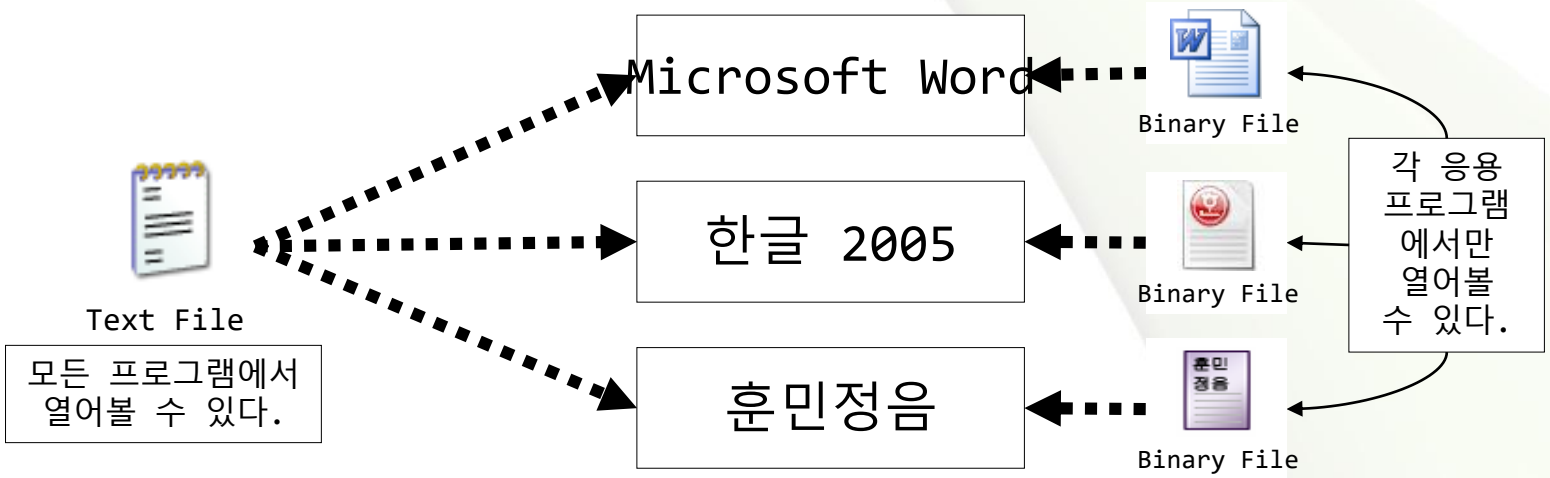
01. Introduction to XML

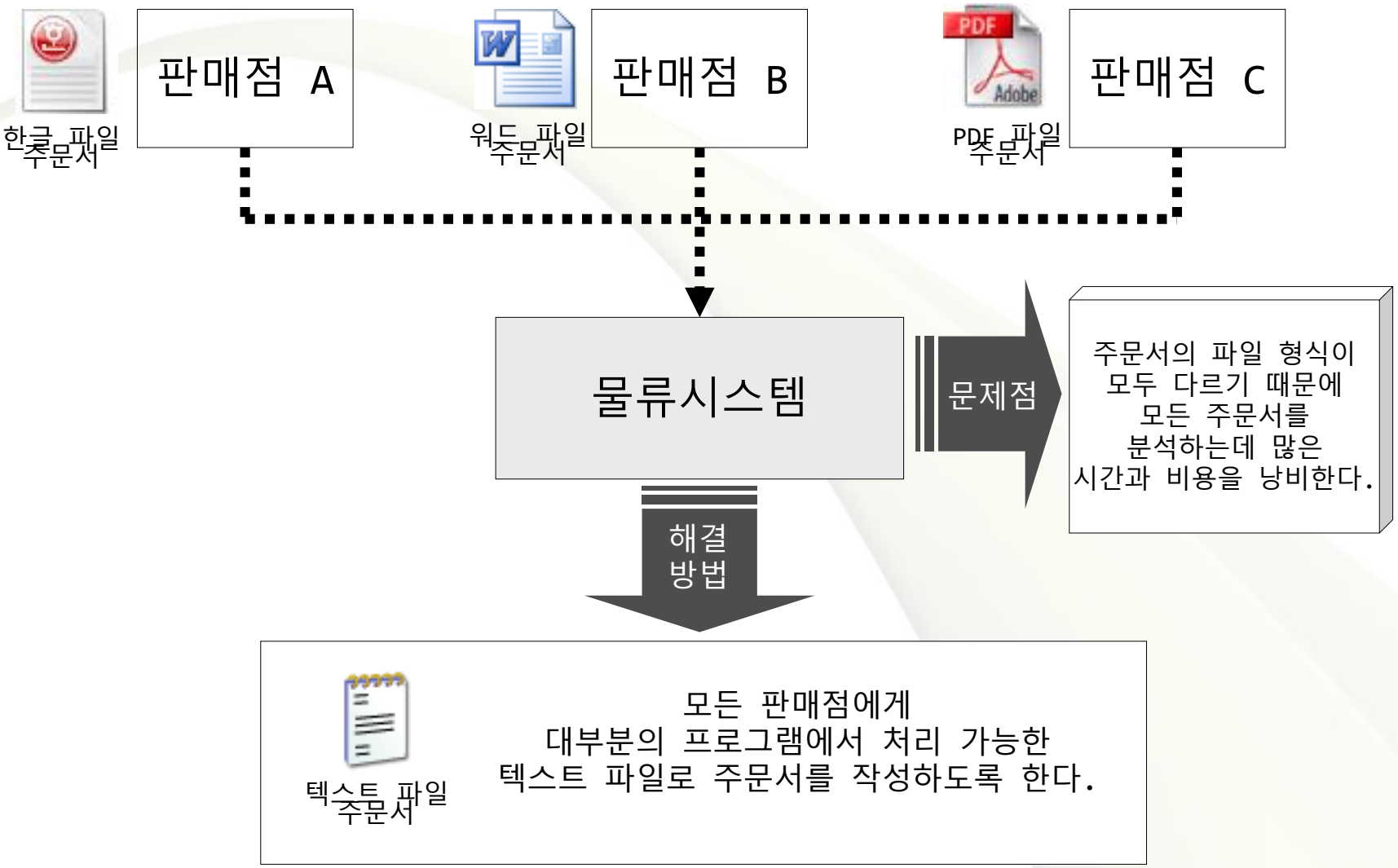
데이터 포맷

- 바이너리 데이터
  - 메타 데이터(Meta Data)를 포함하면서 0과 1의 단순한 나열로 이루어진 파일
  - 해당 파일을 읽을 수 있는 특정 애플리케이션에서만 실행 가능

