

XML

엄진영

<http://www.eomjinyoung.com>

저작권 및 배포안내

- ▶ 이 저작물은 참고 문헌을 토대로 “엄진영”이 재 가공하였습니다.
- ▶ 내용의 일부는 참고 문헌의 저작권자에게 있음을 명시합니다.
- ▶ 이 저작물의 인용이나 배포 시 이점을 명시하여 저작권 침해에 따른 불이익이 없도록 하기시 바랍니다.
- ▶ 위 사항을 명시하는 한 자유롭게 배포하실 수 있습니다.

참고문헌

- ▶ Beginning XML 2nd Edition
 - [David Hunter] – Wrox
- ▶ 기초에서 실무까지 XML
 - [신민철, 이규미, 채규태] – 프리렉
- ▶ 자바 개발자를 위한 XML
 - [최종명,유재우,최재영] – 흥릉과학출판사

차례

- ▶ XML Introduction
- ▶ XML 기술
- ▶ Well-formed XML
- ▶ DTD
- ▶ XPath
- ▶ XSL

XML Introduction

▶ 어원

XML = eXtensible + Markup Language♪

• Markup Language

- 문서의 내용에 추가적인 정보를 표시하기 위한 언어.
- 데이터를 제어하는 명령 데이터.

내용 -----→ 내용 + 명령♪

나는 왕이다. 나는 왕 이다.

결과: 나는 왕 이다.

▶ Markup Language

- 비공개

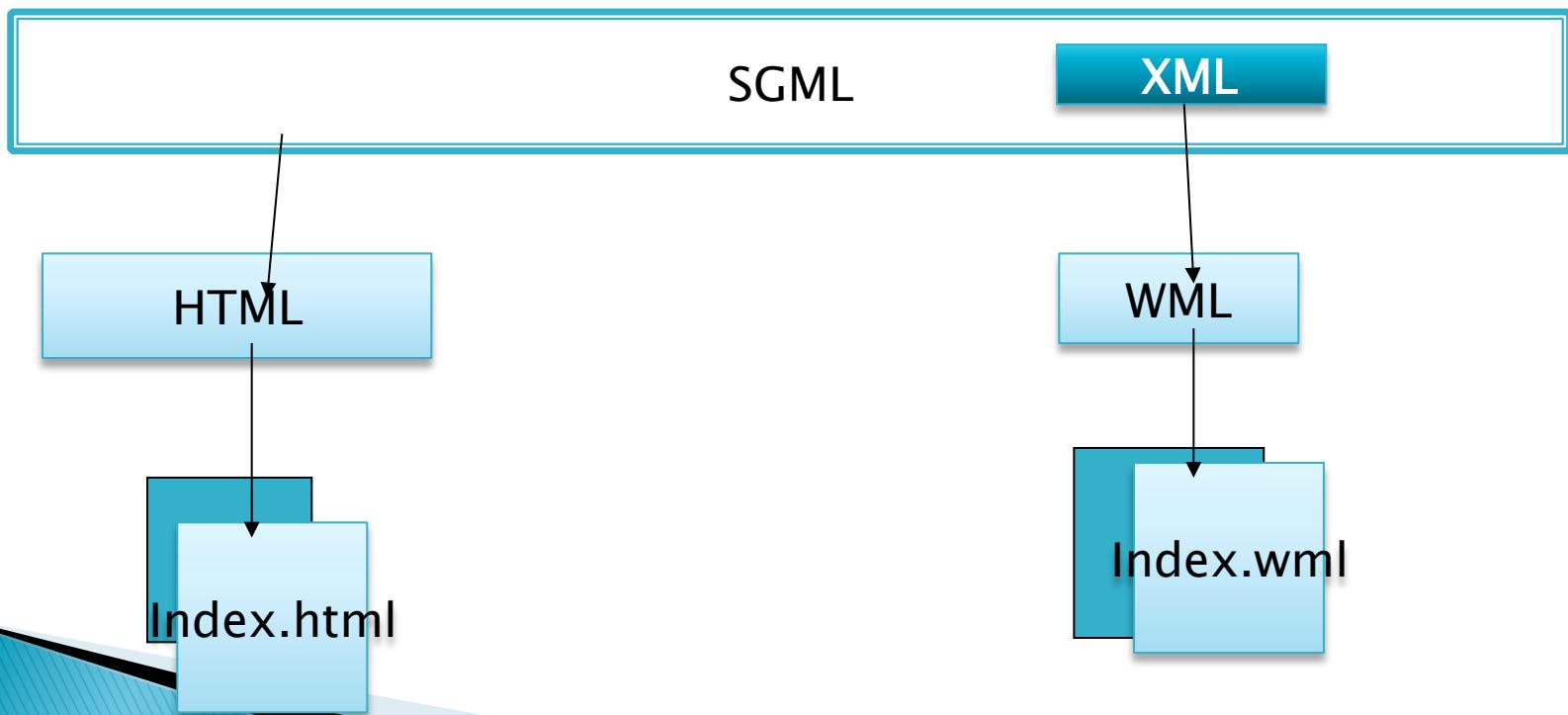
- 특정 Application에서만 해석될 수 있는 명령어
- 예) HWP, MS-Word, Word Perfect
- 단점: 서로 문서 호환이 안 된다.

- 공개

- Application에 상관없이 해석될 수 있다.
- 명령어가 공개되어 있다.
- 예) RTF(Rich Text Format), HTML 등
- 장점: Application 끼리 문서 호환이 된다.

▶ 메타 언어로서의 XML

- 메타언어란, 다른 Markup 언어를 정의하는 언어를 말 한다.



