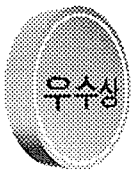
GOD

(Generator Of DTD) : CASE-Tool



정보처리전문가협회장상

1. S/W명 : GOD(Generator Of DTD)

2. 제작자 : 동아대학교 컴퓨터공학과

3. 요 약

HTML의 빠른 파급에 이어 그 한계에 따른 수 많은 문제점들이 나오고 있으며, 해결책으로 이미 S/W 선진국가에서는 XML을 그 대안으로 생각하고 있다. 아마도 가까운 미래에 HTML은 사라지게 되고, XML이 기존의 HTML을 포용하며 web을 장악 하게 될것이다.

하지만 수 많은 web 사용자들은 XML에 익숙하지 않고, 단지 종이와 연필 혹은 키보드로 바로 입력해 자신의(특히 대규모의 단체에서) 문서 구조를 정의하는 일은 결코 쉬운 일이 아닐 것이다.

구현한 작품은 그런 XML 사용자들이 쉽고 안전하게 자신의 문서구조를 생성할 수 있도록 해주는 TOOL로써 요소(ELEMENT), 개체(ENTITY), NOTATION, 속성(ATTRIBUTE), 요소들 간의 관계(RELATION), 요소와 속성간의 관계(LINE)... 등을 단순한 use-case들로 나타냄으로써 문서구조를 보기 편하고 쉽게 수정하고 인식할 수 있도록 해주며, 그런 그림들에서 작성자가 필요한 문서구조의 DTD를 얻을 수 있도록 해준다.

또한 기존 xml문서의 DTD를 편집하거나 수정해야하는 경우도 그 문서를 읽어옴으로써 그 문서에 적절한 use-case들을 그래프로 생성해고, 편집후 수정된 DTD를 생성해 준다.

4. 서 론

UML을 기반으로하여 XML DTD를 생성하기 위한 다이어그램을 생성하는 규칙과 이 툴에 추가된 여러 가지 option기능들, 객체를 저장하고 사용하는데 있어서의 방법과 시각적 객체 모델링 도구를 구현하는 방법에 대해 알아본다.

5. 본 론

1. 1 그래프 모델

기존의 UML은 기본적으로 graph모델을 그 기반으로 하지만 우리가 만든 이 tool에서는 graph 모델을 사용하게 되면 문서구조를 보여줄 때 무한 loop에 빠질 위험이 있어 tree구조를 사용하기로 한다. XML DTD 에서의 각 요소(ELEMENT),

그룹(GROUP), 개체(ENTITY), 노테이션(NOTATION), 속성(ATTRIBUTE)와 그들간의 여러 관계(RELATION과 LINE)등으로 새로운 TREE구조를 형성한다. 따라서 이들을 관리하기 위해 TREE구조를 사용하게 되고, <figure-1>의 왼쪽 편에 몇 개의 요소와 속성, 그룹등으로 구성된 책에 대한 문서구조를 나타내는 tree-viewer가 나타나 있다.

1. 2 Container

TREE 구조의 원소들을 효과적으로 관리할 수 있는 적적할 container가 필요할 것이고, 나는 LIST를 사용하게 되었다. 조건문에 의해 객체를 식별하고 필요시마다 조건 검색후 해당되는 리스트에 새로운 객체를 추가하는 식으로 그 객체를 저장하게 된다. 그리고 객체를 저장된 매체로부터 가져올 경우 적당한 검색후 필요시 BFS를 방식으로 하여 트리를 순회하여 필요한 연산을 수행하게 된다.

리스트에 추가할 경우의 pseudo code는 아래와 같다.

```
for all object in Tree do
begin
  if Object's id = DTD-ELEMENT then ELEMENTLIST.Add(Object)
  else if Object' id = DTD-ENTITY then ENTITYLIST.Add(Object)

  else if ...
   :
end;
```

매체에서 필요한 객체를 가져올 겨우의 pseudo code는 아래와 같다.

```
if (Object's id = DTD-ELEMENT) then begin
  for i:=0 to ELEMENTLIST.count-1 do
    if (Object = ELEMENTLIST[i]) then take_that_ELEMENT;
...
else if (Object's id = DTD-GROUP) then begin
...
```

1. 3 OPTION기능-ZOOMING 기능, SEARCH기능, TREEVIEW기능...

