4. DTD를 이용한 문서 구조 설계

DTD 개요

- DTD (Document Type Definition)
  - 새로운 마크업(markup) 언어를 정의 → 문서/데이터의 구조를 정의
    - 응용 시스템(프로그램)들 간에 문서/데이터의 교환 및 공유 가능
  - XML 문서에서 사용가능한 markup들의 집합과 사용 규칙을 정의
    - 엘리먼트(+ 속성), 개체(entity), 노테이션(notation) 등
  - DTD 문서의 예 (invoice.dtd)

```
<!ELEMENT Invoice (Header, Item+)>
<!ELEMENT Header (Date, BillTo)>
<!ATTLIST Header
          invoiceNumber CDATA #REQUIRED>
<!ELEMENT Item (description*)>
<!ATTLIST Item
          price CDATA #REQUIRED
          discount (promotion | regular) "regular">
<!ELEMENT Date ((Month,Day,Year) | (Day,Month,Year))>
<!ELEMENT BillTo (Address)>
…
```


DTD 개요

- 문서의 유효성
  - 특정 DTD에 따라 작성된 문서를 그 DTD에 대해 유효한 문서(valid document)라 함
    - 예: HTML 문서는 HTML DTD에 대해 유효한 문서
  - DTD를 이용하여 XML 문서가 유효한 문서인지 확인 가능 (유효성 검사)

invoice.dtd

```
<!ELEMENT Invoice (Header, Item+)>
<!ELEMENT Header (Date, BillTo)>
<!ATTLIST Header
  invoiceNumber CDATA #REQUIRED>
<!ELEMENT Item (description*)>
<!ATTLIST Item
  price CDATA #REQUIRED
  discount (promotion | regular) "regular">
<!ELEMENT Date
  ((Month,Day,Year)|(Day,Month,Year))>
<!ELEMENT BillTo (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT Month (#PCDATA)>
<!ELEMENT Day (#PCDATA)>
<!ELEMENT Year (#PCDATA)>
…
```

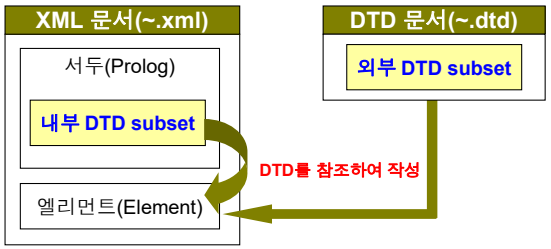
invoice.xml

XML 문서 안에 dtd파일이 있어야 함

```
<!DOCTYPE Invoice PUBLIC "InvoiceId" "invoice.dtd" >
<Invoice>
  <Header invoiceNumber="12345">
    <Date>
      <Month>July</Month>
      <Day>15</Day>
      <Year>2001</Year>
    </Date>
    <BillTo custNumber="X5739" name="Milton McGoo">
      1150 Eglinton Ave East, Toronto, Ontario
    </BillTo>
  </Header>
  <Item discount="promotion" price="57">
    <description>high speed GPU</description>
  </Item>
</Invoice>
```

DTD 사용 방법

- 문서 유형 선언(Document type declaration: DOCTYPE)
  - XML 문서가 어떤 DTD에 의해 (정의된 언어로) 작성된 것인지를 문서 내에 명시
  - DTD는 내부 DTD Subset과 외부 DTD Subset으로 구별됨
  - 외부 DTD subset을 정의한 문서를 DTD 문서라 함



DTD 사용 방법

- 내부 DTD Subset
  - 마크업 언어에 대한 DTD가 XML 문서 내부에서 정의됨
  - 내부 DTD subset은 주로 외부 DTD subset의 일부 내용을 XML 문서에서 재정의해서 사용할 목적으로 이용됨

내부 DTD subset을 이용한 XML 문서

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- 문서 유형 선언 : 내부 DTD subset을 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>
```


DTD 사용 방법

- 외부 DTD Subset
  - 마크업 언어에 대한 DTD가 별도의 문서(파일)로 정의됨
    - 여러 XML 문서에서 공통적으로 이용 가능
  - DTD 문서(파일)은 확장자로 "dtd" 사용
  - 새로운 마크업 언어를 정의 → 새로운 외부 DTD subset을 작성

외부 DTD subset : bml.dtd

```
<!ELEMENT booklist (book+)>
<!ELEMENT book (#PCDATA)>
```

XML 문서: book.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- 문서 유형 선언 : 외부 DTD subset -->
<!DOCTYPE booklist SYSTEM "bml.dtd">
<!-- 루트 엘리먼트 -->
<booklist>
  <book>XML and HTML</book>
  <book>XML and VB</book>
  <book>XML and Java</book>
</booklist>
```

이 내용에 booklist가 나오는걸 알 수 있다!