- 텍스트 데이터
  - 표준화된 문자 표현 방법에 의해 저장된 파일
  - 모든 응용 프로그램에서 해석 가능



바이너리 데이터 포맷의 문제



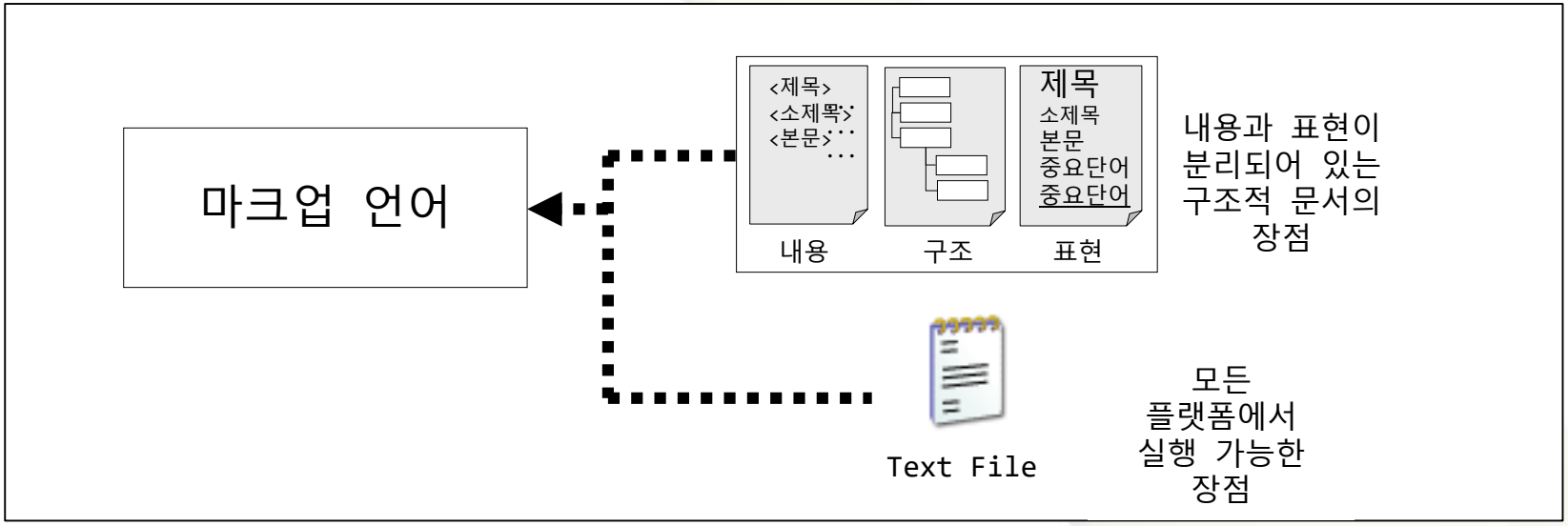
구조화된 문서

- 비구조적 문서
  - 작성자의 스타일에 따라 작성된 문서
  - 내용과 구조, 그리고 표현이 복합되어진 형태

- 구조적 문서
  - 내용과 구조, 표현의 관점에서 문서가 분리되어져 있는 형태
  - 장점
    - 입력, 편집, 출판 등 각 작업의 시간적, 공간적 분리
    - 유통, 배포의 용이성
    - 출판의 다양성
    - 정확한 정보 검색

마크업

- 텍스트데이터의 구조적 문서 표현 방법
  - 텍스트기반 메타데이터 표현
  - 문서의 논리적인 구조를 표현



SGML(Standard Generalized Markup Language)

- SGML이란?
  - 디지털 문서를 플랫폼에 상관없이 사용하기 위한 목적으로 만들어진 언어
  - 1986년에 ISO-8897 표준으로 제정
  - 문서의 논리적 구조와 내용을 기술하기 위한 마크업 언어

- SGML의 장점
  - 시스템과 플랫폼 독립적
  - 재사용성
  - 공개 표준

- SGML의 문제점
  - 작성 규칙의 복잡성으로 인해 전문가들만을 위한 언어
  - 관련 업계의 범용적인 지원 미비

HTML(Hyper Text Markup Language)