▶ 참고

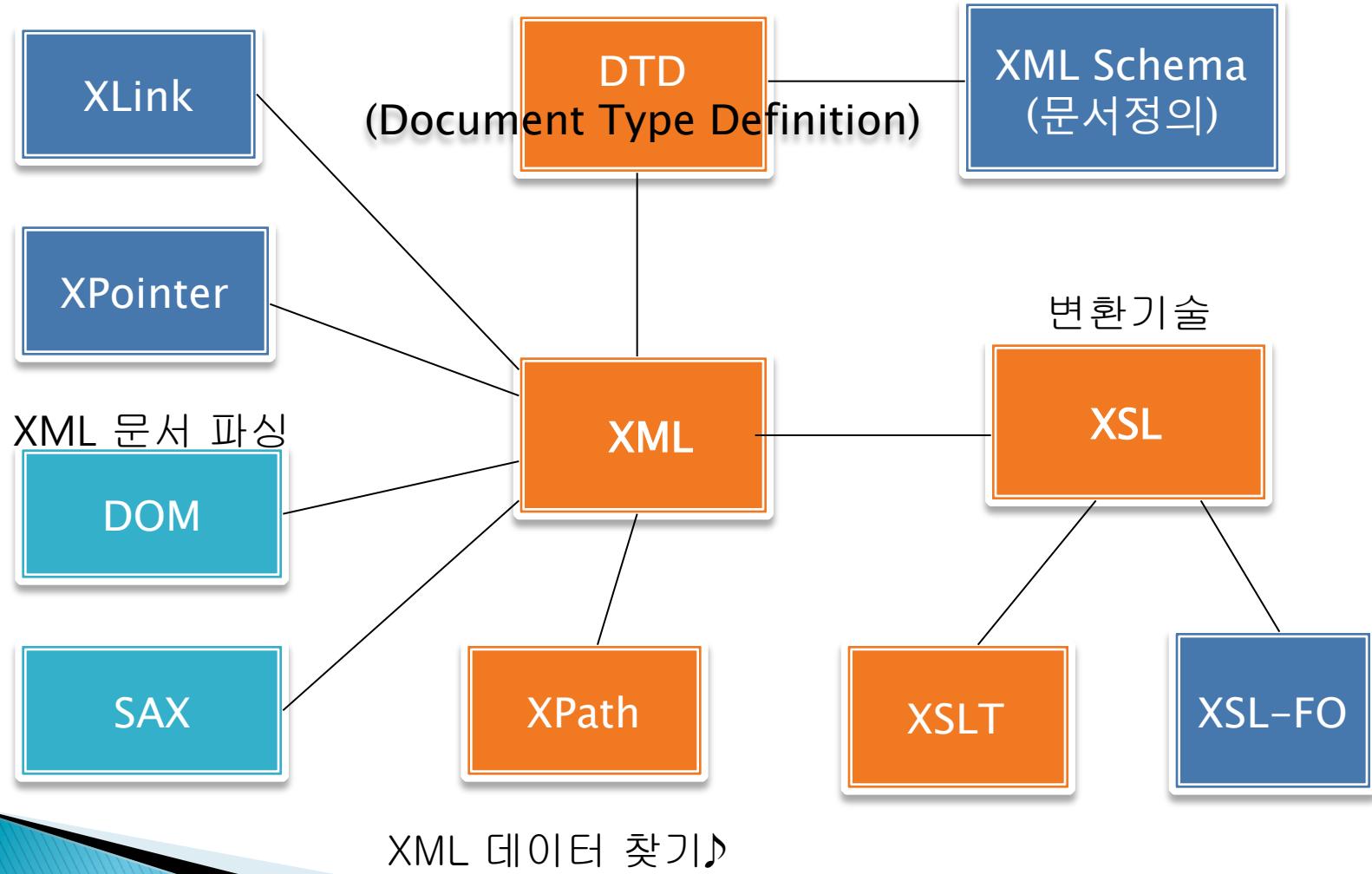
- WML(Wireless Markup Language)
 - 핸드폰이나 PDA와 같은 H/W 성능이 떨어지는 제품에서 웹 서핑을 목적으로 만든 간략한 Markup 언어이다.
 - Markup 언어가 간단해서 브라우저도 작은 사이즈로 만들 수 있다. 즉, 모바일 장치에 적합하다.
- 기타
 - CHTML(Compact HTML), sHTML(Simple HTML) 등이 있다.

XML 문서의 사용 용도

▶ 데이터 전달용 문서

HTML	XML
<p>표현 위주의 명령</p> <pre><h1>달마가 왜 갔을까?</h1> 홍길동 <i>오호라 출판사</i></pre>	<p>형식 명령</p> <pre><book> <caption> 달마가 왜 갔을까?</caption> <author> 홍길동 </author> <press> 오호라 출판사</press> </book></pre>
데이터 분석이 어렵다.	데이터 분석이 용이.

XML 기술



Well-formed XML

- ▶ XML 1.0 권고안에 언급되어 있는 문법에 따라 작성된 XML 문서
- ▶ 엘리먼트 작성 규칙
 - 시작태그와 끝 태그는 반드시 짹을 이뤄라.
 - ‘<’ 또는 ‘>’ 태그와 같이 XML에서 사용하기 위해 확보된 문자는 내용으로서 사용하지 말라. → 대안: XML Entity를 사용.
 - ‘<’ 와 ‘태그이름’ 사이에 공백이 와서는 안된다.
 - 태그이름에도 공백이 와서는 안된다.
 - ‘</’ 와 ‘태그이름’ 사이에 공백이 와서는 안된다.
 - 엘리먼트는 중첩될 수 없다.
 - 태그 명은 이름 짓는 규칙에 따라 지어라.
 - Root Element 는 하나만 와야한다.

▶ 이름짓는 규칙

- 첫문자 ‘_’, 문자등으로 시작. 숫자 나 ‘:’ 로 시작할 수 없다.
- 두번째 문자부터는 숫자 및 ‘_’, ‘-’, ‘.’ 도 가능.
- 태그이름에 공백을 넣을 수 없다.
- ‘:’ 문자는 예약어이므로 태그명으로 쓰지 말라.
- 태그명은 대.소문자 구분한다.
- 태그명은 대소문자 구분없이 “xml”로 시작할 수 없다.
→ 시작하도록 하지 마세요..권고.