내부 DTD 문서 유형 선언

- XML 문서 내에서 DTD 정의

형식	<!DOCTYPE 루트 엘리먼트 [<!-- 새로운 DTD 정의 -->]>

- 예

XML 문서	<?xml version="1.0" encoding="UTF-8"?> <!-- 문서 유형 선언 --> <!DOCTYPE booklist [<!ELEMENT booklist (book+)> <!ELEMENT book (#PCDATA)>]> <!--루트 엘리먼트 --> <booklist> <book>XML and HTML</book> <book>XML and Java</book> </booklist>

외부 DTD 문서 유형 선언

- SYSTEM 식별자 이용
  - SYSTEM 키워드와 함께 DTD 문서의 경로나 URL을 기술

형식	<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE 루트엘리먼트 SYSTEM "SYSTEM 식별자" >

  - 국제적으로 공인되지 않은 단체에서 정의한 외부 DTD subset 문서를 지정할 경우 사용됨
    - 예: 특정 단체나 기업 내에서 DTD를 정의해서 비공개적으로 사용하는 경우
  - 사용 예
    - 특정 웹 서버 상에 DTD 문서가 존재할 경우 → URL 사용

```
<!DOCTYPE Invoice SYSTEM "http://domain-name/.../invoice.dtd">
```

    - 파일 시스템에서 외부 DTD subset 문서가 XML 문서와 동일한 위치에 있을 경우 → 파일 이름만 사용 가능(상대 경로)

```
<!DOCTYPE Invoice SYSTEM "~.dtd">
```

외부 DTD 문서 유형 선언

- PUBLIC 식별자 이용
  - PUBLIC 키워드와 함께 사용

생략 가능

형식	<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE 루트엘리먼트 PUBLIC "PUBLIC 식별자" "SYSTEM 식별자">

  - 공개적인 사용을 목적으로 기업이나 기관에서 정의한 외부 DTD subset 문서를 지정할 경우 사용됨
    - PUBLIC 식별자의 형식: ±//DTD 개발 단체명//DTD 이름 및 버전//사용 언어
      - ✓ '+' 기호: ISO와 같은 국제 공인 단체에서 정의한 DTD인 경우
      - ✓ '-' 기호: 비공인 단체에서 정의한 DTD인 경우
  - PUBLIC 식별자를 이해하지 못하는 응용프로그램을 위해 SYSTEM 식별자를 추가하여 DTD 문서의 URL 또는 파일 경로를 지정 가능
  - 사용 예

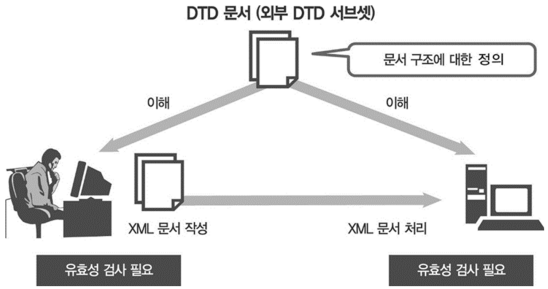
파일 이름(X) 식별자(O)

HTML 문서	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

언어 버전 언어

XML 문서의 유효성 검사

- XML 문서 작성 시 유효성 검사(validation) 실시
  - XML editor에 내장된 기능 사용
- XML 문서 이용 시 유효성 검사 실시
  - 응용 프로그램이 XML 문서를 처리하기 전에 유효성 검사 실시
  - XML parser의 유효성 검사 기능 이용
    - MSXML parser (MS Windows에 포함됨)
    - DOM / SAX parser (JRE에서 라이브러리로 제공)



DTD 구성요소

- 엘리먼트(element) 선언
  - 예: <!ELEMENT 이름 (#PCDATA)>
- 속성(attribute) 선언
  - 예: <!ATTLIST 학생 학년 (1|2|3|4)>
- 개체(entity) 선언
  - 예: <!ENTITY 학교 "동덕여자대학교">
- 노테이션(notation) 선언
  - 예: <!NOTATION gif SYSTEM "gifviewer.exe">
- 조건부 섹션(conditional section) 선언
  - 예: <![INCLUDE [<!ELEMENT 전화 (집, 사무실)>]]>
- 주석(comment)
  - 예: <!-- DTD 주석 -->

엘리먼트 선언

- 엘리먼트(element)
  - XML 문서를 구성하는 논리적 단위
  - 선언 형식

```
<!ELEMENT 엘리먼트명 contents>
```

    - 엘리먼트명: XML 엘리먼트의 이름
    - contents: XML 엘리먼트의 구조(데이터 타입) 정의
      - ✓ contents model 또는 contents category 정의

엘리먼트 선언

- Contents 유형
  - Contents model
    - #PCDATA : 일반적인 문자 데이터(text) 만을 포함
    - children : 자식 엘리먼트(child element)들을 포함
    - mixed : 문자 데이터와 자식 엘리먼트들을 함께 포함
  - Contents category
    - empty : 내용을 갖지 않는 빈 엘리먼트 선언
    - any : 임의의 구조(타입)의 데이터를 포함할 수 있음

엘리먼트 선언

- 문자 데이터(text) 선언
  - 형식

```
<!ELEMENT 엘리먼트명 (#PCDATA) >
```

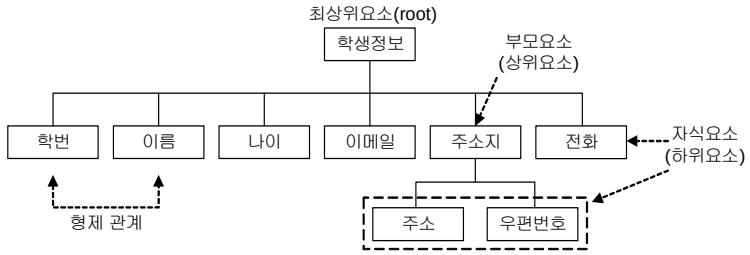
    - PCDATA(Parsed Character Data): XML parser가 해석하는 문자 데이터
  - 사용 예

```
<!DOCTYPE  학생정보 [
  <!ELEMENT 학생정보 (#PCDATA)>
]>

<학생정보>
   학생들의 정보를 문자 데이터(text)로 표현함.
</학생정보>
```