문서구조가 커지게 되면 그 문서 구조를 표현하기 위한 Diagram은 너무 커지게 되고 화면을 넘치게 될 것이다. 그런 경우를 위해 zoom-in과 zoom-out기능을 추가하였으며, Diagram내에 있는 수 많은 원소들 중 필요한 원소를 찾아내기가 쉽지 않을 것이다. 그런 경우를 위해 조건 검색을 취해 그 조건에 해당하는 원소를 식별해 주며 Diagram내에서 빨간색으로 표시하여 주는 option을 추가하였다. 또한 비록 다이어그램이 그래프 구조를 가진다 하더라도 전체적으로 분석하기는 쉽지 않을 것이다. 그런 경우를 위한 option으로 treeview기능을 추가함으로써 Diagram에

나타나는 원소들을 우리가 흔히 사용하는 윈도우 탐색기 형식으로 display해 주게 된다.

zooming 기능의 구현시는 system programming쪽이라 사실 Delphi보다는 C++로 구현하는 것이 훨씬 수월했겠지만 이미 Object Pascal로 구현하던 중이었고 다시 시작하기에는 좀 늦은 감이 있어 Delphi로 구현하게 되었다.

Searching 기능의 경우는 search form에서 옵션을 선택하고 프로그램내에 생성된 인스턴스들의 Tree구조를 BFS 방식으로 순회하게 함으로써 찾고자 하는 객체를 식별해 내게 된다.

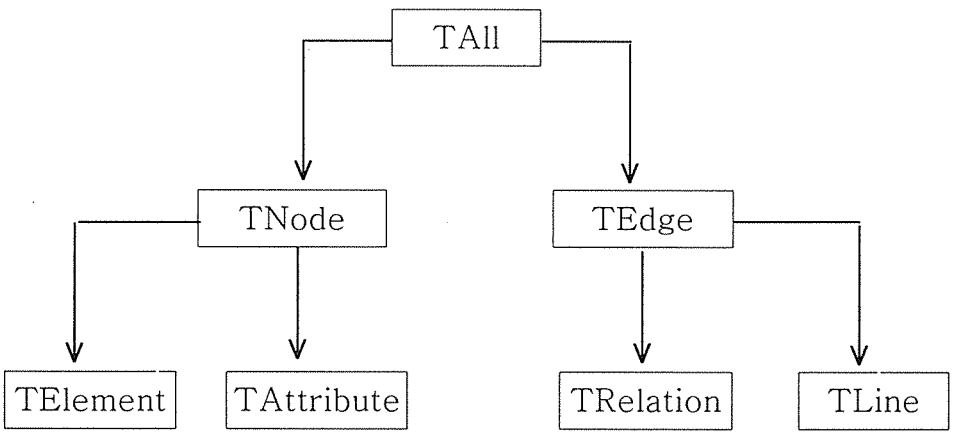
Treeview 기능의 경우는 각각의 Element나 Group이 생성되거나 갱신될 경우에 Treeview Form의 각 노드들을 추가하고 갱신하게 된다.

속성값이 ENTITY, ENTITIES, NOTATION, NOTATION(enumerated)등과 같은 경우는 그것들을 속성값으로 사용하기 전에 미리 정의되어 있어야 하므로 compiler개념을 도입하여 입력으로 string을 받아 그것이 필요한 조건에 맞는 지를 검사한 후 사용으로 적절하다면 그 입력값을 의미있는 토큰으로 분석한 후 적절한 코드를 생성해 주게 되며 적절하지 않다면 error처리를 하게된다.