XML 문서 구조

▶ 구조

Prolog(시작)

- XML 문서 선언문
- 문서 유형 선언문
- 프로세싱 지시자

Element(엘리먼트)

Misc(기타)

▶ XML 문서 선언문

<?xml version="1.0"?> ← 최소문법

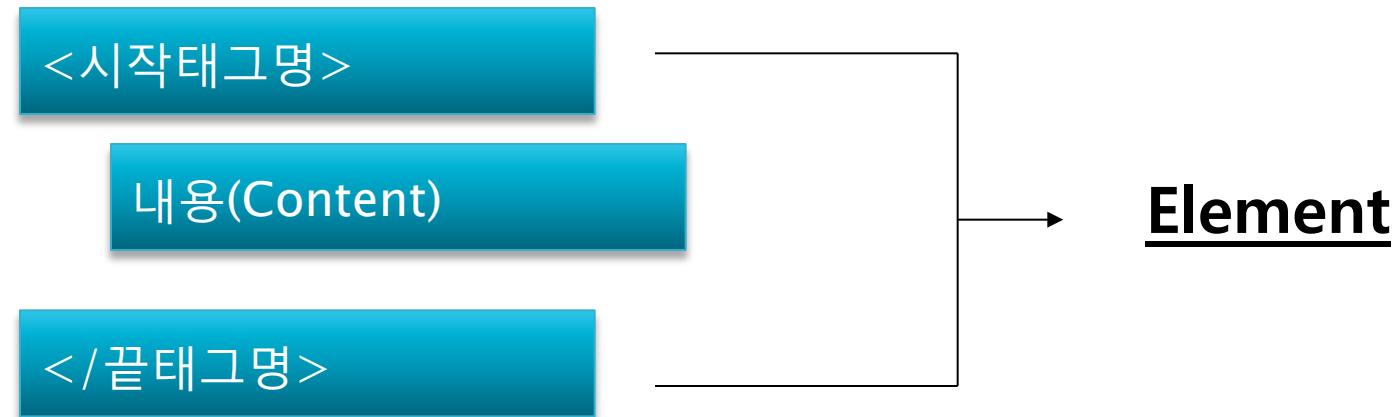
- 반드시 XML 문서의 첫 번째 라인에 와야 한다.
- 선언문 앞에 공백이나 주석이 와서도 안 된다.
- "<?" 와 "xml" 사이에 공백이 있어서도 안된다.
- ‘?’ 와 ‘>’ 사이에 공백이 있어서도 안된다.

▶ 선언문에 올 수 있는 속성

- **version**(필수) : 반드시 “1.0” 이 와야 한다.
- **encoding**(선택) : 기본은 “UTF-8”.
- **standalone**(선택): 기본은 “no”. “yes” 또는 “no” 값이 온다.
 - 문서의 의존성 여부를 표시.
- 속성은 반드시 순서대로 와야한다.

엘리먼트

▶ 문법



● 종류

- 내용을 가지는 엘리먼트
 - 예) <caption>UML과 패턴의 적용</caption>
- 내용이 없는 엘리먼트♪
 - 예) <male></male> → <male/> 표기 가능

- ▶ 엘리먼트 내용
 - 문자데이터(Character Data)
 - 자식 엘리먼트(Child Element)
 - 엔티티 또는 문자 참조
 - CDATA 섹션(Character Data Section)
 - 내용을 글자그대로의 문자로 취급
 - <![CDATA[내용...]]>
 - 프로세싱 지시자
 - 주석
 - 공백문자열
 - 스페이스(#x20), 탭(#x09), 캐리지리턴(#x0d), 라인피드(#x0a) 만을 공백문자로 분류.

속성

▶ 문법

- <시작태그명 속성명=“값” 속성명=‘값’>
내용...</끝태그명>

▶ 속성이름 짓는 규칙

- XML 엘리먼트 이름 짓는 규칙과 상동

▶ 용도

- 주 내용(Data)에 대한 보조적인 내용(Data)
- Element 보다 더 값에 대한 세밀한 제어가 가능 → 속성 타입에 따라 입력값을 제어가능.

주석

- ▶ XML 파서에 의해 무시되는 부분

- ▶ 문법

<!-- 주석 내용 -->

- 주의:

- 주석내용에 절대로 '--' 가 와서는 안된다.

DTD (Document Type Definition)