엘리먼트 선언

- 자식 엘리먼트 선언
  - 상위(부모) 엘리먼트에 포함되는 하위 엘리먼트 선언



  - 형식

```
<!ELEMENT 엘리먼트명 (child1, child2, child3, ...., childn) >
```

    - 주의: 괄호 안에 포함된 자식 엘리먼트들도 DTD에서 각각 별도의 엘리먼트 선언을 해야 함

엘리먼트 선언

- 자식 엘리먼트 선언
  - 사용 예

DTD 문서

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

XML 문서

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

    - 주의: <title>과 <author> 엘리먼트의 사용 순서가 바뀌면 안 됨

엘리먼트 선언

- 자식 엘리먼트들의 표현 방법
  - 자식 엘리먼트들의 순서나 반복 횟수를 지정하기 위해 여러 가지 선언자 기호를 사용함

선언자 기호	설명
,	선언된 엘리먼트들을 순서대로 자식 엘리먼트로 사용
|	선언된 엘리먼트들 중 하나를 선택해서 자식 엘리먼트로 사용
( )	선언된 엘리먼트들을 하나의 그룹으로 선언
*	선언된 엘리먼트가 0번 이상 사용될 수 있음 (0~n)
+	선언된 엘리먼트가 1번 이상 사용될 수 있음 (1~n)
?	선언된 엘리먼트가 0번 또는 1번 사용될 수 있음 (0~1)
무기호	특별한 기호가 없으면 선언된 엘리먼트를 반드시 한번만 사용

엘리먼트 선언

(1) 리스트 연산자

- 자식 엘리먼트들의 순서를 결정짓는 연산자

형식	**<!ELEMENT 엘리먼트명 (child1, child2, ..., child*n*)>**
***DTD* 문서**	<!ELEMENT book (title, author, price, publisher)>
***XML* 문서**	<book> <title>XML Fundamentals</title> <author>신민철</author> <price>20000</price> <publisher>프리렉</publisher> </book>

엘리먼트 선언

(2) 선택 연산자

- 연산자에 의해 분리된 엘리먼트들 중 하나만 선택 가능

형식	**<!ELEMENT 엘리먼트명 (child1, (child2 | child3), ..., child*n*)>**
DTD 문서	<!ELEMENT book (title, **(author | writer)**, price)>
XML 문서(1)	<book> <title>시인과 도둑</title> <author>이문열</author> <price>9000</price> </book>
XML 문서(2)	<book> <title>시인과 도둑</title> <writer>이문열</writer> <price>9000</price> </book>

엘리먼트 선언

(3) ? 연산자

- 해당하는 엘리먼트가 생략되거나 또는 한번만 사용될 수 있음 (zero or only one)

형식	**<!ELEMENT 엘리먼트명 (child1, child2?, child3, ..., child*n*)>**
DTD 문서	<!ELEMENT book (title, author, **description?**)>
XML 문서	<book> <title>VB 프로그래밍</title> <author>신민철</author> <description>쉽고 재미있는 책입니다.</description> </book>

<description> 엘리먼트는 생략 가능

엘리먼트 선언

(4) + 연산자

- 해당 엘리먼트를 한 번 이상 여러 번 사용 가능 (one or unlimited)

형식	**<!ELEMENT 엘리먼트명 (child1, child2+, child3, …, child*n*)>**
DTD 문서	<!ELEMENT book (title, **author+**, publisher)>
XML 문서	<book> <title>XML 기본서</title> <author>신민철</author> <author>채규태</author> <author>이규미</author> <publisher>프리렉</publisher> </book>

엘리먼트 선언

(5) * 연산자

- 해당 엘리먼트를 생략하거나 여러 번 사용 가능 (no or unlimited)

형식	**<!ELEMENT 엘리먼트명 (child1, child2*, child3, …, child*n*)>**
DTD 문서	<!ELEMENT booklist (**book***) >
XML 문서*(1)*	<booklist> </booklist>
XML 문서*(2)*	<booklist> <book> ~~~ </book> <book> ~~~ </book> </booklist>

엘리먼트 선언

- 혼합형 선언
  - 엘리먼트의 content로 문자 데이터와 자식 엘리먼트를 동시에 선언할 때 사용
  - 형식

문자데이터를 의미

<!ELEMENT 엘리먼트명 **(#PCDATA | child1 | child2 | ... | child*n*)*** >

  - 주의사항
    - 문자데이터(#PCDATA)를 먼저 선언
    - 혼합 내용의 반복성은 0번 이상(*)으로 정의
    - 예

<!ELEMENT p (#PCDATA| a | ul | b | i | em)* >

```
<p>
    Oh~ <em>Hello</em> <b>world</b>
</p>
```