1. 4 시각적 객체 모델링 도구의 구현

본 장에서는 시각적 객체 모델링 도구의 DTD 구조 다이어그램 편집을 위한 원소들 간의 관계 및 구조를 보여주고 UML 개념으로부터 XML DTD로의 변환 규칙 및 DTD 생성에 관한 내용을 소개한다.

DTD 다이어그램은 UML의 정적구조 다이어그램(Static structure diagram) 과 거의 유사하며 주로 요소(ELEMENT)와 그들 간의 관계로써 XML DTD의 구조를 표현한다.



위의 그림에서 보면 최상위 클래스인 TAll은 그 하부 객체들의 식별을 위한 정보를 가지고 있으며 각종 Element, Entity, Group, Attribute와 같은 원소들은 Node로부터 상

속되고, Relation(Element와 Group 혹은 Element들 간의 관계등)과 Line(특정 Element와 Attribute 들과의 관계등)은 Edge로부터 상속되게 된다.

2. 2 UML과 XML DTD간의 Mapping 규칙

UML과 XML DTD간의 Mapping 규칙과 어떻게 XML DTD의 Diagram으로 Mapping 시킬 수 있을지를 소개한다.

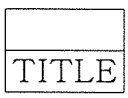
XML DTD의 생성은 아래의 Mapping 규칙을 잘 활용하면 그리 어렵지 않게 구현할 수는 있지만 결과가 얼마나 customize될 수 있는가는 쉽지 않은 문제이다.

- Mapping 규칙(8가지) -

1) ELEMENT

요소는 부모요소와 자식요소가 있는데, 기존의 HTML 개발자들은 리스트와 테이블을 제외하고는 그것이 쓰이는 문맥에 대해 별 관심없이 요소를 사용하는 데 익숙해 있다. 하지만 효율적으로 돌아가는 DTD를 구축하기 위해서는 반드시 그것의 문맥을 이해해야 한다. XML에서 문맥을 제공하는 것은 부모 요소이고, 자식 요소는 자신에게 포함된 요소에게 문맥을 제공한다. 요소를 생성하는 일은 매우 간단하며 아래 문법에 따른다.

<!ELEMENT element-name content>



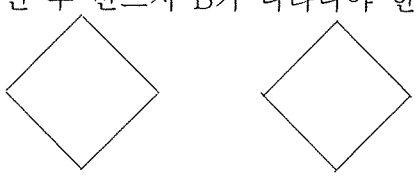
속성 추가시(mousedblclick-event) : <ELEMENT FORM>이 생성

<!ELEMENT TITLE (...)>의 문서가 생성되게 될 것이다.

2) GROUP

부모요소가 자식요소들의 그룹으로 관계를 가질 경우이며, AND 관계나 OR 관계 중의 하나이다.

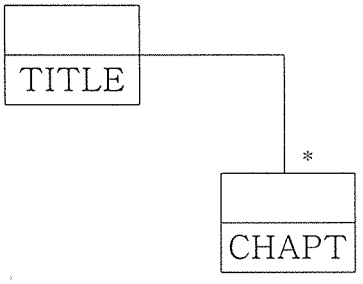
(A | B)의 경우 A나 B중의 하나가 반드시 나타나야 하며, (A, B)의 경우는 A가 나타난 후 반드시 B가 나타나야 한다.



OR AND

3)RELATION

대부분의 요소 선언에는 요소 리스트가 포함되고, 요소에게 필요한 규칙을 지정할 것이다. 요소들이 나타날 위치와 나타날 횟수를 그 규칙에서 명시한다.(리스트에 있는 모든 요소는 항상 자체의 선언에서 따로 정의되어야 한다.) RELATION은 그런 요소들 사이의 관계를 명시해 주며, 아래 그 생성규칙에 대한 몇 가지의 기호가 나와 있다.



속성 추가시(relation을 double-click) : <RELATION FORM>이 생성
<!ELEMENT TITLE (CHAPT)*>의 문서가 생성되게 될 것이다.
아래 표는 부모 요소와 자식 요소간의 존재할 수 있는 관계를 보여준다.