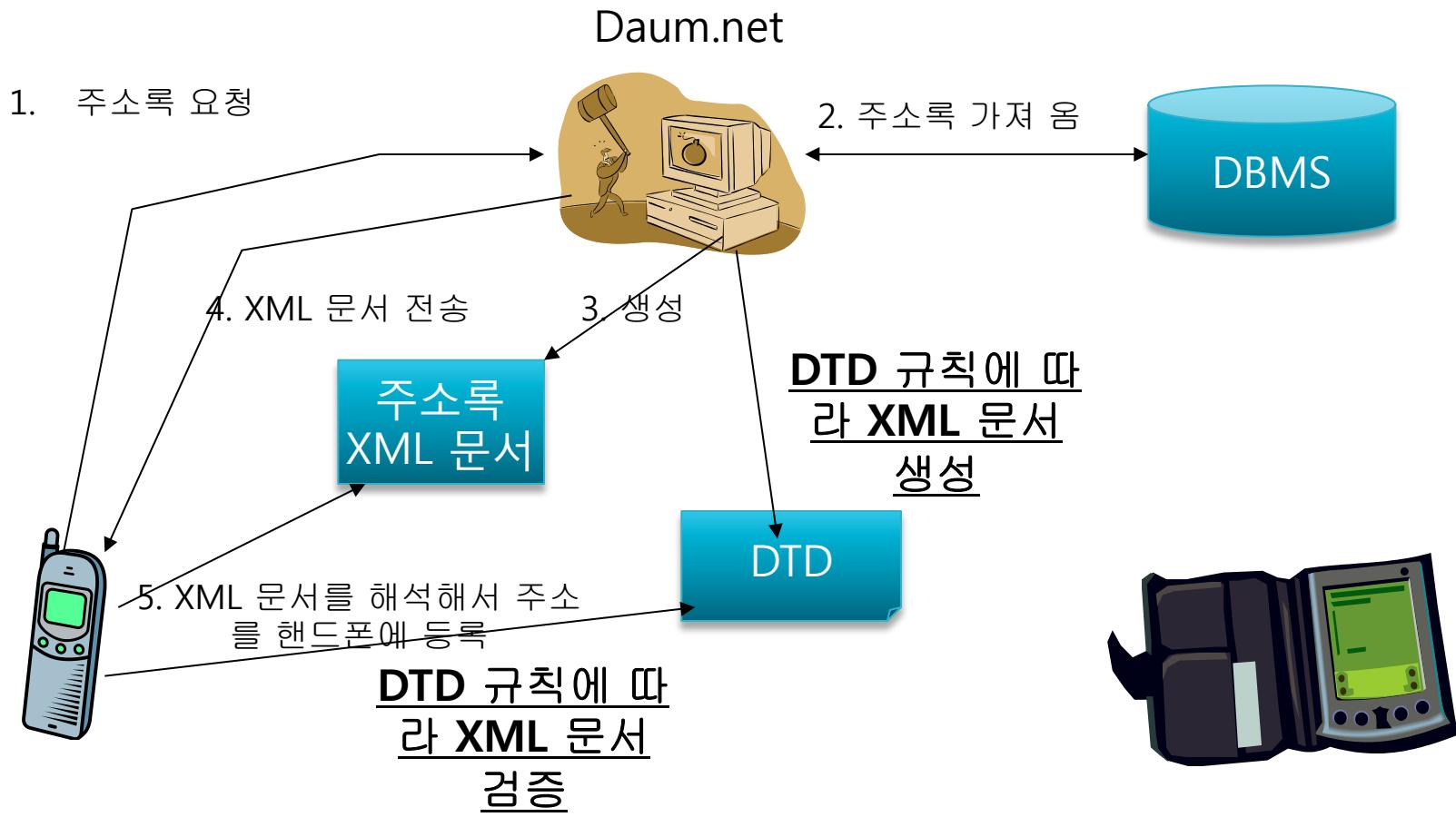
▶ XML 과 DTD

- XML에서 새로운 Markup 언어를 개발하기 위해서는 DTD를 이용하여 XML 문서의 구조를 정의해야 한다.

▶ 유효한 문서(Validate Document)

- Well-formed XML 문서이면서 DTD 규칙에 따라서 작성된 문서를 말함.

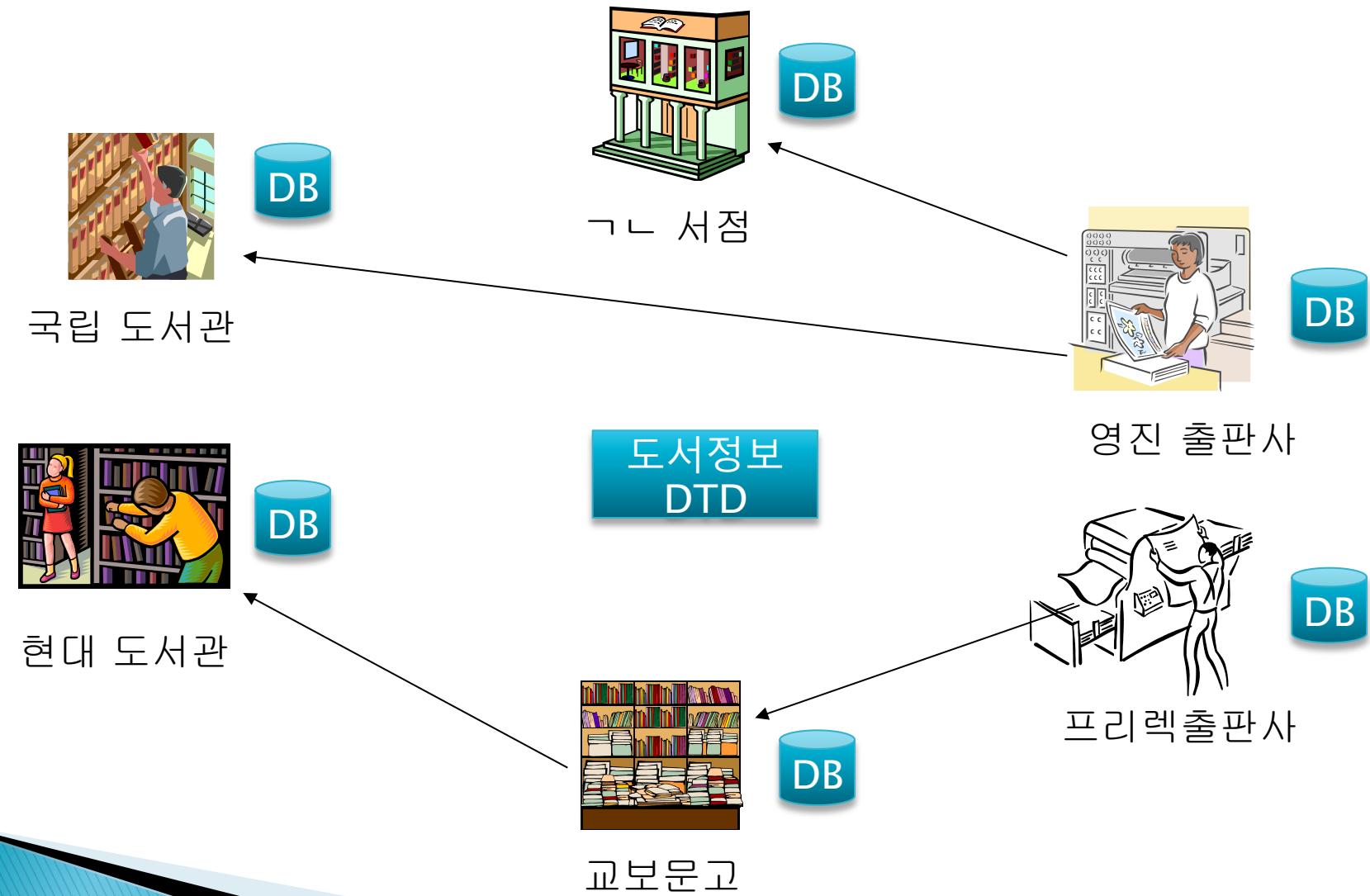
DTD 필요성



- XML 문서는 이기종(H/W 또는 S/W)간의 데이터 교환 시 유용.

- ▶ DTD는 XML 문서 작성자와 XML 문서 이용자 사이의 규칙을 명세한 것이다.
- ▶ DTD는 관련 사람들 및 그룹들이 협의체를 구성하여 XML 문서 교환의 규칙을 정의하게 된다.