엘리먼트 선언

- EMPTY 선언
  - Content를 갖지 않는 빈 엘리먼트를 정의하기 위해 사용
    - 빈 엘리먼트의 경우 content가 없으므로 일반적으로 속성을 가짐
  - 형식

<!ELEMENT 엘리먼트명 **EMPTY**>

  - 엘리먼트 사용 방법
    - <엘리먼트명></엘리먼트명> 또는 <엘리먼트명 />
      - ✓ 주의: 시작 태그와 종료 태그 사이에 공백이 있으면 안 됨
  - 예

```
<!ELEMENT 학번 EMPTY>
<!ELEMENT 이름 EMPTY>
```

```
<학번></학번>
<학번/>
```

엘리먼트 선언

- ANY 선언
  - 엘리먼트의 content에 대해 조건을 두지 않을 경우에 사용
    - 정의되는 엘리먼트는 임의의 구조를 가질 수 있음
      - ✓ DTD 내에 선언된 모든 엘리먼트들을 자식 엘리먼트로 가질 수 있음
    - ANY content 유형은 XML 문서 처리를 위한 응용프로그램 개발을 어렵게 만들기 때문에 잘 사용되지 않음
  - 형식

```
<!ELEMENT 엘리먼트명 ANY>
```


속성 선언

- 속성(attribute) 선언
  - 속성은 엘리먼트에 부가적인 정보를 제공함
  - 엘리먼트와 관련된 속성 이름, 속성 유형, 속성 기본값 등을 정의
  - 형식

형식	<!ATTLIST 엘리먼트명 속성명1 속성유형 속성기본값 속성명2 속성유형 속성기본값 … >

    - 엘리먼트명, 속성명: XML 문서에서 선언된 엘리먼트 이름을 명시한 후 그 엘리먼트에 적용될 속성 이름을 지정
      - ✓ XML 권고안의 XML 이름 작성 규칙을 따름
      - ✓ 하나의 엘리먼트에 같은 이름의 속성을 여러 개 선언할 수 없음
    - 속성 유형: 열거형, 문자열, 토큰 타입
    - 속성기본값: 속성의 초기 값, #FIXED, #IMPLIED, #REQUIRED

CDATA

속성 선언

- 예

DTD 문서 c4_1101.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- 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="UTF-8"?>
<!DOCTYPE booklist
                SYSTEM "c4_1101.dtd">
<booklist>