기호	기호타입	설명	예제	예제 설명
|	세로 막대	OR 조건	이것|저것	이것 또는 저것이 반드시 나타남
,	쉼표	명시된 순서대로 나타남	이것, 저것	이것이 나타난 뒤에 저것이 나타남
?	물음표	옵션이지만 한번은 나타날 수 있음	이것?	이것이 0번이나 1번 나타남
*	별표	안 나타날 수도 있고, 여러번 나타날 수도 있음	이것*	이것이 0번이상 나타남
+	덧셈표	최소 1번 이상 나타남	이것+	이것이 1번 이상 나타남
1	숫자 1	1번만 나타남	이것	이것이 1번만 나타남

기호	기호타입	설명	예제	예제 설명
()	괄호	그룹 요소	(이것|저것), 어느것	이것이나 저것중 하나가 나타난 후, 어느것이 나타남
기호 없음	default	반드시 한번 나타남. 툴내에서는 1이나 공백	이것	이것이 반드시 나타남

각각의 Relation은 +, *, ?, 1의 4가지 중 하나를 가질 수 있을 것이며 default(아무것도 입력되지 않을 경우)로는 공백(한번 나옴)이 설정되어 있다.

4) ATTRIBUTE

HTML에서는 속성이 대부분의 기능을 제공하지만, XML에서는 속성을 약간 덜 사용하게 된다. 속성은 사람보다는 컴퓨터가 관심이 많은 정보를 저장하는 데 매우 유용하다. HTML에서도 요소에 브라우저를 위한 핵심 정보가 저장되는 것이지, 요소의 내용물에 관한 정보가 저장된 게 아니다. 그러나 XML에서도 속성이 요소이상의 융통성을 제공하고, 기반 구조를 굳건하게 함으로써 핵심적인 부분을 차지한다. 속성 선언은 요소 선언과 비슷한 문법을 사용하지만 그것이 허용하는 내용에 대해 좀 더 정확하게 정의하는 경향이 있다.

아래의 문법으로 속성을 정의한다.

```
<!ATTLIST element-name
    AttributeName Type Default
   (AttributeName Type Default...)>
```

속성은 문서 확장시 DTD 전체를 수정할 필요없이 쉽게 DTD를 확장할 수 있게 한다.

속성 Type으로 사용될 수 있는 값은 아래의 표와 같으며, 아래 표에 나타나 있는 각 속성 type의 조건에 만족되지 않을 경우는 적당한 error message를 내 줌으로써 컴파일러까지 가지 않더라도 코드의 잘못된 부분을 수정할 수 있게 해줌으로써 훨씬 안전하고 효율적인 코딩작업과 코드생성이 가능하게 된다.

타 입	설 명
CDATA	속성이 문자 데이터만 포함할 수 있다(마크업은 이 속성값 안에서 해석될 수 없다).
ID	속성값이 유일하여 요소를 식별할 수 있다. 한 문서에서 ID 타입의 두 속성이 같은 값을 가지면 파서는 오류를 내야한다.
IDREF	속성값이, 문서 안의 다른 곳에서 선언된 ID 값을 참조한다. 만일 그 값이 일치하지 않으면 파서는 오류를 내야한다.

타 입	설 명
ENTITY, ENTITIES	ENTITY 속성의 값이 DTD 안에서 선언된 이진 외부 개체의 이름에 대응해야 한다. ENTITIES는 이와 비슷한데, 공백으로 분리된 개체 이름들을 여러개 허용한다.
NMTOKEN, NMTOKENS	속성값이 반드시 CDATA와 같은 이름 토큰이어야 하고, 그 값에 사용되는 문자가 반드시 알파벳, 숫자, 마침표, 붙임표, 밑줄, 쌍점이어야 한다.
NOTATION	속성값이 반드시 그 DTD안에서 선언된 notation이름을 참조해야 한다.
enumerated	속성값이 반드시 나열된 값 중에 하나여야 한다. 값들은 괄호 안에 OR(|)기호로 분리되어 나와야 한다.
NOATATION (enumerated)	속성값이 반드시 나열된 NOTATION 이름중에 하나여야 한다.