도서관련 그룹들의 XML사용



DTD 물리적 구조

▶ 외부 DTD

- 별도의 DTD 파일을 작성하여 XML 문서에 연결
- 여러 XML 문서에서 공유할 수 있다.

▶ 내부 DTD

- XML 문서 내에 DTD를 정의
- 오로지 삽입된 XML 문서에 대해서만 적용된다.
- 주로 외부 DTD를 기반으로 특정 XML 문서에서 일부 규칙을 추가할 때 주로 사용.

외부 DTD

- ▶ XML 문서와는 다른 별도의 파일로 작성된 것을 말한다.
- ▶ 확장자를 관례상 '.dtd'를 붙인다.
- ▶ 일반 텍스트 파일 형식으로 작성된다.
- ▶ 외부 DTD 서브셋 구성요소
 - Text 선언
 - Element 선언
 - 속성(Attlist)선언
 - 엔티티(Entity)선언
 - 기호(Notation)선언
 - 프로세싱 인스트럭션(processsing instruction)
 - 파라미터 엔티티 참조
 - 주석
 - 공백
 - 컨디셔널 섹션(Conditional Section)

외부 DTD 서브셋 예

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT books (book*)>
<!ELEMENT book (caption,author,press)>
<!ELEMENT caption #PCDATA>
<!ELEMENT author #PCDATA>
<!ELEMENT press #PCDATA>
```

문서 유형 선언

- ▶ XML 문서 해석 시 어떤 Markup 언어로 작성되었는지 알려줄 목적으로 문서유형선언
- ▶ 외부 DTD 서브셋에 대한 문서유형선언
 - <!DOCTYPE 루트엘리먼트 SYSTEM “URI”>
 - DTD 파일의 URI 경로를 지정한다.
 - <!DOCTYPE 루트엘리먼트 PUBLIC “PUBLIC식별자” “SYSTEM 식별자”>
 - PUBLIC 식별자를 선언하고, 구체적인 DTD 파일 경로도 지정한다.

▶ PUBLIC 키워드를 사용한 문서유형선언

- 주로 공개적인 사용을 위해서 업체 및 국제 공인 단체에서 작성된 외부 DTD 서브셋을 지정할 경우 사용.
- PUBLIC 식별자
 - “+ 또는 -//dtd 개발및 유지보수 업체명//DTD명 및 버전번호//사용언어”
 - + : ISO처럼 국제적으로 공인된 단체
 - - : 국제적으로 공인되지 않은 단체
 - Dtd 개발 및 유지보수 업체명: dtd를 만들고 유지보수하는 단체이름
- SYSTEM 식별자
 - PUBLIC 식별자 다음에 선택적으로 SYSTEM 식별자가 올 수 있다.
 - 이것은 구체적으로 DTD문서의 URL을 지정함으로써 파일 위치를 알려주는데 보조한다.
- 예)
 - <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
“<http://www.w3c.org/dtd/html4.01.dtd>”>

내부 DTD

- ▶ XML 문서 내부에서 작성되는 것을 말함.
- ▶ 외부 DTD 서브셋의 일부 내용을 재정의 하여 현재 XML 문서에 적용하기 위해 사용
- ▶ 내부 DTD 서브셋의 구성요소
 - 외부 DTD 서브셋의 구성요소와 같다.

내부 DTD 서브셋 예

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE books [
<!ELEMENT books (book*)
<!ELEMENT book (caption, author, press)
<!ELEMENT caption (#PCDATA)
<!ELEMENT author (#PCDATA)
<!ELEMENT press (#PCDATA)
]>
<books>
  <book>
    <caption>아름다운 세상만들기</caption>
    <author>홍길동</author>
    <press>율도출판사</press>
  </book>
</books>
```

ELEMENT 선언

▶ 문법

- <!ELEMENT 이름 (Content type)>
- “<!” 다음에 오는 “ELEMENT”는 반드시 대문자
- 이름은 xml 이름짓는 규칙에 따라 작성
- 같은 이름의 엘리먼트가 중복하여 올 수 없다.

<시작태그명>

내용

</끝태그명>

- #PCDATA
- 자식 엘리먼트
- EMPTY
- MIXED
- ANY

ELEMENT CONTENT TYPE

▶ #PCDATA

- 문자 데이터만 갖는 엘리먼트
- Parsed character data: xml 파서가 해석할 수 있는 문자데이터가 와야한다.
- 예)

```
<!ELEMENT caption (#PCDATA)>
```

▶ EMPTY

- 내용을 갖지 않는 빈 엘리먼트를 지정할 때 사용.
- 문법

<!ELEMENT 엘리먼트명 EMPTY>

- 예)

<!ELEMENT 국내서 EMPTY>

<!ELEMENT 국외서 EMPTY>

ELEMENT CONTENT TYPE

▶ 자식엘리먼트

- 어떤 엘리먼트가 내용에 포함할 엘리먼트를 말함.
- 문법

<!ELEMENT 엘리먼트명 (자식엘리먼트명, ...) >

- 예

<!ELEMENT book (caption, author, press)>

- 반드시 자식 엘리먼트는 순서를 지켜야한다.

▶ 자식 엘리먼트 리스트 표현

- ‘,’ : 사용 순서 결정.

예) <!ELEMENT book (caption, author)>

- ‘|’ : 엘리먼트의 선택

예) <!ELEMENT book
(caption, (author|press))>

→ caption 다음에 author 또는 press가 선택적으로 올 수 있음.

▶ 엘리먼트의 사용빈도 지정

- ‘?’ : zero or one(선택사항)
- ‘+’ : one or more(1번이상)
- ‘*’ : zero or more(0번이상)
- 아무것도 붙지 않으면, 무조한 한번만 온다.

▶ MIXED

- 문자 데이터와 자식 엘리먼트가 혼합되어 내용에 올 수 있도록 한다.
- 문법
`<!ELEMENT 엘리먼트명 (#PCDATA | 자식엘리먼트명,...)*>`
 - 예)
`<!ELEMENT description (#PCDATA | 강조 | 이탤릭)*>`
 - 되도록 사용을 자제 → XML 문서의 근본인 데이터 구조가 무너지게 된다.

