 1은 타 XML 포맷팅 시스템간 비교이다.

5. 결 론

최근 정보화 산업 사회의 가속화와 더불어 전자문서 처리 및 교환이 요구됨에 따라 효율적인 전자문서의 관리와 처리가 요구되고 있다. 또한 인터넷의 발전으로 이러한 전자문서의 교환은 급속도로 발전해 왔고 SGML의

표 1 타 XML 포맷팅 시스템간 비교

시스템	특징 및 장점	단점
MS사의 Internet Explorer	XSLT / XPath를 지원 XML 문서를 스타일링 하기 위한 CSS 제공	CSS와 XLink를 HTML에 의존 XSL-fo 미 지원
Netscape사의 Mozilla	XML를 제공하는 하부 구조가 간단 하고 HTML과 동등한 CSS 표현 가능	XSL 미 지원 XLink의 단순 링크만 지원
Opera Software사의 Browser	링크와 HTML 처리의 세부적인 사항 몇 가지를 제외하고 Mozilla와 핵심 요소가 동일	고정적인 XML 브라우징 Mozilla에 비해 CSS 지원부족 하고 XSL 미 지원
InDelv사의 XML/XSL browser	XSL 구성 요소 지원 JAVA로 구현 공개 소스	XSL의 네임스페이스 등 표준을 완전히 따르지 않음으로 스타일시트의 호환성결여
본 시스템	XSL 구성 요소 지원 (XSLT / XPath / XSL-fo) 및 표준화 지원 C++로 구현하여 빠른 렌더링 속도 단일 브라우저 와 DLL 구성 타 시스템에서 활용가능한 렌더링엔진 입력 시 문서 검증 가능/다양한 입력 형식	XLink 미 지원으로 인한 링크처리 미흡

장점을 살린 XML은 이러한 전자문서 교환의 표준으로 자리 잡았으며, XML은 SGML과 마찬가지로 논리적 구조만을 가지므로 이를 포맷팅 처리하여 출력장치나 디스플레이 장치에 표현하기 위해 W3C에서는 문서 변환 및 포맷팅 정보 기술에 관한 표준인 XSL을 제안하였다.

본 논문에서는 XML 문서와 XSL 스타일시트의 입력으로 생성된 결과 트리를 포맷팅 객체 트리로 변환하고, 포맷팅 객체 트리를 포맷팅 처리하여 표현하는 시스템을 설계 구현하였다.

본 시스템의 장점으로는 웹 표준화 기구인 W3C에서 제안하는 XSL의 포맷팅 표준화에 입각한 처리 시스템이라는 점으로, 결과적으로 표준화의 변화에 빠른 대처가 가능하다. 또한 시스템의 대부분을 모듈화하여 부분적인 수정 및 대체가 가능하다는 장점이 있다.

본 연구 결과는 XML 관련 시스템 개발에 많은 영향을 줄 수 있으며, XML로 구축된 데이타를 검색하여 사용자에게 제공 할 수 있는 기반이 되고, 전자문서 환경이 요구되는 모든 산업 분야와 연구 분야에서 XML 문서를 표현하기 위해 본 시스템이 유용하게 사용되리라 사료된다. 또한 XML을 기반으로 하는 무선 인터넷, 전자상거래, 전자도서관 등에서 XML 문서의 표현을 위한 시스템 모듈로 널리 사용될 수 있어 XML 문서처리 시스템에서 본 시스템 결과의 파급효과가 매우 크리라 본다.

향후, XSL이 더 개발되고 그에 맞는 기능을 수행하기 위해서는 현재 XSL 처리기중 포맷팅 객체 트리 생성 모듈의 수정이 요구되고, XSL 스타일시트에 대한 연구도 병행되어야 할 것이다. 또한 본 시스템에는 인터넷의 요구에 발맞추어 링크 기능의 추가와 다른 스타일시트에 대한 처리가 가능한 기능 추가가 필요하다. 마지막으로 스타일시트 편집기능을 두어 직접 스타일시트의 원하는 포맷팅 정보를 수정할 수 있고, 원하는 형태로의 변환을 직접 볼 수 있도록 해야 할 것이다.

