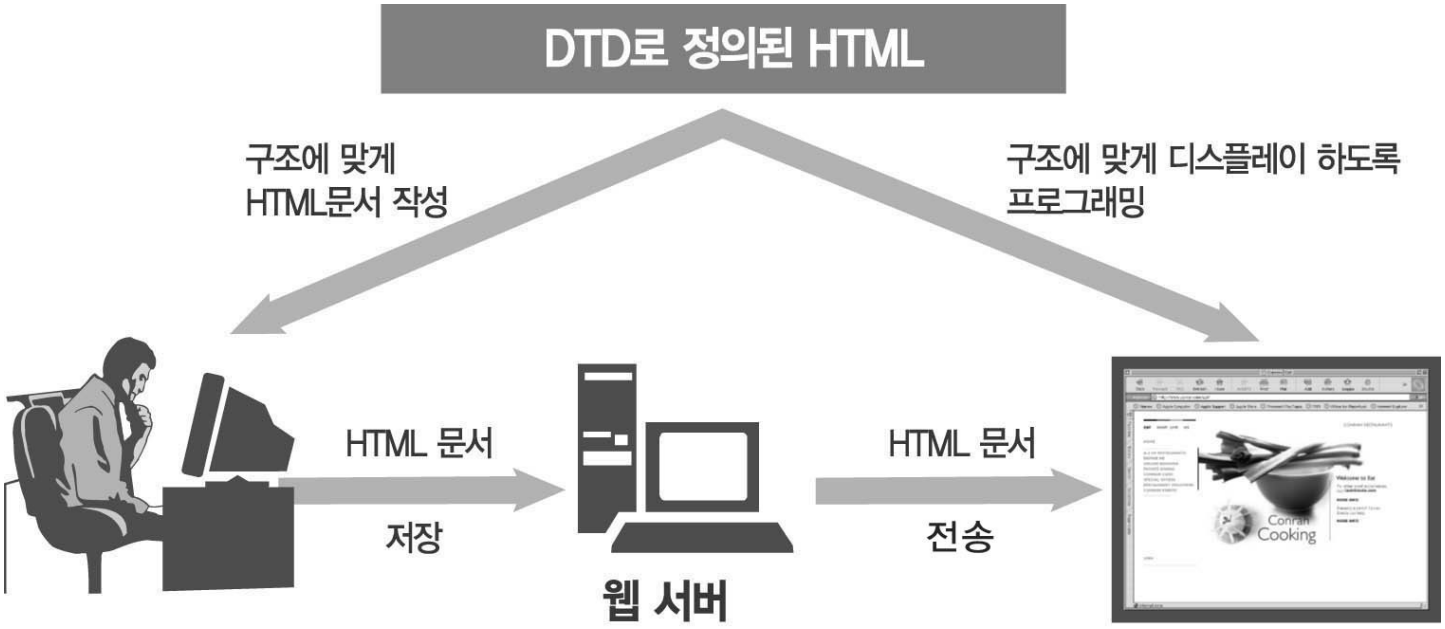
제 4 장 DTD를 이용한 마크업 언어 개발

1. 마크업 언어 개발의 의미
2. 유효한 문서의 개념
3. DTD의 물리적 구조
4. 내부 DTD 서브셋
5. 외부 DTD 서브셋
6. 문서 유형 선언하기
7. DTD 유효성 검사
8. 텍스트 선언
9. 주석
10. 엘리먼트 선언
11. 속성 선언
12. 엔티티 선언
13. 노테이션 선언
14. 컨디셔널 섹션
15. BML(Book Mark Up Language) 개발 예제

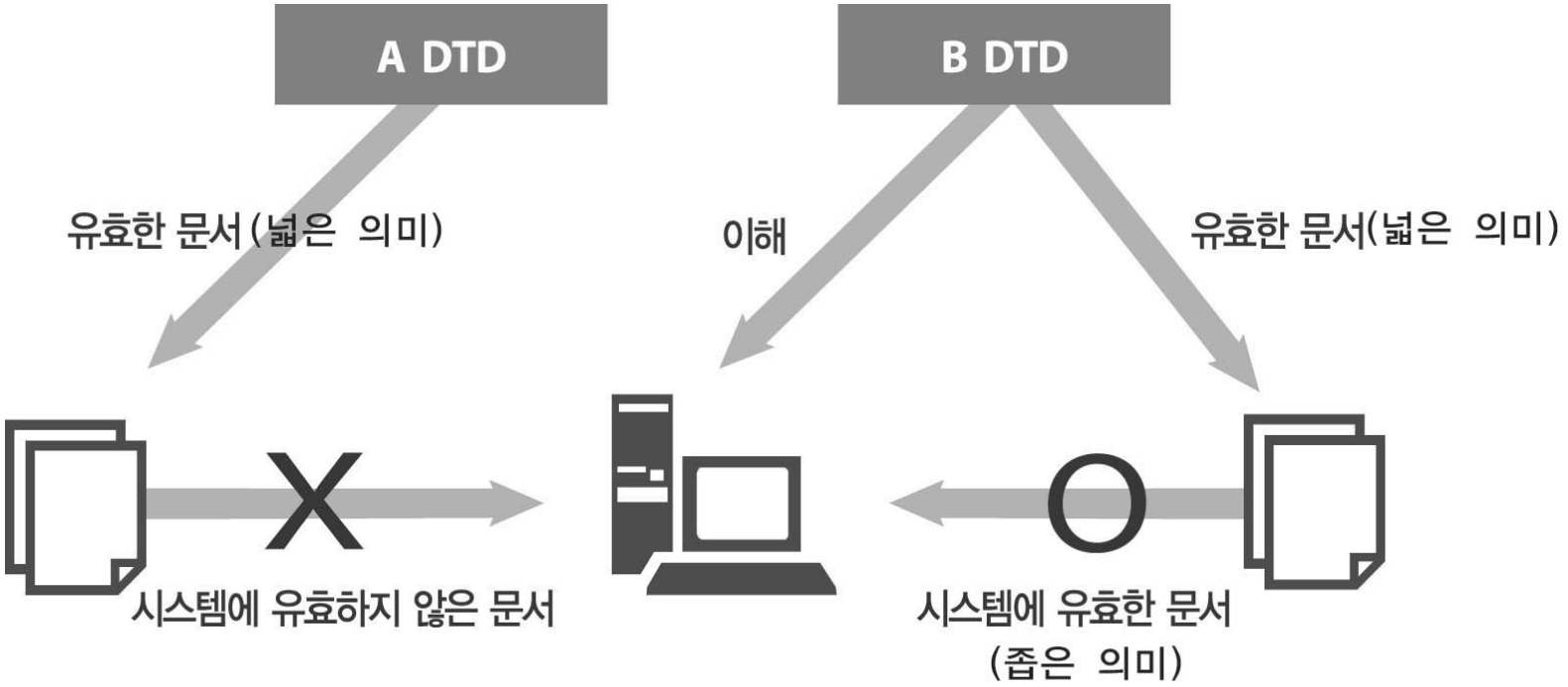
1. 마크업 언어 개발의 의미

- XML은 메타언어이다. 즉, DTD나 Schema를 이용해서 XML 문서의 구조를 정의함으로서 새로운 마크업 언어를 개발 할 수 있다.
- 마크업 언어 개발의 의미 : 프로토콜(Protocol)의 통일을 뜻한다.
- 새로운 마크업 언어를 개발하면 관련 응용프로그램도 DTD 구조를 이해할 수 있도록 개발 되어야 한다.