▶ ANY

- 내용에 문자 및 DTD내에 선언된 모든 엘리먼트들이 올 수 있게 한다.
- 문법
`<!ELEMENT 엘리먼트명 ANY>`
- 예
`<!ELEMENT description ANY>`
- 되도록 사용을 자제 → XML 문서의 근본인 데이터 구조가 무너지게 된다.

Attribute 선언

- ▶ Element에 부가적인 정보를 제공할 때 사용.
- ▶ 속성 대신 엘리먼트를 사용할 수도 있다.
- ▶ 엘리먼트보다 더 정밀한 제어가 가능.
 - 즉, 속성에 들어갈 값을 좀더 세밀하게 조정할 수 있다
-
- ▶ 문법

```
<!ATTLIST 엘리먼트명
      속성명 속성타입 디폴트선언
      속성명 속성타입 디폴트선언
      ...
    >
```

속성 선언 예

▶ DTD 예

```
<!ELEMENT book (caption, author, press)>
```

```
<!ATTLIST book
```

```
    인쇄일 CDATA #REQUIRED
```

```
    발행일 CDATA #REQUIRED
```

```
>
```

▶ XML 예

```
<book 인쇄일="2004-2-2" 발행일="2004-3-3">
```

```
    <caption>세상을 아름답게 만들기</caption>
```

```
    <author>홍길동</author>
```

```
    <press>율도 출판사</press></book>
```

▶ 디폴트선언의 종류

- #IMPLIED

- 속성을 생략가능.

- #REQUIRED

- 반드시 속성지정.

- #FIXED

- 속성값으로 지정한 값 이외에는 넣을 수 없다.

- 디폴트 값

- 생략되었을 경우, 기본 값으로 사용할 값 지정.

▶ 속성 타입의 종류

- CDATA : 문자 데이터
- ENUMERATION : 열거된 속성값
- ID : XML 문서 내에서 유일한 식별자 값
- IDREF/IDREFS : XML 문서내에서의 ID 값
- NMTOKEN/NMTOKENS : 이름작성 규칙을 준수하는 문자 데이터
- NOTATION : NOTATION으로 선언된 이름
- ENTITY : 엔티티로 선언된 이름

▶ CDATA

- #PCDATA 처럼 XML 파서에 의해 해석될 수 있는 문자 데이터.

- DTD

```
<!ATTLIST book  
    부제 CDATA #IMPLIED>
```

- XML

```
<book 부제="3 < 2 인 이유?">
```

....

```
</book>
```

▶ ENUMERATION

- 속성에 들어갈 수 있는 값을 열거할 때 사용.
- DTD

<!ATTLIST book

분류 (컴퓨터 | 인문 | 과학 | 소설) “컴퓨터”>

- XML

<book 분류=“인문”>

....

</book>

▶ ID

- 엘리먼트를 구별하기 위한 식별자 값을 넣을 때 사용.
- 단, 식별자 값은 XML 문서 내에서 언급된 식별자들 중에서 유일한 식별자 값이어야 한다.
- 엘리먼트에 상관없이 모든 엘리먼트의 식별자 값을 중에서 유일 해야한다.
- 숫자로 시작할 수 없다. 중간에 공백이 와서도 안된다.

- DTD

```
<!ATTLIST book  
        no ID #REQUIRED>
```

- XML

```
<book no="a01"> ... </book>
```

▶ IDREF

- XML 문서 내에서 존재하는 식별자 값을 지정할 때 사용

.

- DTD

```
<!ATTLIST book
```

```
    author IDREF #REQUIRED>
```

- XML

```
<book author="a001">...</book>
```

▶ IDREFS

- 여러 개의 식별자 값을 넣을 때 사용
- 각각의 식별자 값은 공백으로 구분.
- DTD

<!ATTLIST book

author IDREFS #REQUIRED>

- XML

<book author="a001 a003 a002">...</book>

▶ NMOKEN

- XML 이름짓는 규칙에 따라서 작성된 값만을 속성값으로 가질 수 있게한다.
- DTD

<!ATTLIST author

 이름 NMOKEN #REQUIRED>

- XML

<author 아이디=“a001” 이름=“홍길동”>...</author>

▶ NMTOKENS

- 여러 개의 NMTOKEN 값은 지정할 때 사용.
- 각 NMTOKEN 값은 공백으로 분리한다.
- DTD

<!ATTLIST author

 연락처 NMTOKENS #IMPLIED>

- XML

<author 연락처=“222-2222 333-3333”>

...

</author>

ENTITY 선언

- ▶ XML 문서를 구성하는 물리적인 저장단위.
- ▶ 상수값을 저장할 때 많이 사용
- ▶ 동일한 데이터를 여러 XML 문서에서 사용할 때
별도의 ENTITY로 정의하여 참조하게 할 수 있다.

ENTITY 분류

- ▶ 물리적인 저장단위 존재 여부
 - 내부 엔티티
 - DTD 문서 내에 정의되어 있다.
 - 외부 엔티티
 - 별도의 파일 형태로 정의되어 있다.
- ▶ 사용되는 곳에 따른 분류
 - 일반 엔티티
 - XML 문서 내에서 참조하여 사용
 - 파라미터 엔티티
 - DTD 문서 내에서 참조하여 사용

- ▶ 문자데이터로 이루어졌는지 여부
 - 파스드(parsed)
 - XML 파서가 해석할 수 있는 문자 데이터로 구성됨.
 - 언파스드(unparsed)
 - XML 파서가 해석할 수 없는 바이너리 데이터로 구성됨. → 그림, 음성, 기타 바이너리 데이터.

▶ Built-in ENTITY

- XML에서 미리 정의된 엔티티

<	<	Less-than
>	>	Greater-than
&	&	Ampersand
"	"	Quotation
'	'	apostrophe

▶ 내부 일반 파스드 엔티티

- DTD문서내에 선언되어 있고 XML문서에서 참조하며, XML 파서가 해석할 수 있는 문자로 이루어진 데이터.