  <book kind="소설">
    <title>시인과 도둑</title>
    <author>이문열</author>
  </book>

</booklist>
```

안 쓰면 빈 칸으로 기본값을 준 것! (Required)

속성 선언

- 속성 기본값 (디폴트 선언)
  - 엘리먼트 작성 시 속성을 생략할 수 있는지, 반드시 기술해야 되는지 등을 지정
  - 속성이 생략될 때 속성이 갖는 디폴트 값 지정 가능

값	의미
#REQUIRED	속성을 반드시 기술해야 함
#IMPLIED	속성을 생략 가능
#FIXED *value*	속성의 값을 미리 지정 (값 변경 불가)
value	속성이 생략될 때 사용될 default 값 지정

속성 선언

- 사용 예

형식	<!ATTLIST 엘리먼트명 속성명 속성유형 #REQUIRED>
DTD 문서	<!ATTLIST book kind CDATA #REQUIRED>
XML 문서	<book kind="소설"> ~ </book> <book> ~ </book> (오류) → kind 생략해서

형식	<!ATTLIST 엘리먼트명 속성명 속성유형 #IMPLIED>
DTD 문서	<!ATTLIST book kind CDATA #IMPLIED>
XML 문서	<book kind="소설"> ~ </book> <book> ~ </book> (kind 속성 생략)

속성 선언

- 사용 예

형식	<!ATTLIST 엘리먼트명 속성명 속성유형 #FIXED "고정값">
DTD 문서	<!ATTLIST book kind CDATA #REQUIRED nation CDATA #FIXED "국내">
XML 문서	<book kind="소설"> ~ </book> (nation="국내") <book kind="소설" nation="국내"> ~ </book> <book kind="소설" nation="해외"> ~ </book> (오류)

형식	<!ATTLIST 엘리먼트명 속성명 속성유형 "디폴트 값">
DTD 문서	<!ATTLIST book kind CDATA #REQUIRED nation CDATA "국내">
XML 문서	<book kind="소설"> ~ </book> (nation="국내") <book kind="소설" nation="해외"> ~ </book>

속성 선언

- 속성 유형
  - 문자열 타입(CDATA): 임의의 문자열을 속성 값으로 가짐
  - 열거형 타입: 열거된 여러 값들 중 하나를 속성 값으로 가짐
  - 토큰 타입: 다음과 같은 키워드를 사용하여 속성 값 유형을 정의

속성 유형	설 명
ID	속성값은 엘리먼트들을 구별하기 위한 식별자로 사용되며, XML 문서 내에서 유일한 값을 가져야 함
IDREF/ IDREFS	속성값은 XML 문서 내에서 ID 속성으로 사용된 값만 사용할 수 있음 다른 엘리먼트의 ID 속성값을 참조하기 위해 사용됨
NMTOKEN/ NMTOKENS	속성값은 XML 이름 작성 규칙을 준수하는 문자데이터만 사용할 수 있음
NOTATION	속성값은 DTD에 명시적으로 선언된 notation명만 사용할 수 있음
ENTITY/ ENTITIES	속성값은 DTD에 명시적으로 선언된 entity명만 사용 가능함

속성 선언

- 문자열 타입(CDATA)
  - 속성 값에 대해 <, >, &, ‘, “ 등과 같은 특수문자를 제외하고 어떤 문자열도 사용 가능
  - 특수문자를 사용할 때는 개체 참조 이용
    - < > & ' " 등
  - 예

형식	<!ATTLIST 엘리먼트명 속성명 CDATA 속성기본값>
DTD 문서	<!ATTLIST book kind CDATA #REQUIRED title CDATA #REQUIRED>
XML 문서(1)	<book kind="소설" title="시인과 도둑"/>
XML 문서(2)	<book kind="컴퓨터" title="XML & Java"/>

속성 선언

- 열거형 타입
  - DTD에 열거된 값들 중에서 하나를 선택하여 사용하도록 함
    - 괄호 안에 나열된 값 이외의 값은 사용 불가
  - 예

형식	<!ATTLIST 엘리먼트명 속성명 (속성값1 | 속성값2 | 속성값N) 속성기본값>
DTD 문서	<!ATTLIST book kind (컴퓨터 | 소설 | 수필) "컴퓨터">
XML 문서(1)	<book kind="컴퓨터"> ~ </book> <book kind="소설"> ~ </book> <book kind="수필"> ~ </book> <book kind="시"> ~ </book> (오류)
XML 문서(2)	<book> ~ </book> (kind="컴퓨터")

속성 선언

- ID 타입
  - XML 문서에서 엘리먼트들을 유일하게 식별해야 할 경우 사용
  - 예

형식	<!ATTLIST 엘리먼트명 속성명 ID 속성기본값>
DTD 문서	<!ATTLIST book bid ID #REQUIRED
XML 문서	<book bid="b1"> ~ </book> <book bid="b2"> ~ </book> <book bid="b3"> ~ </book>

  - 주의사항
    - 속성기본값은 #IMPLIED 또는 #REQUIRED 만 사용 가능
    - 각 엘리먼트에서 ID 타입 속성은 하나만 정의 가능
    - 문서 내에서 ID 타입 속성의 값은 중복될 수 없음
    - ID 타입 속성의 값은 반드시 XML 이름 작성 규칙을 따라야 함
      - ✓ 예: 반드시 문자(letter)로 시작, 특수기호 사용 제한

속성 선언

- IDREF(S) 타입
  - ID 타입으로 선언된 속성이 갖는 값들 중 하나를 가짐
    - IDREF(S) 타입 속성의 값은 문서 내에 포함된 ID 속성 값들 중에 일치(참조)하는 것이 반드시 있어야 함
  - 형식

형식	<!ATTLIST 엘리먼트명 속성명 IDREF(S) 속성기본값>

    - 속성기본값은 #IMPLIED 또는 #REQUIRED 만 사용 가능
  - IDREFS: ID 타입으로 선언된 속성 값들 중 여러 개를 동시에 참조 가능
    - 속성 값들은 공백 문자로 구분

속성 선언

- 예

```
<!ELEMENT  employee EMPTY>
<!ELEMENT  student EMPTY>
<!ELEMENT  department EMPTY>
<!ELEMENT  team EMPTY>
<!ATTLIST  employee    empno        ID      #REQUIRED>
<!ATTLIST  student     sno          ID      #REQUIRED>
<!ATTLIST  department  deptManager  IDREF   #IMPLIED>
<!ATTLIST  team        members      IDREFS  #IMPLIED>
```