다른 타입도 쓸 수 있지만, 대부분의 개발자들은 CDATA, ID, enumerated의 타입을 사용할 것이다.

NOTATION, ENTITY, ENTITIES, enumerated, NOTATION(enumerated) 와 같은 타입의 경우는 그 사용 전에 사용하고자 하는 NOTATION이나 ENTITY 들을 미리 정의해 주어야 하며 그렇지 않을 경우에는 XML 컴파일러까지 가지 않더라도 툴 내에서 사용전에 먼저 정의하라는 메시지를 보내주게 된다.

속성선언에서 마지막으로 필요한 부분은 기본값이다. default value는 아래의 4가지 중 하나여야 한다.

값	설 명
#REQUIRED	요소에서 속성이 실제 쓰일 때 반드시 값이 지정되어야 한다. 그렇지 않으면 파싱 오류가 된다.
#IMPLIED	속성에 값이 지정되어 있지 않으면 파서가 그 속성을 무시한다. XML Specification에는 다음과 같이 씌어져 있다. "xml 처리기는 응용에게 아무 값도 지정되어 있지 않음을 반드시 알려 줘야한다. 그 응용의 반응에 대해서는 아무런 제약도 없다."
#FIXED value	속성에 대한 값을 지정하는 요소의 용례에서 반드시 value를 지정해야 한다. 만일 요소 용례가 이 속성을 포함하지 않으면 그것의 값이 value로 간주될 것이다.
default value	속성을 위한 기본값을 제공한다. 그 속성이 요소 용례에서 명시적으로 선언되지 않으면, 그 속성값은 default value로 간주될 것이다.

아래의 Diagram은 A 요소가 id속성을 가지는 경우이다.

<!ATTLIST TITLE
id ID #REQUIRED>의 DTD를 생성해 준다.

5) ENTITY

XML에서는 2가지 개체를 제공하는데 일반 개체와 매개변수 개체이다. HTML 개발자들은, 예외적인 문자와 마크업을 위해 사용되는 문자들(악명높은 <, >, &)의 인코딩을 위해 미리 정의된 일반 개체를 사용하는 것에 익숙하다. 일반개체는 DTD에서 정의되지만, &개체이름; 형태의 개체 참조가 나오면, 그 값을 개체 참조 대신 사용하여 문서에 정보를 추가한다. 매개변수 개체는 외부 DTD에서만 정의할 수 있다. 이 개체는 일반 개체처럼 개발자가 입력하는 수고를 덜어줄 뿐 아니라, DTD 안에 다른 DTD나 다른 정보를 포함시킬 수 있는 엄청난 기능을 제공한다. 매개변수 개체를 활용하면, 개발자는 옛 DTD를 재사용할 수 있기 때문에, 다른 사람들이 이미 해 놓은 일들을 하지 않고도, 기존 DTD를 쉽게 확장할 수 있다. 개체는 Diagram 내에서는 나타나지 않으며, 툴의 오른쪽 상단에 있는 리스트 박스에 추가되게 되며 마우스 버튼으로 선택할 경우 그 리스트 박스 하단에 그 개체에 대한 설명이 나타나게 된다.

일반 개체의 경우는 <!ENTITY entity-name entity-definition>의 형식이며, 매개변수 개체의 경우는 개체이름 앞에 &를 추가함으로써 일반개체와 구분하여 준다.

매개변수 개체의 형식은 <!ENTITY & entity-name entity-definition>을 따른다.

6)NOTATION

Notation 선언은 문서에 외부(비 xml)소스로부터 오는 데이터가 필요하다는 선언이며, 파서 이외의 다른 응용에게 처리를 넘겨주는 데 필요하다. 때때로 notation 선언은 처리 명령과 함께 문서에 있는 비텍스트 정보를 다루는 수단을 제공하기도 한다. notation의 선언은 다음의 문법에 따른다.