- 선언

DTD 문서 내:

```
<!ENTITY 엔티티명 "값">
```

- 사용

XML 문서 내에서 참조:

```
&엔티티명;
```

▶ 외부 일반 파스드 엔티티

- 별도의 파일에 엔티티 데이터가 저장되어 있고, XML 문서에서 참조하며, XML 파서가 해석할 수 있는 문자로 이루어진 데이터.
- 장점: 여러 XML 문서 및 DTD에서 entity 데이터를 공유할 수 있다.
- 선언
 - DTD 문서 내:
 - <!ENTITY 엔티티명 SYSTEM “URI”>
- 사용
 - XML 문서 내에서 참조:
 - &엔티티명;

▶ 내부 파라미터 엔티티

- DTD 문서 내에 선언되었으며, DTD 문서 내에서만 참조되는 엔티티를 말함.
- 자주 사용되는 속성들이나 자식 엘리먼트들을 정의함으로써 DTD 문서에서 참조할 수 있다.
- 선언

DTD 문서 내에서 선언:

```
<!ENTITY % 엔티티명 "데이터">
```

- 사용

DTD 문서 내에서 사용:

```
%엔티티명;
```

XPath

- ▶ XML 문서에서 특정 엘리먼트나 속성에 접근하기 위해 경로를 지정하는 언어.
- ▶ XSLT 및 Xpointer 언어에 사용될 목적으로 설계됨.
- ▶ XSL 언어의 부분집합이다.

Xpath 와 Node

- ▶ XML 문서의 트리 구조를 구성하는 구성원을 “노드”라 한다.
- ▶ 노드 종류
 - 루트 노드 : XML 문서 자체를 표현하는 최상위 노드.
 - 엘리먼트 노드 : 루트엘리먼트도 여기에 속한다.
 - 속성 노드
 - 텍스트 노드 : 시작태그와 끝태그 사이의 내용표현.
 - 네임스페이스 노드
 - 프로세싱 지시자 노드
 - 주석노드

Location Path

- ▶ ‘/’로 위치 경로를 표현한다.
- ▶ 절대 경로 와 상대 경로로 나뉘어진다.
- ▶ 절대경로
/루트엘리먼트/자식엘리먼트/.../자식엘리먼트
- ▶ 상대경로
자식엘리먼트/.../자식엘리먼트
. /자식엘리먼트/.../자식엘리먼트 (현재기준)
.. /형제엘리먼트 (부모엘리먼트기준)

Location Step

▶ 문법

Axis::NodeTest[Predicate]

- Axis : 노드를 찾기 위한 검색 방향
- NodeTest : 찾을 노드의 이름
- Predicate : 필터링을 하기 위한 표현식

Axis

▶ Ancestor

- 직계 조상(아버지->할아버지->증조->고조)
- 작은아버지,큰아버지,작은할아버지,큰할아버지 등은 제 외. 오로지 직계조상만을 의미.

▶ Child

- 현재노드의 직계 자식노드를 가르킴.

▶ Descendant

- 현재 노드의 직계 자손노드를 가르킴.

▶ Following

- 현재노드 다음에 오는 노드들을 가르킴.

▶ Preceding

- 현재노드 이전에 오는 노드들을 가르킴
- 현재노드의 ancestor는 제외.

▶ Parent

- 현재노드의 부모노드.

Xpath 조건주기

▶ Predicate

- [엘리먼트명='내용값']
- [@속성명='속성값']

XML Namespace

- ▶ 보통 하나의 XML 문서는 하나의 Markup 언어를 사용하게 되어 있다.
- ▶ 그러나, 때론 하나의 문서에 여러사람들에 의해 정의된 Markup언어를 사용할 수도 있는데. 이때, 내가 만든 Markup 과 다른 이가 만든 Markup을 구분하기 위해 이름들의 영역을 정의하게 된다.
- ▶ 이것이 Namespace다.

이름 충돌 방지

- ▶ 접두사를 사용하여 이름충돌을 방지한다.
접두사:엘리먼트명
- ▶ 접두사의 이름 충돌가능성을 배제하는 방법
 - 접두사에 대해 정확히 URL로 정의한 후 사용.
 - 문법
 - xmlns:접두사명="URL"
 - 이때 이 URL이 Namespace 이름이다.

XSL

- ▶ eXtensible Stylesheet Language 의 약자
- ▶ XML 문서를 다양하게 표현하기 위한 스타일시트 언어.
- ▶ 즉, XML 문서를 다양한 다른 형태의 문서로 변환하기 위한 언어.

XSL의 구성

▶ XSLT(XSL Transformation)

- XML 문서의 구조를 다른 구조로 변환시키기 위해 설계된 언어.

▶ XPath (XML Path Language)

- XML 문서내의 특정 엘리먼트나 속성을 찾아가기 위해 사용되는 경로 표기 언어.

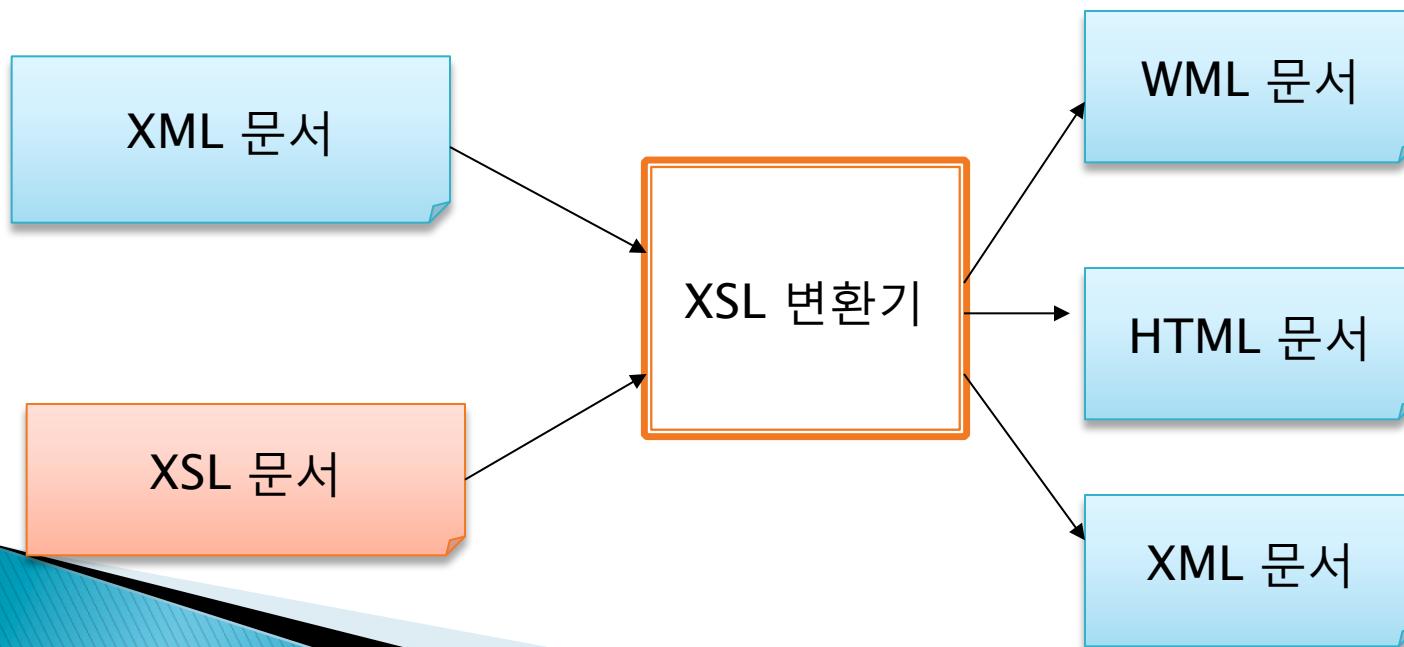
▶ XSL-FO(XSL Formatting Objects)