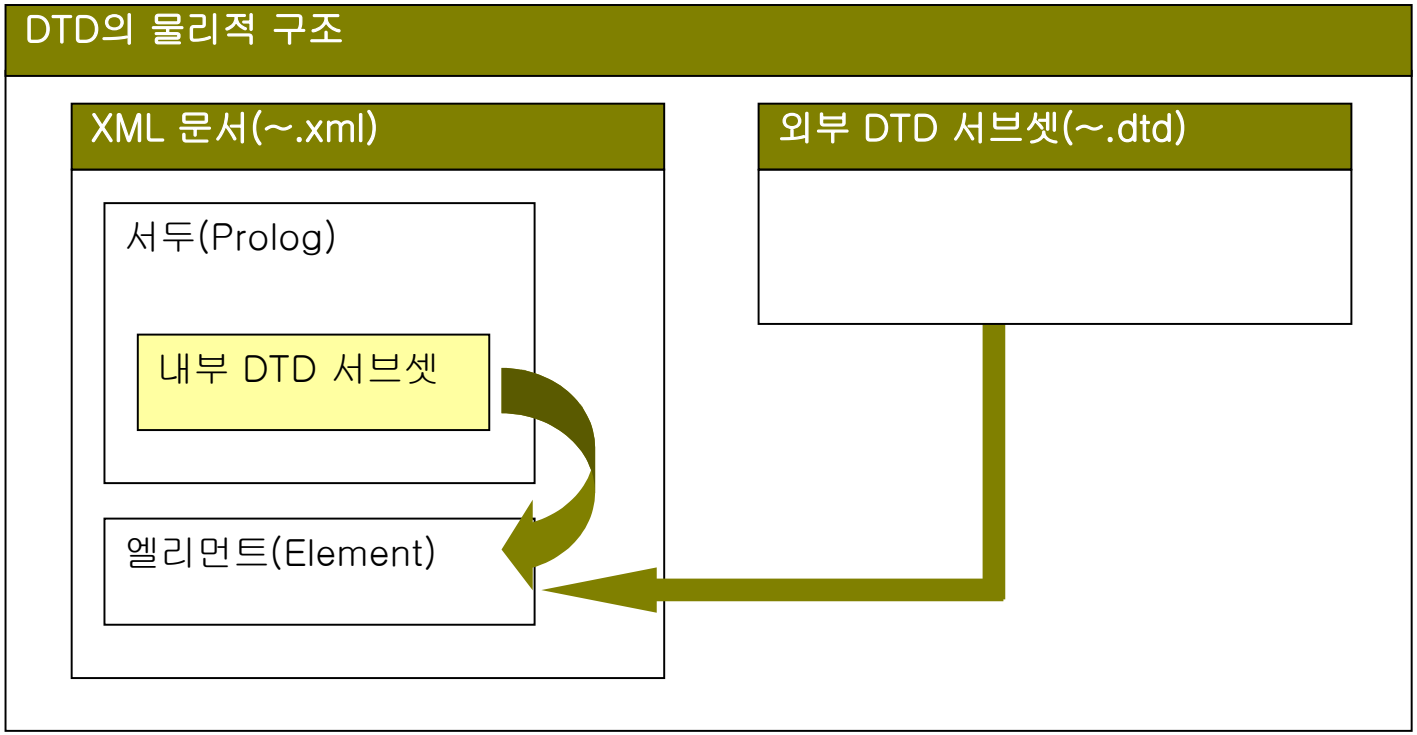
2. 유효한 문서의 개념

- 유효한(valid) 문서 : "특정 DTD 구조 대로 작성된 문서를 해당 DTD에 대해 유효한 문서라 한다."
  Well-formed + 특정 마크업 언어(DTD or Schema)



3. DTD(Data Type Definition)의 물리적 구조

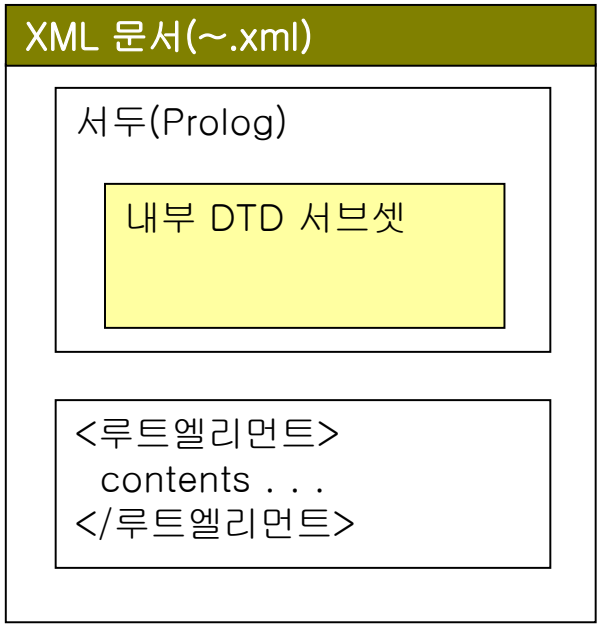
- XML 권고안에서는 DTD를 **내부 DTD** 서브셋(Subset)과 **외부 DTD** 서브셋으로 구분한다.
- 일반적으로 DTD 문서라고 하면 외부 DTD 서브셋이 정의되어 있는 문서를 말한다.



4. 내부 DTD 서브셋

4-1 내부 DTD 서브셋 작성 위치

- 마크업 언어에 대한 정의가 XML 문서 내부에서 작성되는 것을 내부 DTD 서브셋이라 한다.
- 내부 DTD 서브셋은 주로 외부 DTD 서브셋의 일부 내용을 재정의해서 XML 문서에 적용할 목적으로 이용된다.



XML 문서 + 내부 DTD 서브셋

```
<?xml version="1.0" encoding="euc-kr"?>
<!-- 내부 DTD 서브셋 정의 -->
<!DOCTYPE booklist [
  <!ELEMENT booklist (book+)>
  <!ELEMENT book (#PCDATA)>
]>
<!-- 루트 엘리먼트 -->
<booklist>
  <book>XML And HTML</book>
  <book>XML And VB</book>
  <book>XML And Java</book>
</booklist>
```

5. 외부 DTD 서브셋

5-1. 외부 DTD 서브셋 작성 위치

- 외부 DTD 서브셋은 마크업 언어에 대한 정의가 별도의 파일로 작성된 것을 말한다.
- 외부 DTD 서브셋은 “dtd”라는 확장자를 사용한다.(확장자 자체는 의미 없음)
- 새로운 마크업 언어를 개발한다는 것은 새로운 외부 DTD 서브셋을 작성하는 것을 의미한다.

외부 DTD 서브셋 : bml.dtd

```
<?xml encoding="euc-kr"?>
<!ELEMENT booklist (book+)>
<!ELEMENT book (#PCDATA)>
```

XML 문서

```
<?xml version="1.0" encoding="euc-kr"?>
<!-- 문서 유형 선언 -->
<!DOCTYPE booklist SYSTEM “bml.dtd”>
<!-- 루트 엘리먼트 -->
<booklist>
  <book>XML And HTML</book>
  <book>XML And VB</book>
  <book>XML And Java</book>
</booklist>
```

6. 문서 유형 선언하기

6-1 문서 유형 선언

- XML 문서 작성자가 XML 문서를 해석하는 측에 전달하는 마크업 언어에 대한 정보.

6-2 외부 DTD 서브셋에 대한 문서 유형 선언

문법	<?xml version="1.0" encoding="euc-kr"?> <!-- 문서 유형 선언 --> <! DOCTYPE 루트엘리먼트 SYSTEM "SYSTEM 식별자" >

- SYSTEM 키워드는 주로 국제적으로 공인되지 않은 단체에서 만든 외부 DTD 서브셋 문서를 지정할 경우 사용된다.
- SYSTEM 식별자에는 DTD 문서를 실제 다운로드 할 수 있는 URL을 기술한다.

< 특정 웹서버상에 DTD 문서가 있을 경우 >

XML 문서	<?xml version="1.0" encoding="euc-kr"?> <!-- 문서 유형 선언 --> <!DOCTYPE 루트엘리먼트 SYSTEM "http://웹서버주소/경로/../~.dtd">

6. 문서 유형 선언하기

< 외부 DTD 서브셋 문서가 XML 문서와 동일한 로컬 디렉토리에 있을 경우 >

- 새로운 마크업 언어를 개발할 때 테스트 용도로 많이 사용된다.