```
<employee empno="e1" />
<employee empno="e2" />
<employee empno="e2" />          <!-- 오류: ID 속성 값 중복 발생 -->
<student  sno="s1" />
<department  deptManager="e1" />
<department  deptManager="s1" />
<department  deptManager="e3" />   <!-- 오류: e3가 ID 타입 속성의 값이 아님 -->
<department  deptManager="e1 e2" />  <!-- 오류: deptManager는 IDREF 타입 -->
<team        members="e1 e2 s1" />   <!-- IDREFS 타입이므로 가능 -->
```

경우 상황이 생길 수도 있겠으나 ID 값이 겹쳐서는 안 됨

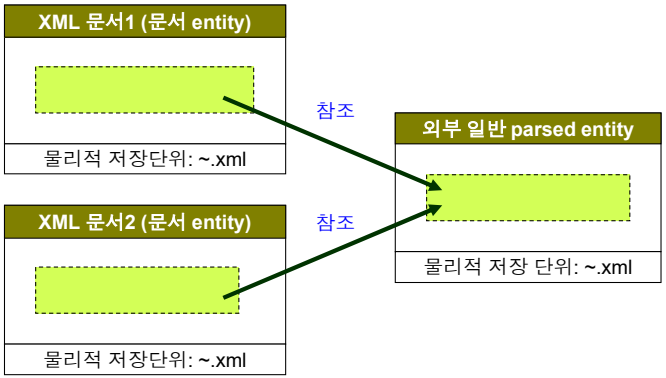
개체(Entity) 선언

- 개체(Entity)의 개념
  - XML 문서를 구성하는 물리적인 저장 단위(storage unit)
  - 종류
    - 문서 entity: 일반적인 XML 문서
    - 외부 DTD subset entity: DTD 문서
    - 빌트인(built-in) entity
    - 내부 일반 parsed entity
    - 외부 일반 parsed entity
    - 외부 일반 unparsed entity
    - 내부 파라미터(parameter) entity
    - 외부 파라미터(parameter) entity

DTD 에서 참조됨

개체 선언

- 개체의 사용 목적
  - XML 문서에서 구성요소의 공유 및 재사용



개체 선언

- 개체 분류
  - 물리적 저장 단위에 따른 구분

구분	물리적 저장 단위
내부(internal)	같은 **DTD 내**에 정의
외부(external)	**별도의 파일(XML/DTD 문서)**로 정의

  - 참조되는 위치에 따른 구분

구분	참조되는 곳
일반(general)	**XML 문서**에서 참조하여 사용
파라미터(parameter)	**DTD 문서**에서 참조하여 사용

  - XML parser가 해석 가능한지 여부에 따른 구분
    - 일반 개체만 해당됨

구분	개체 내용
Parsed	XML parser가 해석할 수 있는 내용으로 구성
Unparsed	XML parser가 해석할 수 없는 **비-문자 데이터**로 구성

개체 선언

- 빌트인(built-in) entity
  - 미리 정의되어 있는 개체

XML 문서에서 참조 방법	치환될 문자	의미
<	<	less-than
>	>	greater-than
&	&	ampersand
"	"	double-quote
'	'	single-quote

  - 사용 예

XML 문서		
	<ex1>XML **&** Java</ex1>	→ XML & Java
	<ex2>x **<** y</ex2>	→ x < y
	<ex3 attr="**<**xml**>**" />	→ "<xml>"

개체 선언

- 내부 일반 parsed entity
  - DTD 내에서 문자 데이터로 선언되고, 별도의 물리적인 파일 형태를 갖지 않음 (내부, parsed)
  - XML 문서에서 참조 (일반)

형식(DTD)	<!ENTITY entity명 "대체할 문자 데이터">
참조 형식 (XML)	&entity명;

DTD 문서	<!ENTITY kr "대한민국"> <!ENTITY writer "Park"> <!ENTITY copyright "@Copyright Dongduk Women's Univ.">
XML 문서	<author nation="&kr;">&writer; &copyright;</author> → <author nation="대한민국">Park @Copyright Dongduk Women's Univ.</author>

개체 선언

- 외부 일반 parsed entity
  - XML 문서에서 자주 사용되는 엘리먼트들을 별도의 물리적인 파일로 저장한 것 (외부, parsed)
  - XML 문서에서 참조 (일반)

형식 (DTD)	<!ENTITY entity명 SYSTEM "entity file의 경로">
참조 형식 (XML)	&entity명;

DTD 문서	<!ENTITY kind SYSTEM "c4_1203_1.xml">
entity file (c4_1203_1.xml)	<kinds> <kind id="k1">컴퓨터</kind> <kind id="k2">소설</kind> <kind id="k3">수필</kind> </kinds>
XML 문서	&kind; → 위 entity file의 내용으로 치환됨

- p.175~177 참조

개체 선언

- 외부 일반 unparsed entity
  - 비-문자 데이터로 이루어진 물리적인 저장 단위를 참조
    - 음악 파일, 그림 파일, 동영상 파일 등
  - 선언 (DTD 문서)
    - Notation과 외부 일반 unparsed entity 선언 (슬라이드 #49~50 참조)
    - 엘리먼트에 대해 ENTITY 타입의 속성 선언
  - 참조 (XML 문서)
    - ENTITY 타입 속성으로 외부 일반 unparsed entity의 이름을 지정(참조)

DTD 문서	<!NOTATION jpeg SYSTEM "imageviewer.exe"> → Notation **jpeg** 선언 <!ENTITY pic SYSTEM "pic.jpg" NDATA jpeg> → Entity **pic** 선언 <!ELEMENT picture EMPTY> <!ATTLIST picture img ENTITY #IMPLIED> → ENTITY 타입 속성 **img** 선언
XML 문서	<picture img="pic" /> → **pic** entity를 통해 pic.jpg 파일 참조

개체 선언

- 내부 파라미터(parameter) entity
  - DTD 내용의 일부를 DTD 문서 내에서 참조하기 위해 선언
  - DTD 내에서 선언되고 참조되기 때문에 선언되는 위치는 반드시 참조되기 전에 와야 함

형식(DTD)	<!ENTITY % entity명 "대치할 DTD 내용의 일부분">
참조 형식 (DTD)	%entity명;

DTD 문서	<!ENTITY % maninfo "(name, age, tel)"> 주의: % 다음에 반드시 공백 필요! <!ELEMENT chef %maninfo;> → <!ELEMENT chef (name, age, tel)> <!ELEMENT manager %maninfo;> → <!ELEMENT manager (name, age, tel)> <!ELEMENT waiter %maninfo;> → <!ELEMENT waiter (name, age, tel)>

개체 선언

- 외부 파라미터(parameter) entity
  - DTD 내용의 일부를 DTD 문서와 다른 물리적인 저장 단위로 저장한 것
  - 여러 DTD 문서들에서 공통적으로 사용되는 부분을 별도의 파일로 저장하고, 외부 파라미터 entity를 선언한 후 각 DTD 문서에서 참조해서 이용

형식 (DTD)	<!ENTITY % entity명 SYSTEM "entity 파일의 경로">
참조 형식 (DTD)	%entity명;

DTD 문서	<!ENTITY % maninfo_element SYSTEM "c4_1205_1.dtd"> %maninfo_element; → c4_1205_1.dtd 파일의 내용 참조

개체 선언

- parameter entity 사용 예

외부 parameter entity : c4_1205_1.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- 내부 parameter entity 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="UTF-8"?>
<!-- 외부 parameter entity 선언 -->
<!ENTITY % maninfo_element SYSTEM
                    "c4_1205_1.dtd">
<!-- 외부 parameter entity 참조 -->
%maninfo_element;
<!-- 엘리먼트 선언 -->
<!ELEMENT members
          (chef|manager|waiter)*>
  <!ELEMENT chef %maninfo; >
  <!ELEMENT manager %maninfo; >
  <!ELEMENT waiter %maninfo; >
```