- XML 문서를 비 XML 문서(예: PDF)로 변환하기 위해 설계된 언어.

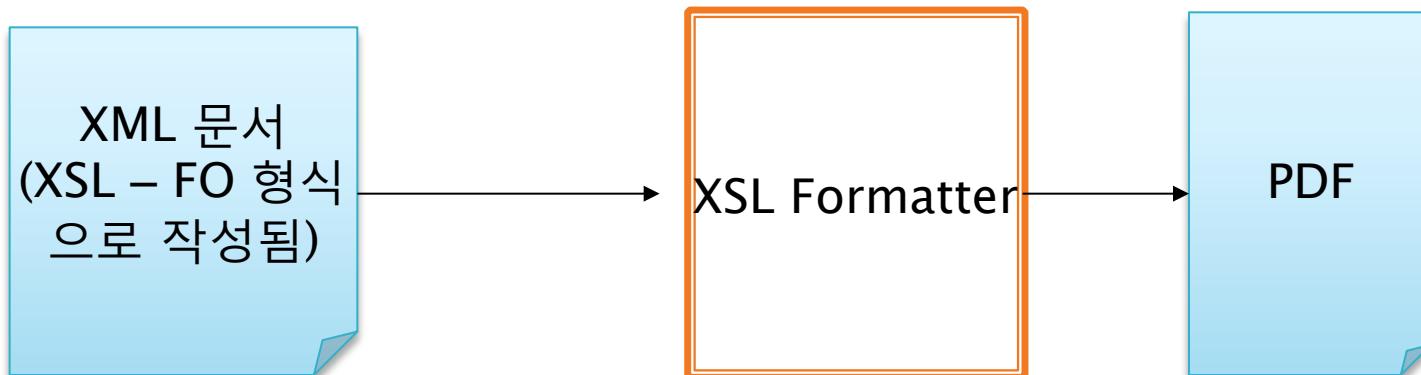
XSL 처리 과정

▶ Transformation

- XML 문서를 다른 구조의 XML 문서로 변환.



- ▶ **Formatting**
 - XML 문서를 특정한 S/W, H/W에 맞게
서로 변환.



XSL 문서를 XML 문서에 적용

▶ 문법

```
<?xml-stylesheet type="text/xsl"  
    href="XSL 문서 URL"?>
```

XSL 문서 작성

▶ XSL 문서 루트 엘리먼트

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3c.org/1999/XSL/Transform">

</xsl:stylesheet>
```

▶ output 엘리먼트

- XML 변환기를 통해 나온 결과물을 “결과 트리 문서” 라 한다.
- 이 “결과 트리 문서”的 종류를 지정한다.
- 속성
 - method : 결과 문서의 종류(예: xml, html, text)
 - version : 결과 문서가 XML 문서일 경우 XML 권고안 버전 (예: 1.0)
 - encoding : 결과 문서의 인코딩 방식(예: UTF-8, EUC-KR)
 - omit-xml-declaration : XML 선언 생략 여부 (예: yes, no)
 - standalone : 외부문서 참조 여부 (예: yes, no)
 - doctype-public : DTD 지정 선언문
 - doctype-system : DTD 지정 선언문
 - cdata-section-elements : CDATA 섹션으로 사용되어야할 노드
 - indent : 결과 문서에 오와 열을 맞출것인가(예: yes, no)
 - media-type: 미디어 타입지정

▶ Template Rule

- 어떤 구조를 다른 구조로 바꾸는 방법.
- 보통 XSL 문서는 여러 개의 template 룰이 존재하며 각각의 Rule은 원본 XML 문서를 다른 구조의 노드들로 변화하는 역할을 한다.
- 최상위 노드부터 변환 적용을 받는다.
- 문법

```
<xsl:template 속성="값" ..>  
    변환될 내용  
</xsl:template>
```

▶ Template 룰의 속성

- match : 변환을 적용할 원본 XML 문서의 노드를 가르킴
- priority : 같은 레벨의 템플릿이 있을 경우 priority 속성 값이 높은 템플릿이 우선한다.
- name : 자주 쓰이는 변환 명령어를 별도의 템플릿으로 지정했다가 사용할 때 name 속성을 사용한다.

▶ Template rule의 적용

- 선택된 노드에 대해 어떤 템플릿을 적용할 지 지정한다.
- 문법

```
<xsl:apply-template  
    select="적용할 원본 XML문서 노드"/>
```

▶ 원본 XML 문서 데이터 얻기

- 현재 노드를 기점으로 XPath에 지정된 노드의 값을 가져온다.
- 노드는 엘리먼트 노드이거나 속성 노드이다.
- <xsl:value-of match="노드 XPath"/>