XML 문서	<?xml version="1.0" encoding="euc-kr"?> <!-- 문서 유형 선언 --> <!DOCTYPE 루트엘리먼트 SYSTEM "~.dtd">

< PUBLIC 키워드를 사용해서 문서 유형을 선언하는 경우 >

- PUBLIC 키워드는 주로 공개적인 사용을 목적으로 업체 및 단체에서 작성된 외부 DTD 서브셋 문서를 지정할 경우 사용된다.

문법	<?xml version="1.0" encoding="euc-kr"?> <!-- 문서 유형 선언 --> <! DOCTYPE 루트엘리먼트 PUBLIC "PUBLIC 식별자" "SYSTEM 식별자" >

6. 문서 유형 선언하기

- PUBLIC 키워드를 사용하기 위해서는 PUBLIC(공개) 식별자를 생성해야 한다.

문법	±//DTD를 개발 및 유지보수 업체명//DTD명 및 버전번호//사용된 언어

- 국제적으로 공인된 단체가 아닐 경우에는 '-'를 사용하고, ISO와 같은 국제적으로 공인된 단체는 '+' 기호를 붙인다.
- SYSTM 식별자에는 PUBLIC 식별자를 알지 못하는 응용프로그램을 위해서 DTD 문서를 실제로 다운로드할 수 있는 URL 경로를 기술해 주면 된다.

< 사용 예 >

HTML 문서	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

WML 문서	<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">

6. 문서 유형 선언하기

6-3 내부 DTD 서브셋에 대한 문서 유형 선언

- 일반적으로 새로운 마크업 언어는 외부 DTD 서브셋에서 정의된다.

문법	<! DOCTYPE 루트엘리먼트 [<!-- 새로운 DTD 내용 -->]>

XML문서	<?xml version="1.0" encoding="euc-kr"?> <!-- 문서 유형 선언 --> <!DOCTYPE booklist [<!ELEMENT booklist (book+)> <!ELEMENT book (#PCDATA)>]> <!--루트 엘리먼트 --> <booklist> <book>사랑과 영혼</book> </booklist>

6. 문서 유형 선언하기

< 외부 DTD를 내부 DTD에서 재정의 해서 사용하는 경우 >

- 비공개적인 사용 목적으로 만든 외부 DTD 서브셋의 내용을 내부 DTD 서브셋을 이용하여 재정의 할 경우에 사용하는 문서 유형 선언.

문법	<! DOCTYPE 루트엘리먼트 SYSTEM “SYSTEM 식별자” [<!-- 재 정의 DTD 내용 -->]>

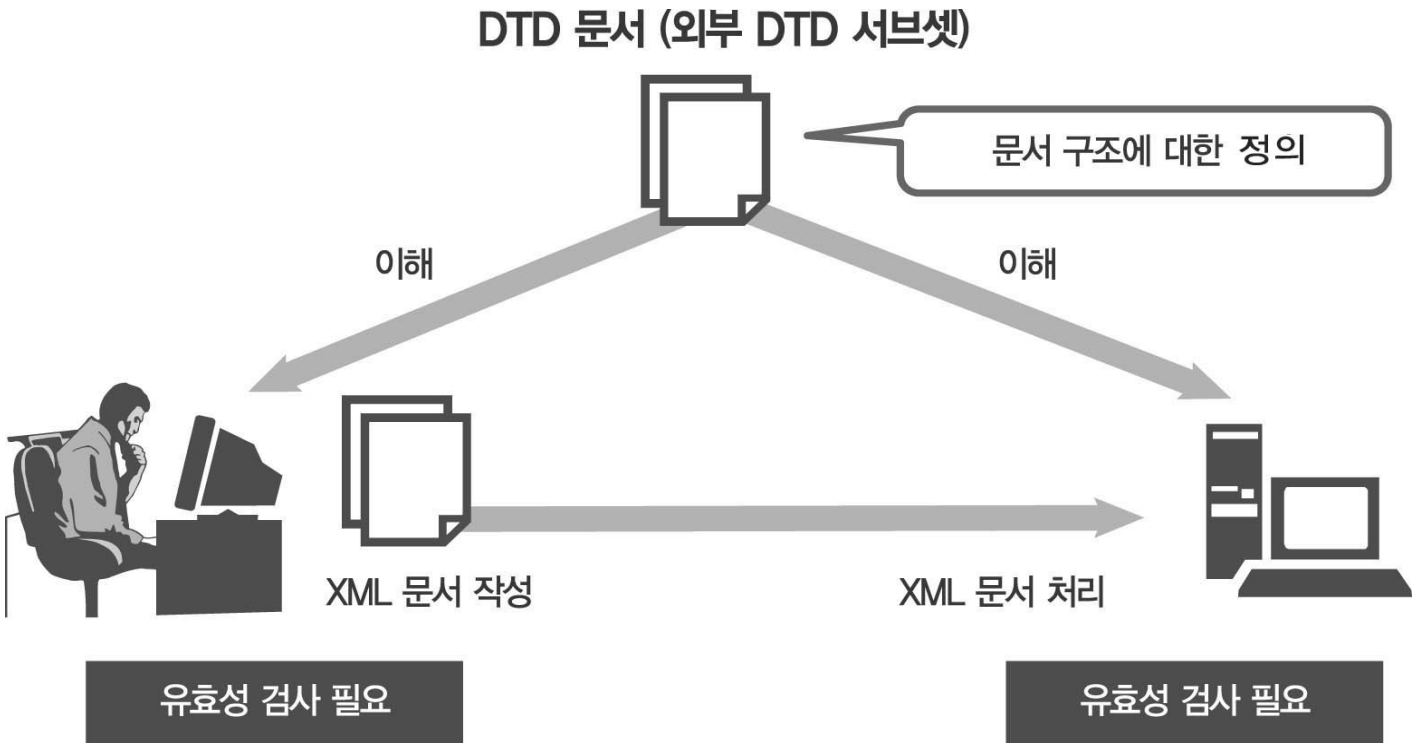
- 공개적인 사용 목적으로 만든 외부 DTD 서브셋의 내용을 내부 DTD 서브셋을 이용하여 재정의할 경우에 사용하는 문서 유형 선언.

문법	<! DOCTYPE 루트엘리먼트 PUBLIC “PUBLIC 식별자” “SYSTEM 식별자” [<!-- 재 정의 DTD 내용 -->]>

7. DTD 유효성 검사

7-1 유효성 검사의 필요성

- 유효성 검사 결과 이상이 없는 XML 문서를 '해당 DTD 문서에 대해 유효한 문서'라고 부른다.



7. DTD 유효성 검사

7-2 XML 문서 작성시의 유효성 검사

- XML 문서 작성하는 측 : XML 문서가 잘 작성되었는지 확인.(by validator)

7-3 XML 문서 처리시의 유효성 검사

- XML 문서를 처리하는 측 : 응용프로그램
  - XML 문서를 처리하기 전에 자기가 처리할 수 있는 구조로 되어있는지 유효성 검사를 해야 한다.

7-4 MSXML 파서를 이용한 유효성 검사 프로그램

- DTD로 문서 유형 선언이 된 XML 문서에 대해 MSXML 파서가 유효성 검사를 할 수 있는 방법.
  - iexmltls.exe 이용하기

http://msdn.microsoft.com/downloads/default.asp

7-5 다른 파서를 이용한 유효성 검사 프로그램

- MSXML 3.0에 있는 파서는 W3C XML 1.0 권고안에 소개되어 있는 내용을 완벽하게 따르지 않기 때문에 일부 유효성 검사 기능이 작동되지 않는다.
- Sun Microsystems에서 제공하는 파서 이용하기.
  (반드시 C:₩XMLSW가 되도록 한다.)