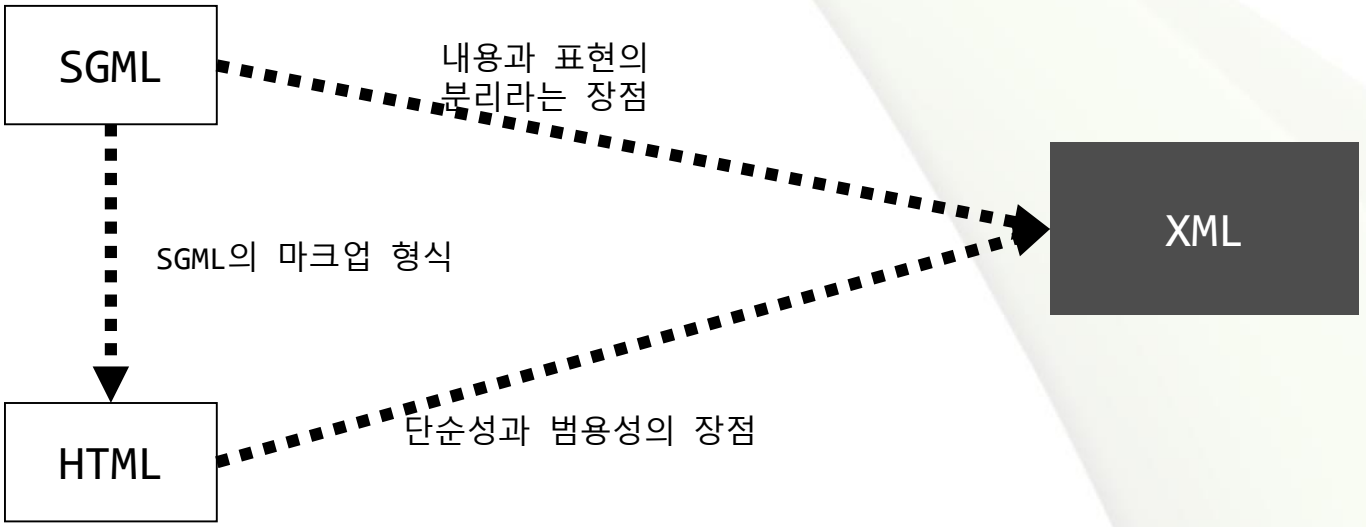
- HTML이란?
  - 웹에서 문서를 표현하기 위한 마크업 언어
  - SGML의 응용 언어

- HTML의 장점
  - 구현이 쉽고 이로 인해 생산성이 높습니다.

- HTML의 한계
  - 고정된 태그 집합
  - 디스플레이 기능에 치중 - 다양한 비즈니스 데이터 표현 불가능

XML(Extensible Markup Language)

- 인터넷 환경에서 구조화된 문서를 전송 가능하도록 설계된 표준 마크업 언어
- SGML의 서브셋(subset) 형태로 SGML과 HTML의 장점을 수용하여 만들어진 언어
- 1998년 2월 XML 1.0이 W3C에 의해 표준안으로 확정