<!NOTATION Notation-Name ExternalID>

notation도 개체와 마찬가지로 tool의 diagram내에서는 표현되지 않으며, notation 추가시 오른쪽 하단의 리스트 박스에 그 notation이 추가되게 되며 마우스 클릭시 그 notation에 대한 세부 설명을 그 리스트 박스 하단에 보여주게 된다.

7)LINE

Line은 특정 요소와 그 요소가 가지는 속성간의 관계를 보여주는 diagram의 원소이다.

단순하게 특정 element를 헤더로 가지고 특정 속성을 tail로 가지는 선분이다.

8) 통합

7)번까지의 Mapping 규칙을 바탕으로 간단한 XML DTD를 위한 Diagram을 그려보면 다음 페이지와 같으며, 이 툴의 실행결과 생성된 DTD는 Diagram 다음 페이지에 보여진다.

3. 실험결과 및 분석

본 작품은 그래프 모델, UML과 XML DTD간의 Mapping 규칙등을 그대로 적용하여 XML DTD를 자동 생성하며 어렵지 않게 복잡한 문서구조를 생성, 편집할 수 있게 해주는 'GOD(Generator Of DTD)'이라는 시각적 객체 모델링 도구이다. 개발은 Delphi Object Pascal 언어를 사용하여 구현하였으며, 여러 프로그래밍 언어들(특히 객체지향 언어들)에 대해서는 이미 세계적으로 유명한 수많은 CASE-TOOL(사용해 본 건 Cayenne Software의 Object-team과 Plastic밖에 없지만, 그외에도...)들이 나와 있지만 XML이 아직은 덜 알려져 있는 상황이고, 전문적인 문서구조를 필요로 하지 않을 경우에는 대부분의 경우 HTML을 사용하므로 다행스럽게도(?) XML DTD를 생성해 주는 CASE-TOOL은 아직 나오지는 않았지만 앞으로 XML의 사용은 불을 보듯 분명한 일이다.

생성된 DTD에 이상이 있는 경우는 Compiler까지 가지 않더라고, 이 툴로 DTD를 생성할 때 적절한 error를 처리하게 해 줌으로써 훨씬 안전하고 정확하며 빠르게 자신만의 혹은 그 집단만의 문서구조를 생성할 수 있게 하는 것이 이 TOOL의 목적이다.

6. 결 론

본 작품에서는 표준 객체 모델링 언어인 UML을 채택하여 XML DTD를 시각적으로 모델링할 수 있는 시각적 객체 모델링 도구를 개발하였다. 이 툴을 사용함으로써 사용자들은 좀 더 편리하고 효율적으로, 그리고 안전하게 자신의 문서구조를 생성할 수 있을 것이며 대규모의 문서구조를 필요로하는 경우도 이런 툴을 사용하여 각 부분의 DTD를 생성하고 외부 개체를 이용하여 하나로 합침으로써 거대한 문서구조를 생성하는 일도 훨씬 수월하게 될 것이다.

기본적인 코드는 다 생성되지만 차후에 시간과 능력이 된다면 기존의 html문서를 읽어들여 그 문서에 맞는 문서구조(DTD)를 생성해 줌으로써 그 문서를 XML 문서로 변환하는 기능과 기존의 XML문서를 읽어 use-case로 변환해 줄 수 있는 역공학 기능을 포함해 이 TOOL내에서 생성된 XML DTD를 사용하여 바로 xml문서를 만들어 그 문서를 compile할 수 있게 해 주는 간단한 xml compiler도 추가할 예정이다.