8. 텍스트 선언

- 텍스트 선언은 외부 DTD 서브셋과 외부 파라미터 엔티티, 외부 파스트 엔티티 문서의 첫 줄에 기술되는 내용으로 문서를 작성한 XML 문법의 버전과 사용된 인코딩 방식을 지정하는데 사용된다.
- XML 선언과 마찬가지로 텍스트 선언도 문서 내에서 생략 가능하다.

문법	<?xml version="버전번호" encoding="인코딩방식"?>

- encoding 속성은 해당 문서를 XML 파서가 어떤 인코딩방식으로 해석해야 되는지 기술.
- XML 선언에서는 생략 가능하지만, 텍스트 선언에서는 필수 속성이다.

외부 DTD 서브셋

```
<?xml version="1.0" encoding="euc-kr"?>
<!ELEMENT booklist (book+)>
<!ELEMENT book (#PCDATA)>
```

9. 주석

- 주석은 파서를 위한 것이 아니라 XML문서나 DTD를 작성하는 사람, 혹은 읽는 사람에게 부가적인 정보를 전달하기 위해서 사용한다.

외부 DTD 서브셋

```
<?xml version=”1.0” encoding="euc-kr"?>
<!-- 루트 엘리먼트 선언 -->
<!ELEMENT booklist (book+)>

    <!-- 자식 엘리먼트 선언 -->
    <!ELEMENT book (#PCDATA)>
```

10. 엘리먼트 선언

10-1 엘리먼트(ELEMENT) 선언 문법

- 엘리먼트 : 특정 사물이나 추상적인 개념을 구성하는 명사적인 특징
- XML 문서 : '기본 정보 단위'

문법	<!ELEMENT 엘리먼트명 컨텐트유형>

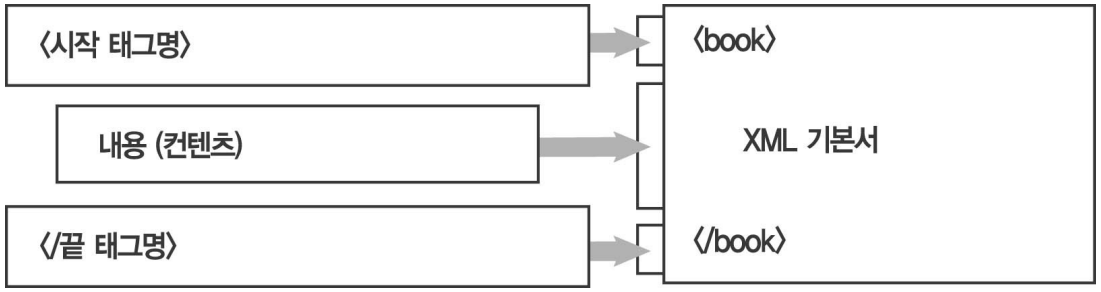
- 엘리먼트의 이름 작성 규칙

이름 작성 규칙	
1	이름은 문자(한글도 포함)로 시작할 수 있다. 그리고 특수 문자 중에서 '_'로 시작 할 수 있다. 하지만 숫자나 '.'으로 시작할 수는 없다.
2	두 번째 문자부터 숫자 및 특수 문자 "_", "-", "."도 가능하다.
3	이름에 공백을 포함시킬 수는 없다.
4	특수 문자 중에 ':'는 태그 이름에 포함할 수 있지만 네임스페이스에 관련된 기호이므로 되도록이면 사용하지 않는 것이 바람직하다.
5	태그 이름은 xml로 시작할 수 없다.

10. 엘리먼트 선언

10-2 엘리먼트의 종류

- XML 문서를 작성할 때 엘리먼트는 시작 태그명, 컨텐트, 끝 태그로 구성된다.



- 엘리먼트가 가지는 컨텐트에 따라 엘리먼트의 선언이 달라진다.

컨텐트 유형	설명
#PCDATA	■ 내용으로 문자 데이터만 갖는 엘리먼트
자식엘리먼트_리스트	■ 내용으로 자식 엘리먼트만 갖는 엘리먼트
EMPTY	■ 내용으로 아무 것도 갖지 않는 빈 엘리먼트
MIXED	■ 내용으로 문자 데이터나 자식 엘리먼트를 혼합 형태로 가지는 엘리먼트
ANY	■ 내용으로 문자 데이터나 모든 엘리먼트를 가지는 엘리먼트

10. 엘리먼트 선언

10-3 컨텐트로 문자 데이터만 갖는 엘리먼트 선언

- 엘리먼트명과 ()사이에는 반드시 공백을 두어야 한다.
- PCDATA(Parsed Character Data)는 XML 파서가 해석할 수 있는 문자 데이터라는 뜻으로 반드시 대문자로 기술되어야 한다.

문법	<!ELEMENT 엘리먼트명 (#PCDATA)>
DTD 문서	<!ELEMENT book (#PCDATA)>

XML 문서	<book>XML Fundamentals</book> <book><![CDATA[XML Fundamentals]]></book>

CDATA 섹션도 문자 데이터로 인식
(94page 참조)

10. 엘리먼트 선언

10-4 컨텐트로 자식 엘리먼트만을 갖는 엘리먼트 선언

- XML 문서는 트리 구조를 가지고 있다.

문법	<!ELEMENT 엘리먼트명 (자식엘리먼트1, 자식엘리먼트2, ...)>

- 괄호 안에 기술되는 자식 엘리먼트들도 하나의 독립적인 엘리먼트로서의 조건을 갖춰야 하기 때문에 DTD에서 별도의 엘리먼트 선언을 해야 한다.

DTD 문서	<!ELEMENT book (title, author)> <!ELEMENT title (#PCDATA)> <!ELEMENT author (#PCDATA)>

XML 문서	<book> <title>XML Fundamentals</title> <author>신민철</author> </book>

태그의 순서가 바뀌면 안된다.

10. 엘리먼트 선언

10-5 자식 엘리먼트 리스트 표현 방법

- 자식 엘리먼트의 순서나 등장 횟수를 지정하기 위해 여러 가지 연산 기호를 사용할 수 있다.

분 류	기 호	용 도
리스트연산자	,	엘리먼트의 작성 순서를 결정
선택 연산자	|	엘리먼트를 선택해서 작성
사용빈도 연산자	?	생략하거나 한번만 엘리먼트 작성
	+	한번 또는 여러 번의 동일 엘리먼트 작성 가능
	*	생략하거나 여러 번의 동일 엘리먼트 작성 가능
	기호없음	오직 한번만 엘리먼트 작성

10. 엘리먼트 선언

(1) 리스트 연산자

- 자식엘리먼트들의 작성 순서를 결정짓는 연산자이다.

문법	<!ELEMENT 엘리먼트명 (자식엘리먼트1, 자식엘리먼트2, ...)>
DTD문서	<!ELEMENT book (title, author, price, publisher)>
XML문서	<book> <title>XML Fundamentals</title> <author>신민철</author> <price>20000</price> <publisher>프리렉</publisher> </book>

10. 엘리먼트 선언

(2) 선택 연산자

- 연산자에 의해 분리된 엘리먼트들 중에 하나만 선택해서 작성해야 하는 연산자이다.