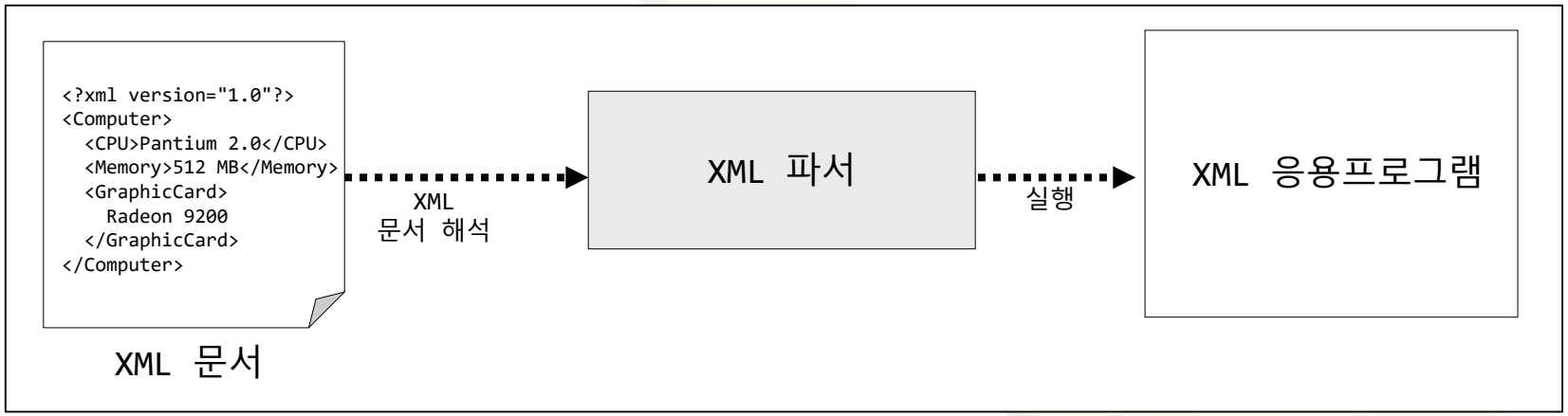


XML의 장점

- 호환성
  - 표준 규약을 따르기 때문에 작성 규칙만 지키면 어디에서든지 사용 가능
- 독립성
  - 하드웨어, 운영체제, 프로그래밍 언어에 독립적
- 확장성
  - XML 문서의 작성자가 새로운 태그를 정의해서 사용할 수 있기 때문에 다양한 형태의 XML 문서 작성이 가능
- 다양한 포맷으로 변경 가능
  - 내용과 표현이 분리된 구조화된 문서로써 여러 가지 표현이 가능
- 검색 능력의 향상
  - XML 태그는 논리적인 구조와 내용을 기술하기 때문에 검색 능력 향상

XML 문서 처리

- XML 파서(Parser)
  - XML 문서를 해석한 후 XML 응용프로그램에 인터페이스를 제공



- XML 파서의 종류
  - Microsoft
    - MSXML - 인터넷 익스플로러에 내장된 XML 파서
  - Java
    - SUN 사의 JAXP
    - IBM 사의 XML4J

Well-Formed XML과 Valid XML

- Well-Formed XML
  - XML 1.0 스펙에서 정의된 문서 규칙을 따르는 XML 문서를 Well-Formed XML이라 한다.

- Valid XML
  - DTD나 XML Schema에 정의된 XML 구조를 만족하는 Well-Formed XML 문서를 Valid XML이라 한다.

02. Well-Formed XML

XML 선언문(XML Declaration)

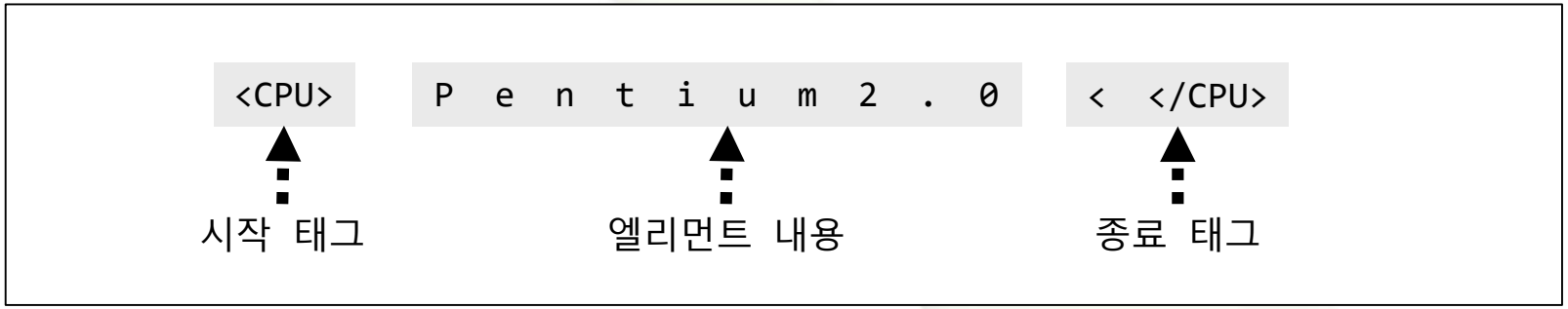
- XML 선언문의 작성
  - XML 선언문의 사용은 선택적이지만, 사용할 경우 XML 선언문은 반드시 XML 문서의 제일 처음에 와야 한다.

<?xml	version="1.0"	encoding="EUC-KR"	standalone="yes"?>
	필수 XML 버전 명시	옵션 XML 문서의 인코딩 지정	옵션 외부 파일 참조 여부 지정 디폴트 값은 no

  - XML 선언문의 세 가지 속성
    - version
      - XML 문서의 권고안 버전을 기술하며, 그 값은 항상 1.0 이다.
    - encoding
      - XML 문서가 어떤 문자 인코딩 방식에 따라 해석될 지를 지정한다.
    - standalone
      - 외부 파일의 참조 없이 독립적으로 실행 가능한지를 지정한다.
      - standalone 속성을 사용하지 않을 경우 기본값은 no이다.