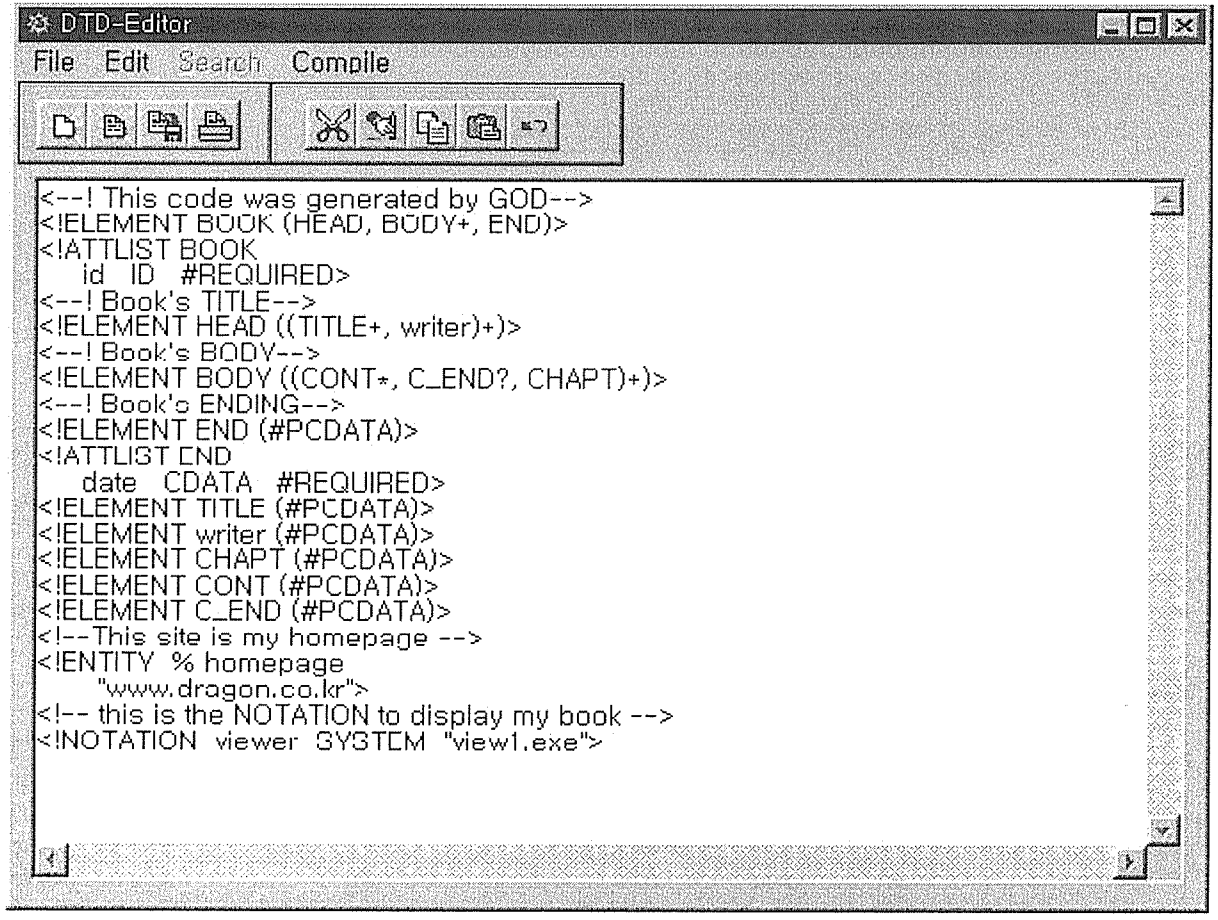
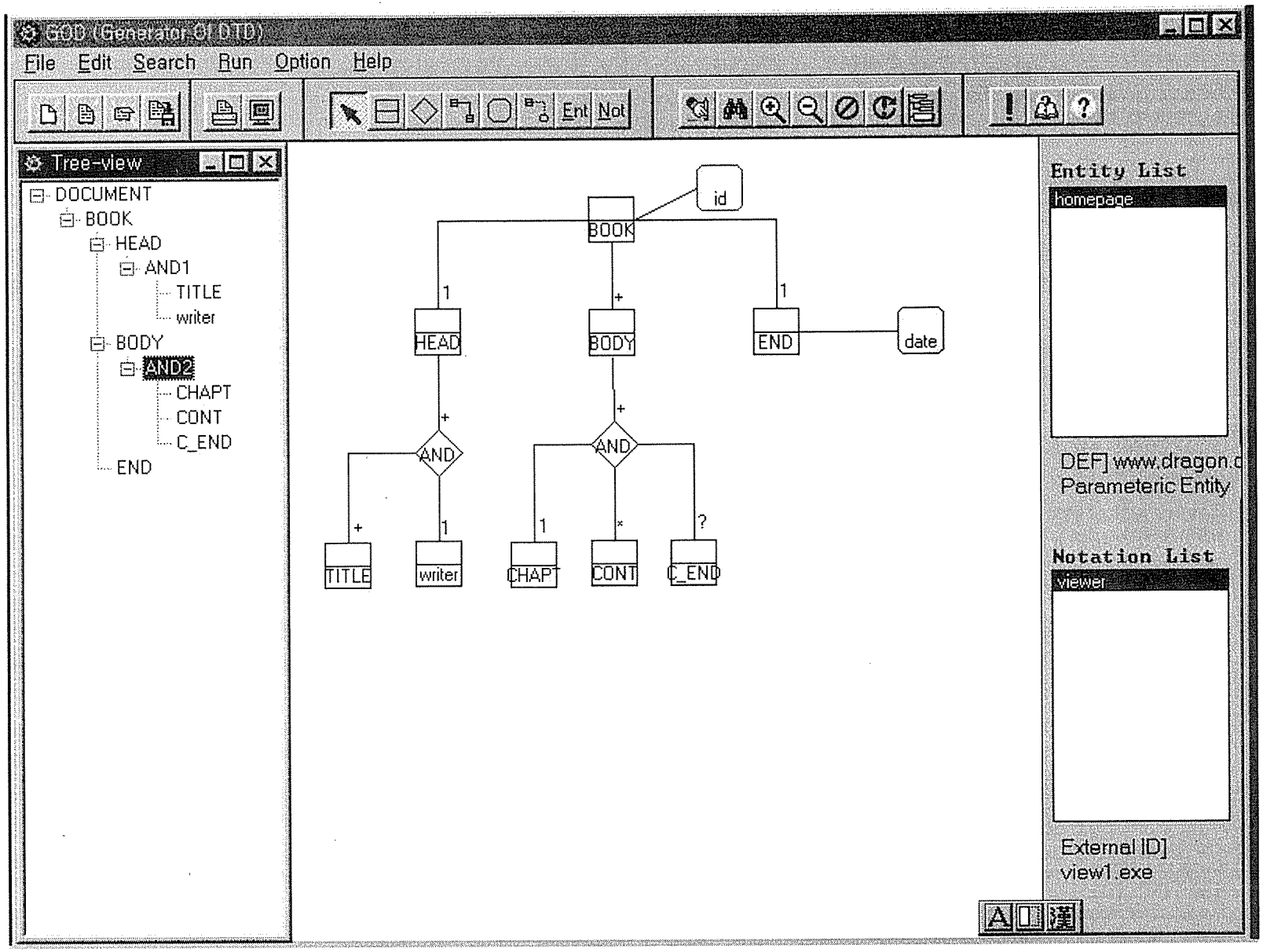


그림 1

<GOD에 의해 생성된 DTD 문서>


GOD (Generator Of DTD)
File Edit Search Run Option Help
Ent Not
Tree-view
DOCUMENT
BOOK
HEAD
AND1
TITLE
writer
BODY
AND2
CHAPT
CONT
C_END
END
id
BOOK
1
+
1
HEAD
BODY
END
date
+
+
AND
AND
+
1
1
*
?
TITLE
writer
CHAPT
CONT
C_END
Entity List
homepage
DEF] www.dragon.c
Parameteric Entity
Notation List
viewer
External ID]
view1.exe