문법	<!ELEMENT 엘리먼트명 (child1, (child2 | child3), . . , childN)>
DTD문서	<!ELEMENT book (title, (author|writer), price)>
XML문서(1)	<book> <title>시인과 도둑</title> <author>이문열</author> <price>9000</price> </book>
XML문서(2)	<book> <title>시인과 도둑</title> <writer>이문열</writer> <price>9000</price> </book>

10. 엘리먼트 선언

(3) ? 연산자

- 해당하는 엘리먼트가 안나오거나 나온다면 한번만 나올 수 있다. (zero or only one)

문법	<!ELEMENT 엘리먼트명 (child1, child2?, child3, . . , childN)>
DTD문서	<!ELEMENT book (title, author, description?)>
XML문서	<book> <title>VB 프로그래밍</title> <author>홍길동</author> <description>쉽고 재미있는 책입니다.</description> </book> 생략 가능

10. 엘리먼트 선언

(4) + 연산자

- 해당 엘리먼트를 한번 이상, 여러 번까지 작성할 수 있다. (one or unlimited)

문법	<!ELEMENT 엘리먼트명 (child1, child2+, child3, . . , hildN)>
DTD문서	<!ELEMENT book (title, author+, publisher)>
XML문서	<book> <title>XML</title> <author>홍길동</author> <author>김하나</author> <author>이미영</author> <publisher>중앙</publisher> </book>

10. 엘리먼트 선언

(5) * 연산자

- 해당 엘리먼트를 여러 번까지 작성할 수 있도록 해준다. (no or unlimited)

문법	<!ELEMENT element_name (child1, child2*, child3, . . , childN)>
DTD문서	<!ELEMENT booklist (book*)> OR <!ELEMENT booklist (book)*>
XML문서(1)	<booklist> </booklist>
XML문서(2)	<booklist> <book> ~~~ </book> <book> ~~~ </book> </booklist>

10. 엘리먼트 선언

10-6 컨텐트를 갖지 않는 빈(Empty) 엘리먼트 선언

- 컨텐트로 자식 엘리먼트나 문자 데이터를 갖지 않는 엘리먼트

문법	<!ELEMENT 엘리먼트명 EMPTY>

- 빈 엘리먼트는 컨텐트가 없으므로 일반적으로 **속성(Attribute)**을 갖는다.

DTD 문서	<!ELEMENT image EMPTY>

XML 문서	<image/> 또는 <image></image>

10. 엘리먼트 선언

10-7 컨텐트로 혼합 형태를 갖는 엘리먼트 선언

- 엘리먼트의 컨텐츠로 문자 데이터가 올 수도 있고 자식 엘리먼트도 올 수도 있다.

문법	<!ELEMENT 엘리먼트명 (#PCDATA|child1|...|childN)*>

< 주의 사항 >

- 문자 데이터를 나타내는 #PCDATA는 제일 먼저 등장해야 한다.
- () 다음에 '*'를 반드시 붙여 주어야 한다.
- () 안에는 '|' 연산자만 사용할 수 있다.

DTD문서	<!ELEMENT author (#PCDATA|name)*>
XML 문서(1)	<author>이문열</author>
XML 문서(2)	<author> <name>존그레이</name> </author>

10. 엘리먼트 선언

10-8 컨텐트로 ANY 유형을 가지는 엘리먼트 선언

- DTD 내에 선언된 모든 엘리먼트들을 자식 엘리먼트로 가질 수 있는 컨텐트 유형.
- ANY 컨텐트 유형은 XML 문서를 처리하는 응용프로그램 개발을 어렵게 만들기 때문에 DTD에서 잘 사용되지 않는다.

문법	<!ELEMENT 엘리먼트명 ANY>
DTD문서	<!ELEMENT author ANY>
XML문서(1)	<author> 이문열 <tel>016-456-7890</tel> </author>
XML문서(2)	<author> <name>황석영</name> <tel>011-123-1234</tel> </author>

11. 속성 선언

11-1 속성(Attribute) 선언 문법

- 속성(Attribute)은 엘리먼트에 부가적인 정보를 줄 수 있으며, 속성으로 만들 수 있는 것은 자식 엘리먼트로도 만들 수 있다.
- 속성명은 XML 권고안의 이름 작성 규칙을 따른다.

문법	<!ATTLIST 엘리먼트명 속성명1 속성유형 디폴트선언 속성명2 속성유형 디폴트선언 … >

- 한 엘리먼트에 여러 개의 속성을 선언할 수 있지만 동일이름의 속성을 중복 선언할 수는 없다.
- 속성유형은 XML 문서에서 속성값으로 어떤 유형의 값이 올 수 있는지 지정한다.

DTD문서	<!ATTLIST book kind CDATA #REQUIRED>

XML문서	<book kind="소설">

11. 속성 선언

11-1 속성 선언 문법(계속)

DTD 문서 c4_1101.dtd

```
<?xml version="1.0" encoding="euc-kr"?>

<!-- 엘리먼트 선언 -->
<!ELEMENT booklist (book*)>
  <!ELEMENT book (title, author)>
     <!ELEMENT title (#PCDATA)>
     <!ELEMENT author (#PCDATA)>

<!-- 속성 선언 -->
<!ATTLIST book
   kind CDATA #REQUIRED>
```

XML 문서 : c4_1101.xml

```
<?xml version="1.0" encoding="euc-kr"?>
<!DOCTYPE booklist
                    SYSTEM
     "c4_1101.dtd">
<booklist>

  <book kind="소설">
    <title>시인과도둑</title>
    <author>이문열</author>
  </book>

</booklist>
```

11. 속성 선언

11-2 디폴트선언의 종류

- 디폴트선언에는 엘리먼트 작성시 속성을 생략할 수 있는지, 아니면 반드시 기술해야 되는 것인지를 지정한다.

< 디폴트선언에 기술할 수 있는 4가지의 유형 >

종 류	설 명
#IMPLIED	엘리먼트 작성시 속성을 생략할 수 있다.
#REQUIRED	엘리먼트 작성시 속성을 반드시 기술해야 한다.
#FIXED	속성값으로 지정한 문자열 이외의 값은 넣을 수 없다.
디폴트값	속성이 생략 되었을 경우, 기본 속성값으로 사용될 값을 기술한다.

11. 속성 선언

11-3 #IMPLIED 디폴트선언

- XML 문서에서 엘리먼트 작성시 속성을 생략해도 될 경우 #IMPLIED를 사용한다.

문법	<!ATTLIST 엘리먼트명 속성명 속성유형 #IMPLIED>
DTD문서	<!ATTLIST book kind CDATA #IMPLIED>
XML문서	<book kind="소설"> ~ </book> OR <book> ~ </book>

11-4 #REQUIRED 디폴트선언

- XML 문서에서 엘리먼트 작성시 속성을 반드시 기술하도록 할 경우 #REQUIRED를 사용한다.

문법	<!ATTLIST 엘리먼트명 속성명 속성유형 #REQUIRED>
DTD문서	<!ATTLIST book kind CDATA #REQUIRED>
XML문서	<book kind="소설"> ~ </book>

11. 속성 선언

11-5 #FIXED 디폴트선언

- 속성값이 고정된 값을 가지도록 하기 위해서는 디폴트선언에 #FIXED를 지정해 주면 된다.

문법	<!ATTLIST 엘리먼트명 속성명 속성유형 #FIXED “고정값”>
DTD문서	<!ATTLIST book kind CDATA #REQUIRED nation CDATA #FIXED "국내">
XML문서	<book kind="소설"> ~ </book> OR <book kind="소설" nation=”국내”> ~ </book>

11-6 디폴트값이 있는 디폴트선언

- 속성 기술을 생략하면 자동으로 디폴트값을 가지게 할 경우.

문법	<!ATTLIST 엘리먼트명 속성명 속성유형 “디폴트값”>
DTD문서	<!ATTLIST book kind CDATA #REQUIRED nation CDATA "국내">
XML문서	<book kind="소설"> ~ </book> OR <book kind="소설" nation=”원서”> ~ </book>

11. 속성 선언

11-7 속성 유형의 종류

- 속성 선언 시 사용할 수 있는 유형의 종류

속성 유형	설 명
CDATA	속성값으로 문자 데이터가 온다.
ENUMERATION	속성값으로 반드시 DTD에 나열된 값 중 하나가 와야 한다.
ID	속성값은 엘리먼트의 인스턴스(instance)를 구별하기 위한 식별자로서 사용되며 XML 문서 내에서 유일한 값을 지정해야 한다.
IDREF/ IDREFS	속성값은 XML 문서내에서 ID 속성값으로 지정된 값만 사용할 수 있다. ID 속성값을 참조하는 레퍼런스값으로 사용한다.
NMTOKEN/ NMTOKENS	속성값으로 이름 작성 규칙을 준수하는 데이터만 사용할 수 있다.
NOTATION	속성값은 DTD에 명시적으로 선언된 NOTATION명만 사용할 수 있다.
ENTITY	속성값은 DTD에 명시적으로 선언된 엔티티명만 가능하다.