엘리먼트(Element)

- 엘리먼트(Element)란?
  - XML 문서를 이루는 기본 단위



- 작성 규칙
  - 시작 태그와 종료 태그, 그리고 엘리먼트의 내용으로 구성된다.
  - 태그 이름은 대소문자를 구분한다.
  - 종료 태그가 없는 빈 태그(Empty Tag)는 뒤에 "/"를 붙인다.
  - 올바른 중첩 구조를 가져야 한다.

루트 엘리먼트 (Root Element)

- 루트 엘리먼트란?
  - XML 문서의 최상위 엘리먼트를 의미한다.
  - 루트 엘리먼트는 XML 문서에서 반드시 하나만이 존재해야한다.

```
<?xml version="1.0" encoding=“utf-8"?>
<Book>
   <Title>XML</Title>
   <Author>홍길동</Author>
</Book>
```

어트리뷰트 (Attribute)

- 어트리뷰트란?
  - XML 문서에서 엘리먼트를 꾸며주는 형용사와 같은 역할을 합니다.

- 작성 규칙
  - 어트리뷰트는 이름과 값의 쌍으로 이루어집니다.
  - 어트리뷰트의 값은 큰 따옴표나 작은 따옴표로 둘러 싸여져 있어야 합니다.
  - 순서는 강제사항이 아니지만 중복된 어트리뷰트를 사용할 수 없습니다.

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

<Book>
        <Title number="3">XML</Title>
        <Title number='3'>XML</Title>
</Book>
```

엘리먼트와 어트리뷰트의 이름 규칙

- 엘리먼트와 어트리뷰트의 이름
  - 엘리먼트와 어트리뷰트의 이름은 올바른 이름(qName, Qualified Name) 규칙에 따라 작성되어야 한다.

- 이름 규칙
  - 이름은 반드시 문자로 시작해야 하며, 숫자로는 시작할 수 없다.
  - 이름 내부에 "_" 나 "." 같은 일부 기호를 사용할 수 있다.
  - 공백 문자를 포함할 수 없다.
  - 대소문자 구분 없이 XML(xml)이란 단어로 시작할 수 없다.
  - "<" 뒤에 공백 문자를 포함할 수 없다.
  - ":" 기호는 네임스페이스에 의해 예약된 문자이므로 사용하지 않는 것을 권장한다.

주석문 (Comments)

- 주석문(Comments)
  - XML 문서 내부에 사용자를 위해 삽입된 추가 설명 구문
  - XML 구문규칙이 적용되지 않는 영역

- 주석문의 작성
  - <!-- 와 --> 기호 사이에 추가 설명을 삽입

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

<!-- 서적 정보를 나타내는 XML 문서 -->
<Book>
        <Title>XML과 XML Web Service</Title>
        <!-- 여기에도 주석문을 쓸 수 있다. -->
        <Author>이명진</Author>
</Book>
```

- 주석문 작성시 유의사항
  - 주석문의 내부에는 주석문의 시작과 끝을 의미하는 "--" 기호를 삽입할 수 없다.
  - 주석은 XML 문서의 어디든 사용할 수 있지만, XML 선언문 앞에는 쓸 수 없다.

특수문자 표기

- PCDATA(Parsed Character Data)
  - PCDATA란 XML 문서에서 엘리먼트나 어트리뷰트의 값으로 사용될 수 있는 문자들을 말한다.