참 고 문 헌

[1] 정회경, "WWW 문서 작성을 위한 차세대 언어 XML 가이드", 그린,1998

[2] W3C, Extensible Markup Language(XML) Version 1.0(Second Edition), http://www.w3.org /TR/REC-xml, Oct. 6, 2000

[3] Frank Boumphery, "Professional Style Sheets for HTML and XML," June 1998, WROX Press

[4] 정회경, "문서 스타일 의미 지시 언어 표준개발에 관한 연구 보고서", 1997, 배재대학교

[5] W3C, HTML 4.01 Specification, http://www.w3.org/TR/html, Dec. 24, 1999

[6] W3C, Cascading Style Sheets, level 1, http://www.w3.org/TR/REC-CSS1.html, Jan. 11, 1999

[7] W3C, Cascading Style Sheets, level 2, http://www. w3.org/TR/REC-CSS2/, May 12, 1998

[8] W3C, A Proposal for XSL, http://www.w3.org/TR/NOTE-XSL.html, Aug. 27, 1997

[9] W3C, XSL Transformations (XSLT) Version 1.0, http://www.w3.org/TR/xslt, Nov. 16, 1999

[10] W3C, XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath, Nov. 16, 1999

[11] W3C, Extensible Stylesheet Language (XSL) Version 1.0, http://www.w3.org/TR/xsl/, Oct. 18, 2000

[12] Jon Bosak, DSSSL Online Application Profile, http://www.ibiblio.org/pub/sun-info/standards/dsssl/dssslo/do960816.htm

[13] Frank Boumphrey, XSL Tutorials, http://www.hypermedic.com/style/xsl/

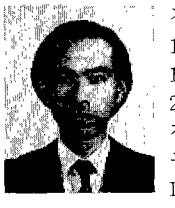
[14] Nic Miloslav, XSLT Tutorials, http://www.zvon.org/HTMLonly/XSLTutorial/Books/Book1/bookInOne.html

[15] Mulberry Technologies, Quick References, http://www.mulberrytech.com/quickref/index.html

[16] CRANE SOFTWRIGHTS, Practical Transformation Using XSLT and XPath, http://www.cranesoftwrights.com/training/index.htm

[17] MSDN Online (XML), http://msdn.microsoft.com/xml

[18] InDelv, XML/XSL browser,



김 진 수

1975년 숭실대학교 전자계산학과(학사). 1985년 홍익대학교 전자계산학과(석사). 2000년 충북대학교 전자계산학과(박사과정수료). 1988년 현재 배재대학교 컴퓨터공학과 교수. 관심분야는 Algorithm, Distributed AI, Spatial Database

강 치 원

1993년 2월 광운대학교 전자계산기공학과 졸업(공학사). 1995년 2월 광운대학교 전자계산기공학과 졸업(공학석사). 2000년 ~ 현재 배재대학교 컴퓨터공학과(박사과정). 1995년 ~ 현재 한전KDN(주) 재직. 관심분야는 XML, XML/EDI, SGML, 멀티미디어 문서처리

류 근 호

1976년 숭실대학교 전산학과 졸업(이학사). 1980년 연세대학교 산업 대학원 전산전공(공학석사). 1988년 연세대학교 대학원 전산전공(공학박사). 1976년 ~ 1986년 육군 군수 지원사 전산실(ROTC 장교), 한국전자통신연구소(연구원), 한국방송대학교 전산학과(조교수)근무. 1989년 ~ 1991년 Univ. of Arizona. Research Staff(TempIS 연구원, Temporal DB). 1986년 ~ 현재 충북대학교 컴퓨터과학과 교수. 관심분야는 시간 데이타베이스, 시공간 데이타베이스, 지식기반 정보검색, Temporal GIS, 객체 및 지식베이스 시스템

정 회 경

1985년 광운대학교 컴퓨터공학과(학사). 1987년 광운대학교 컴퓨터공학과(석사). 1993년 광운대학교 컴퓨터공학과(박사). 1994년 ~ 현재 배재대학교 컴퓨터공학과 교수. 관심분야는 하이퍼미디어/멀티미디어 문서정보처리, SGML/XML, HyTime, IETM, DSSSL/XSL, XML/EDI