11. 속성 선언

11-8 CDATA 속성 유형

- CDATA 속성 유형은 속성값으로 임의의 문자 데이터를 가질 수 있도록 한다.
- 속성값으로 '<' 또는 '&' 문자를 사용할 수 없다. (빌트인 엔티티 사용)

문법	<!ATTLIST 엘리먼트명 속성명 CDATA 디폴트선언>
DTD문서	<!ATTLIST book kind CDATA #REQUIRED title CDATA #REQUIRED>
XML문서(1)	<book kind="소설" title="시인과도둑"/>
XML문서(2)	<book kind="컴퓨터" title="XML & Java"/>

11. 속성 선언

11-9 ENUMERATION 속성 유형

- ENUMERATION(열거형) 타입은 속성의 값으로 올 수 있는 문자열을 DTD에 미리 열거해 놓고 그 중에서 하나를 선택하여 사용하도록 하는 속성 유형이다.

문법	<! ATTLIST 엘리먼트명 속성명 (속성값1 | 속성값2 | 속성값N) 디폴트선언 >
DTD문서	<!ATTLIST book kind (컴퓨터 | 소설 | 수필) “컴퓨터”>
XML문서(1)	<book kind="컴퓨터"> ~ </book> <book kind="소설"> ~ </book> <book kind="수필"> ~ </book>
XML문서(2)	<book> ~ </book>

11. 속성 선언

11-10 ID 속성 유형

- XML 문서에서 엘리먼트를 식별할 수 있도록 속성 유형에 ID를 지정해 줌으로서 유일한 값을 가지게 할 수 있다.
- 속성값은 숫자로 시작할 수 없으며, 중간에 공백이 와서도 안된다.
- 한 개의 엘리먼트는 ID 유형의 속성을 한 개만 가질 수 있다.
- XML 문서 전체에서 ID 유형의 속성은 여러 개 올 수 있지만 동일한 값을 가질 수는 없다.

문법	<!ATTLIST 엘리먼트명 속성명 ID 디폴트선언>
DTD문서	<!ATTLIST book id ID #REQUIRED kind (컴퓨터|소설|수필) "컴퓨터">
XML문서	<book id=”b1” kind=”컴퓨터”> ~ </book>

11. 속성 선언

11-11 IDREF / IDREFS 속성 유형

- IDREF(ID Reference) 속성 유형으로 선언된 속성의 속성값은 반드시 XML 문서 내에서 ID 속성 유형으로 선언된 속성의 값만 사용할 수 있다.
- IDREFS 속성 유형은 속성값으로 복수개의 ID 속성 유형으로 선언된 속성값을 가질 수 있다.

문법	<!ATTLIST 엘리먼트명 속성명 IDREF 디폴트선언>
DTD문서	<!ATTLIST kind id ID #REQUIRED> <!ATTLIST book id ID #REQUIRED kind IDREF #REQUIRED>
XML문서	<kinds> <kind id="k1">소설</kind> <kind id="k2">수필</kind> <kind id="k3">컴퓨터</kind> </kinds> <book id="b1" kind="k1"> ~ </book> <book id="b2" kind="k2"> ~ </book> <book id="b3" kind="k3"> ~ </book>

11. 속성 선언

11-12 NMTOKEN / NMTOKENS 속성 유형

- NMTOKEN 속성 유형은 임의의 문자 데이터를 속성값으로 가질 수 있지만, 해당 문자데이터는 XML 권고안에 언급된 이름 작성 규칙을 따라야 한다.

DTD문서	<!ATTLIST author name NMTOKEN #REQUIRED>
XML문서	<author name="이문열"/>

- NMTOKENS 속성 유형은 속성값으로 복수개의 NMTOKEN 속성 유형의 속성값을 가질 수 있도록 해준다.

DTD문서	<!ATTLIST author name NMTOKENS #REQUIRED>
XML문서	<author name="신민철 채규태 이규미"/>

11-13 NOTATION 속성 유형

11-14 ENTITY / ENTITIES 속성 유형

- 노테이션(NOTATION) 선언에서 설명.

12. 엔티티 선언

12-1 엔티티 개념

- 엔티티(Entity)는 XML 문서를 구성하는 물리적인 저장 단위(storage unit) 이다.
  - XML 문서 : 도큐먼트 엔티티 (XML문서의 물리적 저장단위)
  - 외부 DTD 서브셋 문서 : 외부 DTD 서브셋 엔티티

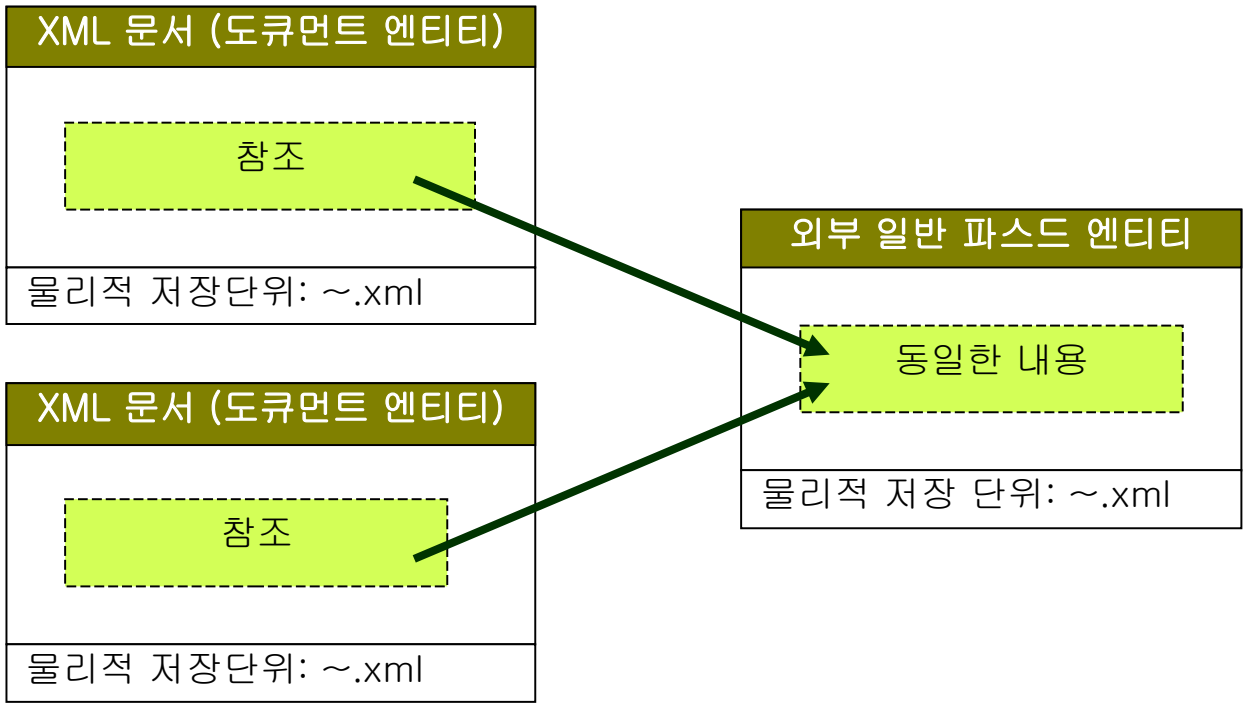
< 엔티티의 종류 >

엔티티명
■ 도큐먼트 엔티티
■ 외부 DTD 서브셋 엔티티
■ 빌트인(Built-in) 엔티티
■ 내부 일반 파스드 엔티티
■ 외부 일반 파스드 엔티티
■ 외부 일반 언파스드 엔티티
■ 내부 파라미터 엔티티
■ 외부 파라미터 엔티티

12. 엔티티 선언

12-2 엔티티의 용도

- XML 문서를 엔티티인 저장 단위로 나누는 이유는 엔티티의 재사용성을 높이기 위해서 이다.