XML 문서 : c4_1205.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE members SYSTEM
    "c4_1205.dtd">

<members>
   <chef>
     <name>홍길동</name>
      <age>35</age>
     <tel>02-123-1234</tel>
   </chef>
   <manager>
     <name>김민아</name>
     <age>30</age>
     <tel>02-234-4567</tel>
   </manager>
   <waiter>
     <name>박수빈</name>
     <age>23</age>
     <tel>02-567-6789</tel>
   </waiter>
</members>
```


Notation 선언

- Notation 선언과 사용
  - 그림, 동영상, 음악 등 이진(binary) 파일의 포맷(format)을 식별
    - MIME type 이용 (예: text/html, image/jpeg, video/mpeg)
  - XML parser가 해석할 수 없는 비-문자 데이터의 포맷과, 데이터를 처리할 응용 프로그램(helper application)을 지정
  - Notation 선언

응용 프로그램

형식	<!NOTATION notation명 SYSTEM "SYSTEM 식별자">

형식	<!NOTATION notation명 PUBLIC "PUBLIC식별자" "SYSTEM 식별자">

MIME type

  - 예

```
<!NOTATION gif PUBLIC "image/gif" "photoshop.exe">
<!NOTATION jpeg PUBLIC "image/jpeg" "photoshop.exe">
```


Notation 선언

1. NOTATION 타입의 속성 값으로 사용
   - Notation명을 속성 값으로 사용
   - 형식

형식	<!ATTLIST 엘리먼트명 속성명 NOTATION (notation명1|notation명2|…) "기본값선언" >

   - 주의사항
     - EMPTY 엘리먼트는 NOTATION 타입의 속성을 가질 수 없음
     - 각 엘리먼트는 NOTATION 타입의 속성을 최대 하나만 가질 수 있음

Notation 선언

– 사용 예

```
<?xml version="1.0" encoding="UTF-8"?>
<!NOTATION gif PUBLIC "image/gif" "photoshop.exe">
<!NOTATION jpeg PUBLIC "image/jpeg" "photoshop.exe">
<!NOTATION bmp PUBLIC "image/bmp" "mspaint.exe">
<!ELEMENT student (name, picture, address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT picture (#PCDATA)>
<!ATTLIST picture src CDATA #REQUIRED
                  type NOTATION (gif | jpeg | bmp) "gif" >
<!ELEMENT address (#PCDATA)>
```

NOTATION 타입의 속성 정의

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student SYSTEM "notation.dtd">
<student>
  <name> 홍길동 </name>
  <picture src="pic.jpg" type="jpeg"></picture>
  <address> 서울시 성북구 월곡동 </address>
</student>
```

jpeg notation 사용

Notation 선언

2. 외부 일반 unparsed entity의 포맷을 지정하기 위해 사용
   – 외부 일반 unparsed entity를 정의하기 위해서는 그 entity가 어떤 포맷으로 저장되어 있는지를 선언해야 함

형식	<!ENTITY entity명 SYSTEM "entity 파일의 경로" NDATA notation명>

Notation 선언

– 사용 예 1

"외부 일반 unparsed entity" 정의

```
<?xml version="1.0" encoding="UTF-8"?>
<!NOTATION jpeg SYSTEM "imageviewer.exe">
<!ENTITY pic SYSTEM "pic.jpg" NDATA jpeg>
<!ELEMENT student (name, picture, address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT picture EMPTY>
<!ATTLIST picture img ENTITY #REQUIRED>
<!ELEMENT address (#PCDATA)>
```

jpeg notation 사용

entity 타입의 속성 정의

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student SYSTEM "notation.dtd">
<student>
   <name> 홍길동 </name>
   <picture img="pic"/>
   <address>서울시 성북구 월곡동</address>
</student>
```

속성 값으로 pic entity 참조

Notation 선언

– 사용 예 2 (p.187)

DTD 문서 : c4_1302.dtd

```
<!-- notation 선언 -->
<!NOTATION gif PUBLIC "image/gif" "paint.exe">
<!NOTATION bmp PUBLIC "image/bmp"
    "paint.exe">