- XML 문서에 사용할 수 없는 문자들
  - XML 문법에 사용되기 때문에 XML 문서에서 사용할 수 없는 예약된 문자들이 정의되어 있습니다.
  - 특수문자를 표시하기 위해 엔터티 레퍼런스 또는 문자 레퍼런스를 사용
  - 종류

문자	엔티티 레퍼런스
<	<
>	>
‘	'
“	"
&	&

CDATA 섹션(CDATA Section)

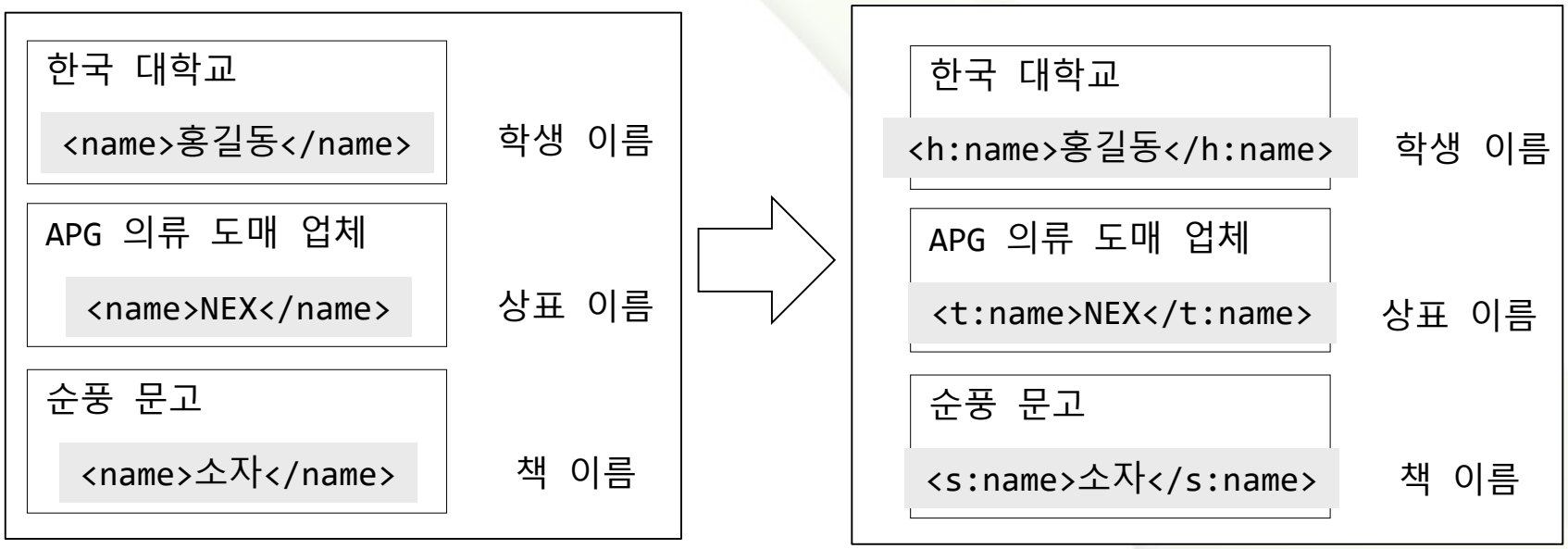
- CDATA 섹션
  - 특수문자 표기가 많아서 효과적인 데이터 표현이 어려운 경우에 사용
  - 파서는 CDATA 섹션으로 선언된 구문을 해석하지 않는다.
  - "<![CDATA["로 시작하며, "]]>"로 끝난다.

```
<?xml version=“1.0” ?>

<person>
        <id>아이디</id>
        <name>이름</name>
        <resume>
                <![CDATA[
                        <html>
                                <body>…</body>
                        </html>
                ]]>
        </resume>
</person>
```