12. 엔티티 선언

12-3 엔티티 분류

(1) 물리적인 저장 단위(파일)에 따른 구분

구 분	물리적 저장 단위
내부(internal)	DTD 내에 존재
외부(external)	독립적인 파일 형태로 존재

(2) 참조되는 위치에 따른 구분

참조되는 곳	엔티티명
일반(general)	XML 문서에서 참조하여 사용
파라미터(parameter)	DTD 내에서 참조하여 사용

(3) 문자 데이터로 구성되었는지에 따른 구분

참조되는 곳	엔티티명
파스드(parsed)	XML 파서가 해석할 수 있는 문자 데이터로 구성
언파스드(unparsed)	XML 파서가 해석할 수 없는 비 문자 데이터로 구성

12. 엔티티 선언

12-4 빌트인(Built-in) 엔티티

- 빌트인(Built-in) 엔티티는 미리 정의되어 있어, 별도의 엔티티 선언 없이 XML 문서에서 사용 가능한 엔티티를 말한다.

XML 문서에서 참조 방법	치환될 문자	의 미
&lt;	<	less-than
&gt;	>	greater-than
&amp;	&	ampersand
&quot;	“	double-quote
&apos;	‘	single-quote

XML 문서	<exam1>XML & Java</exam1> <exam2>x < y</exam2> <exam3 attr=”<xml>”/>

12. 엔티티 선언

12-5 내부 일반 파스드 엔티티

- 내부 일반 파스드 엔티티는 DTD 문서 내에서 특정 문자 데이터 값으로 선언되기 때문에 물리적인 저장 단위인 파일 형태를 가지지 않는다.

문법(DTD)	<!ENTITY 엔티티명 “대치할 문자 데이터”> 텍스트 선언 바로 밑에 선언
문법(XML)	&엔티티명;
DTD문서	<!ENTITY kr "대한민국"> <!ENTITY fr "프랑스"> <!ENTITY usa "미국">
XML문서	<author nation="&fr;">Jone</author> <nation>&usa;</nation>

12. 엔티티 선언

12-6 외부 일반 파스드 엔티티

- XML 문서에서 자주 사용되는 엘리먼트들을 XML 문서와 다른 물리적인 저장 단위로 저장한 것
  - 파일 확장자명은 무관하다.

문법 (DTD 1)	<!ENTITY 엔티티명 SYSTEM “외부일반파스드엔티티 URI 경로”>
문법 (DTD 2)	<!ENTITY 엔티티명 SYSTEM “외부일반파스드엔티티 파일명”> XML문서와 동일 디렉토리에 있는 엔티티 지정 <!ENTITY 엔티티명 SYSTEM “http://웹서버주소/경로명/.../외부일반파스드엔티티 파일명”>
문법 (XML)	&엔티티명; 특정 웹서버에 있는 엔티티 지정
DTD문서	<!ENTITY kind SYSTEM "c4_1203_1.xml">
XML문서	&kind;

선언

참조

12. 엔티티 선언

12-7 외부 일반 언파스드 엔티티

- 외부 일반 언파스드 엔티티는 비 문자 데이터로 이루어진 저장 단위를 말한다.
  - 음악 파일, 그림 파일, 동영상 파일 등등.
- 외부 일반 언파스드 엔티티를 XML 문서에서 참조하기 위해서는 DTD 내에서 NOTATION 선언이 필요하다. (13절에서 다루도록 함)

12. 엔티티 선언

12-8 내부 파라미터 엔티티

- 내부 파라미터 엔티티는 DTD 내용의 일부가 DTD 문서 내에서 문자열 형태의 값으로 선언된다.
- 내부 파라미터 엔티티는 DTD 내에서 선언하고 참조되기 때문에 선언하는 위치는 반드시 참조되기 전에 와야 한다. 일반적으로 텍스트 선언 바로 밑에 선언한다.

'%'의 앞과 뒤에는 반드시 공백 문자열이 와야 한다.

엔티티 선언	<!ENTITY % 엔티티명 "대치할 DTD 내용의 일부분">
엔티티 참조	%엔티티명;
DTD문서	<!ENTITY % maninfo "(name, age, tel)">
XML문서	<!ELEMENT chief %maninfo;> <!ELEMENT manager %maninfo;> <!ELEMENT waiter %maninfo;>

12. 엔티티 선언

12-9 외부 파라미터 엔티티

- 외부 파라미터 엔티티는 DTD 내용의 일부를 DTD 문서와 다른 물리적인 저장 단위로 저장한 것
- 여러 개의 DTD 문서 내에서 공통적으로 사용되는 부분을 외부 파라미터 엔티티로 선언해 두고, DTD 문서 내에서 엔티티명으로 참조해서 이용한다.

문법 (DTD 1)	<!ENTITY % 엔티티명 SYSTEM “외부파라미터엔티티 URI 경로”>	
문법 (DTD 2)	<!ENTITY 엔티티명 SYSTEM “외부파라미터 엔티티 파일명”> DTD 문서와 동일 디렉토리에 있는 엔티티 지정 <!ENTITY 엔티티명 SYSTEM “http://웹서버주소/경로명/.../외부파라미터 엔티티 파일명”> 특정 웹서버에 있는 엔티티 지정	선언
문법	%엔티티명;	참조
DTD문서	<!ENTITY % maninfo_element SYSTEM "c4_1205_1.dtd">	
DTD문서	%maninfo_element;	

12. 엔티티 선언

외부 파라미터 엔티티 : c4_1205_1.dtd

```
<?xml version="1.0" encoding="euc-kr"?>

<!-- 내부 파라미터 엔티티 선언 -->
<!ENTITY % maninfo "(name, age, tel)">

<!ELEMENT name (#PCDATA)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
```

DTD 문서 : c4_1205.dtd

```
<?xml version="1.0" encoding="euc-kr"?>

<!-- 외부 파라미터 엔티티 선언 -->
<!ENTITY % maninfo_element SYSTEM
                                "c4_1205_1.dtd">

<!-- 외부 파라미터 엔티티 참조 -->
%maninfo_element;

<!-- 엘리먼트 선언 -->
<!ELEMENT members chief|manager|waiter)*>
   <!ELEMENT chief %maninfo;>
   <!ELEMENT manager %maninfo;>
   <!ELEMENT waiter %maninfo;>
```

XML 문서 : c4_1205.xml

```
<?xml version="1.0" encoding="euc-kr"?>
<!DOCTYPE members SYSTEM "c4_1205.dtd">

<members>
   <chief>
      <name>홍길동</name>
      <age>45</age>
      <tel>011-123-1234</tel>
   </chief>

   <manager>
      <name>김민아</name>
      <age>30</age>
      <tel>016-234-4567</tel>
   </manager>

   <waiter>
      <name>박수빈</name>
      <age>23</age>
      <tel>017-567-6789</tel>
   </waiter>
</members>
```

13. 노테이션 선언

13-1 노테이션 이란

- 노테이션(notation)은 그림, 동영상, 음악 파일의 포맷을 식별하기 위해 사용되는 특별한 엘리먼트이다. (MIME 타입과 유사한 기능)
- XML 파서가 해석할 수 없는 비 문자 데이터의 포맷과, 데이터를 처리할 헬퍼(helper) 프로그램을 응용프로그램에게 알려준다.