<!--외부 일반 unparsed entity -->
<!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)>
<!-- entity type 속성 선언 -->
<!ATTLIST book image ENTITY #REQUIRED>
```

XML 문서 : c4_1302.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE booklist SYSTEM
                    "c4_1302.dtd">

<booklist>
   <!-- 외부 일반 unparsed entity 참조 -->
    <book image="front_image">
        <title>시인과 도둑</title>
        <author>이문열</author>
    </book>
</booklist>
```

조건부 Section 선언

- 조건부 Section 선언과 사용
  - DTD 문서 내부에서 어떤 조건에 따라 DTD의 내용을 포함하거나 포함하지 않도록 하기 위해 사용
  - 조건부 Section 선언 형식

```
<![INCLUDE [
   적용시킬 DTD 내용
]]>
<![IGNORE  [
   무시할 DTD 내용
]]>
```

    - INCLUDE 키워드로 정의한 부분은 유효성 검사 대상
    - IGNORE 키워드로 정의한 부분은 XML parser가 읽기는 하지만 유효성 검사에서는 제외

조건부 Section 선언

  - 사용 예 1

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % student_view "IGNORE">
<!ENTITY % professor_view "INCLUDE">
<![%stduent_view;
[
 <!ELEMENT students (student)>
 <!ELEMENT student (sno, name, age)>
]]>
<![%professor_view;
[
 <!ELEMENT students (student)>
 <!ELEMENT student (sno, name, age, phone, address)>

 <!ELEMENT phone (#PCDATA)>
 <!ELEMENT address (#PCDATA)>
]]>
<!ELEMENT sno (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT age (#PCDATA)>
```


조건부 Section 선언

  - 사용 예 2 (pp.191~192)

DTD 문서 : c4_1401.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- 내부 parameter entity 선언 -->
<!ENTITY % en "INCLUDE">
<!ENTITY % kr "IGNORE">

<!-- Conditional Section 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="UTF-8"?>
<!DOCTYPE booklist SYSTEM "c4_1401.dtd">
<booklist>
   <book>
      <title>시인과 도둑</title>
      <author>이문열</author>
   </book>
</booklist>
```

XML 문서 : c4_1401.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- 내부 DTD subset 정의 -->
<!DOCTYPE 책목록 SYSTEM "c4_1401.dtd"
[
   <!ENTITY % en "IGNORE">
   <!ENTITY % kr "INCLUDE">
]>
<책목록>
   <책>
      <제목>시인과 도둑</제목>
      <저자>이문열</저자>
   </책>
</책목록>
```

DTD 사용 예: Book Markup Language

- 마크업 언어 개발 절차

  (1) 어떤 목적으로 마크업 언어를 개발할 것인가?

  (2) 전체 구조는 어떻게 구성할 것인가?

  (3) 엘리먼트들의 선택과 배치는 어떻게 할 것인가?

  (4) 속성 선택과 배치는 어떻게 할 것인가?

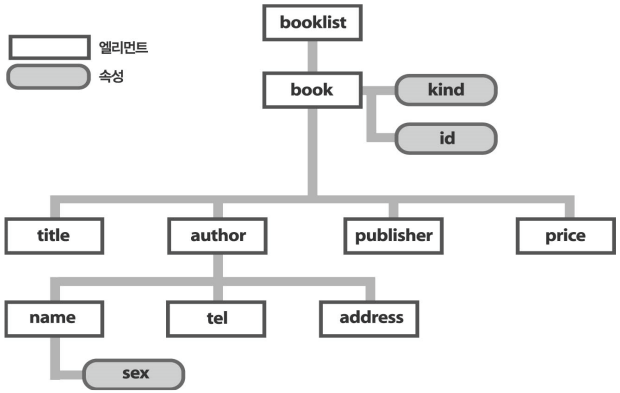
  (5) DTD 작성

  (6) XML 문서 작성 및 유효성 검사 실시

- BML 개발 목적
  - 서점이나 도서관에서 관리하고 있는 책에 대한 정보들을 표현하기 위한 마크업 언어 개발

DTD 사용 예: BML

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



DTD 사용 예: BML

- DTD 작성 (c4_1501.dtd)

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- 엘리먼트 선언 -->
<!ELEMENT booklist (book*)>
<!ELEMENT book (title, author+, publisher, price)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (name, tel, address)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT price (#PCDATA)>

<!-- 속성 선언 -->
<!ATTLIST book id ID #REQUIRED
                  kind (컴퓨터|소설|언어) #IMPLIED>
<!ATTLIST name sex (man | woman) #REQUIRED>
```

필요 X
닫는 태그 없다.

DTD 사용 예: BML

- 문서 작성 (c4_1501.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE booklist SYSTEM "c4_1501.dtd">   <!-- 문서 유형 선언 -->
<booklist>
   <book id="b1" kind="컴퓨터">
      <title>JSP And Servlet</title>
      <author>
         <name sex="man">신민철</name>
         <tel>010-123-4567</tel>
         <address>경기도 일산시</address>
      </author>
      <publisher>프리렉</publisher>
      <price>25000</price>
   </book>
   <book id="b2" kind="컴퓨터">
      <title>Inside XML</title>
      <author>
         <name sex="man">채규태</name>
         <tel>010-555-7777</tel>
         <address>서울시 중랑구 면목1동</address>
      </author>
      <publisher>디지털북스</publisher>
      <price>35000</price>
   </book> …
```