13-2 노테이션 선언 문법

- 다음은 DTD 내부에서 노테이션을 선언하는 문법이다.

문법	<!NOTATION 노테이션명 SYSTEM "SYSTEM 식별자">

문법	<!NOTATION 노테이션명 PUBLIC "PUBLIC식별자" "SYSTEM 식별자">

MIME 타입

헬퍼 프로그램 정보

13. 노테이션 선언

13-3 NOTATION 속성 유형의 값으로 사용

- 노테이션명을 속성값으로 가지기 위한 선언

문법	<!ATTLIST 엘리먼트명 속성명 NOTATION (노테이션명1|노테이션명2|…) 디폴트선언 >
DTD문서	<!NOTATION gif PUBLIC "image/gif" "photoshop.exe"> <!NOTATION jpeg PUBLIC "image/jpeg" "photoshop.exe"> <!NOTATION bmp PUBLIC "image/bmp" "mspaint.exe"> <!ATTLIST image src CDATA #REQUIRED type NOTATION (gif|jpeg|bmp) #REQUIRED>
XML문서	<image src="book1.gif" type="gif"/>

13. 노테이션 선언

13-4 외부 일반 언파스드 엔티티의 포맷을 지정할 경우 사용

- 외부 일반 언파스드 엔티티를 XML 문서에서 참조하기 위해서는 외부 일반 언파스드 엔티티가 어떤 포맷으로 되어 있는지 DTD 내에 선언되어야 한다.

문법	<!ENTITY 엔티티_이름 SYSTEM "외부일반언파스드엔티티 URI 경로" NDATA 노테이션명>
DTD문서	<!-- 노테이션 선언 --> <!NOTATION gif PUBLIC "image/gif" "photoshop.exe"> <!--외부 일반 언파스드 엔티티 --> <!ENTITY front_image SYSTEM "front.gif" NDATA gif> <!--ENTITY 속성 유형의 속성값으로 사용--> <!ATTLIST book image ENTITY #REQUIRED>
XML문서	<book image="front_image"> ~ </book>

ENTITY / ENTITIES 속성 유형의 속성값으로만 사용된다.

- ENTITIES 속성 유형은 속성값으로 복수개의 엔티티 참조를 가질 수 있도록 해준다.

13. 노테이션 선언

DTD 문서 : c4_1302.dtd

```
<?xml version="1.0" encoding="euc-kr"?>

<!-- 노테이션 선언 -->
<!NOTATION gif    PUBLIC "image/gif" "photoshop.exe">
<!NOTATION jpeg PUBLIC "image/jpeg" "photoshop.exe">
<!NOTATION bmp PUBLIC "image/bmp" "mspaint.exe">

<!--외부 일반 언파스드 엔티티 -->
<!ENTITY front_image SYSTEM "book1.gif" NDATA gif>
<!ENTITY back_image SYSTEM "book2.bmp" NDATA bmp>

<!-- 엘리먼트 선언 -->
<!ELEMENT booklist (book*)>
  <!ELEMENT book (title, author)>
    <!ELEMENT title (#PCDATA)>
    <!ELEMENT author (#PCDATA)>

<!-- 속성 선언 -->
<!ATTLIST book
                    image ENTITY #REQUIRED>
```

XML 문서 : c4_1302.xml

```
<?xml version="1.0" encoding="euc-kr"?>
<!DOCTYPE booklist SYSTEM "c4_1302.dtd">

<booklist>

      <!-- 외부 일반 언파스드 엔티티 참조 -->
      <book image="front_image">
            <title>시인과도둑</title>
            <author>이문열</author>
      </book>

</booklist>
```

14. 컨디셔널 섹션

14-1 컨디셔널 섹션이란

- **컨디셔널 섹션**(Conditional Section)이란 XML 문서를 작성할 때 DTD 내용의 일부를 적용시킬 것인지 무시하도록 할 것인지를 결정하기 위해 사용된다.

14-2 컨디셔널 섹션 정의 문법

문법(1)	<!-- 컨디셔널 섹션 정의 --> <![INCLUDE [적용시킬 DTD 내용]]> <![IGNORE [무시할 DTD 내용]]>

14. 컨디셔널 섹션

- 컨디셔널 섹션 정의는 외부 DTD 서브셋과 외부 파라미터 엔티티 내에서만 사용 가능하다.

문법(2) DTD문서

```
<!-- 내부 파라미터 엔티티 -->
<!ENTITY % part1 “INCLUDE”>
<!ENTITY % part2 “IGNORE”>
<!-- 컨디셔널 섹션 정의 -->
<![%part1; [
   DTD 내용
]]>
<![%part2; [
   DTD 내용
]]>
```

문법(3) XML문서

```
<! DOCTYPE  루트엘리먼트  SYSTEM  “SYSTEM 식별자” [
  <!-- 내부 파라미터 엔티티 값 변경 -->
  <!ENTITY % part1 “IGNORE”>
  <!ENTITY % part2 “INCLUDE”>
] >
<루트 엘리먼트>
  ~~~
</루트 엘리먼트>
```

내부 DTD 서브셋에서 컨디셔널 섹션 제어하기

DTD 문서 : c4_1401.dtd

```
<?xml version="1.0" encoding="euc-kr"?>

<!-- 내부 파라미터 엔티티 선언 -->
<!ENTITY % en "INCLUDE">
<!ENTITY % kr "IGNORE">

<!-- 컨디셔널 섹션 정의 -->
<![%en;[
   <!-- 영문 엘리먼트 선언 -->
   <!ELEMENT booklist (book*)>
   <!ELEMENT book (title, author)>
   <!ELEMENT title (#PCDATA)>
   <!ELEMENT author (#PCDATA)>
]]>

<![%kr;[
   <!-- 한글 엘리먼트 선언 -->
   <!ELEMENT 책목록 (책*)>
   <!ELEMENT 책 (제목, 저자)>
   <!ELEMENT 제목 (#PCDATA)>
   <!ELEMENT 저자 (#PCDATA)>
]]>
```

XML 문서 : c4_1401.xml

```
<?xml version="1.0" encoding="euc-kr"?>
<!DOCTYPE booklist SYSTEM "c4_1401.dtd">

<booklist>
   <book>
      <title>시인과도둑</title>
      <author>이문열</author>
   </book>
</booklist>
```

XML 문서 : c4_1401.xml

```
<?xml version="1.0" encoding="euc-kr"?>

<!-- 내부 DTD 서브셋 정의 -->
<!DOCTYPE 책목록 SYSTEM "c4_1401.dtd" [
   <!ENTITY % en "IGNORE">
   <!ENTITY % kr "INCLUDE">
]>
<책목록>
   <책>
      <제목>시인과도둑</제목>
      <저자>이문열</저자>
   </책>
</책목록>
```

15. BML (Book MarkUp Language) 개발 예제

15-1 마크업 언어 개발 절차

< 마크업 언어를 개발하기 위해 필요한 절차 >

(1) 어떤 목적으로 마크업 언어를 개발할 것인가.
(2) 전체 구조는 어떻게 구성할 것인가.
(3) 엘리먼트 선택과 배치는 어떻게 할 것인가.
(4) 속성 선택과 배치는 어떻게 할 것인가.
(5) DTD 작성
(6) XML 문서 작성 및 유효성 검사하기.

15-2 BML 개발 목적

- 서점이나 도서관에서 관리하고 있는 책에 대한 정보들을 관리하기 위한 마크업 언어.
- 대형 서점에서는 재고도서를 관리하거나 새로 들어온 책에 대한 정보를 관리하기 위한 마크업 언어.

15. BML (Book MarkUp Language) 개발 예제

15-3 BML의 구조 설정

- 구조나 문법에 대한 정의는 누가, 어떤 방식으로 작성하느냐에 따라 전혀 다른 구조가 나올 수도 있기 때문에 자신만의 기능과 구조를 생각해서 작성하는 것이 중요하다.