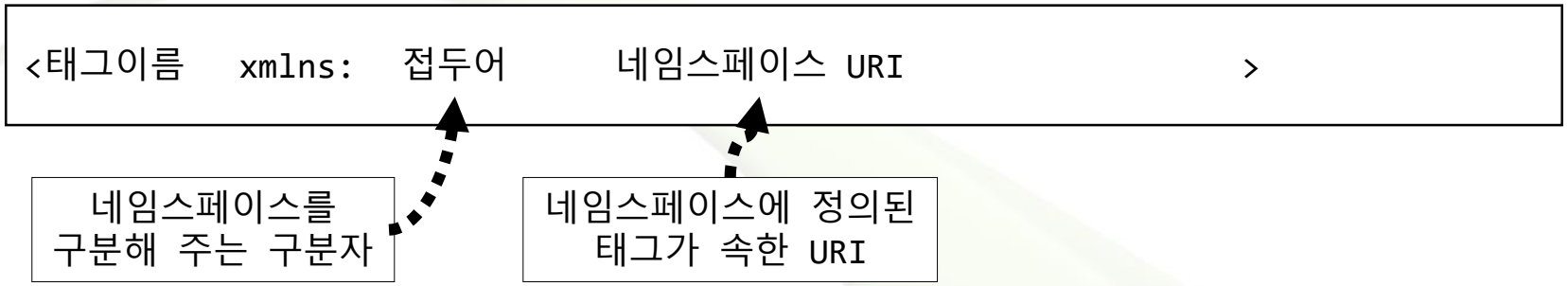
XML 네임스페이스 (Namespace)

- 태그의 이름 충돌 문제를 해결하는 방법
- XML 문서의 엘리먼트와 어트리뷰트에 고유한 접두사를 추가해 충돌을 방지하고 추상적인 의미 부여



네임스페이스의 선언과 적용

- 네임스페이스의 선언

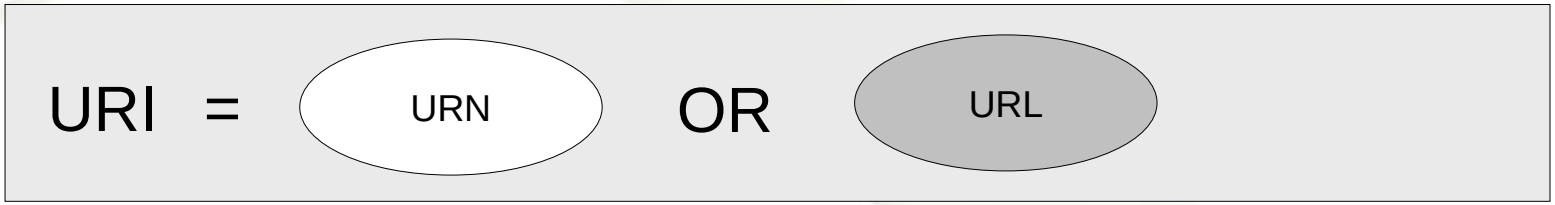


- 네임스페이스의 사용
  - 네임스페이스 선언에서 정의된 접두어(Prefix)를 엘리먼트의 이름 앞에 ":"과 함께 붙여서 사용한다.

```
<prefix:tag xmlns:prefix="url or uri">
…
</prefix:tag>
```

네임스페이스 식별자 URI

- URI(Uniform Resource Identifier)
  - 인터넷 상의 자원을 표현하기 위한 방법



  - URN(Uniform Resource Name)
    - 인터넷 상의 자원을 이름으로 표시하는 방법
    - urn:def://blue_laser
  - URL(Uniform Resource Locator)
    - 인터넷 상의 자원을 자원이 존재하는 위치로 표현하는 방법
    - http://www.xmlcommunity.net/blue_laser.html

- 네임스페이스 식별자 적합성
  - URN은 독립성을 보장하기 힘들다.

네임스페이스의 의미

- 네임스페이스가 갖는 의미
  - 단순한 구분을 위한 고유한 문자열
  - 네임스페이스가 적용된 엘리먼트들은 선언된 네임스페이스 안에서 고유한 의미를 갖는 요소로 해석됩니다.

- 기본 네임스페이스
  - 접두어가 없이 선언된 네임스페이스
  - 접두사 없이 사용되는 모든 요소는 기본 네임스페이스에 포함된 요소로 해석됩니다.
  - 한 요소에 기본 네임스페이스는 하나만 정의할 수 있습니다.

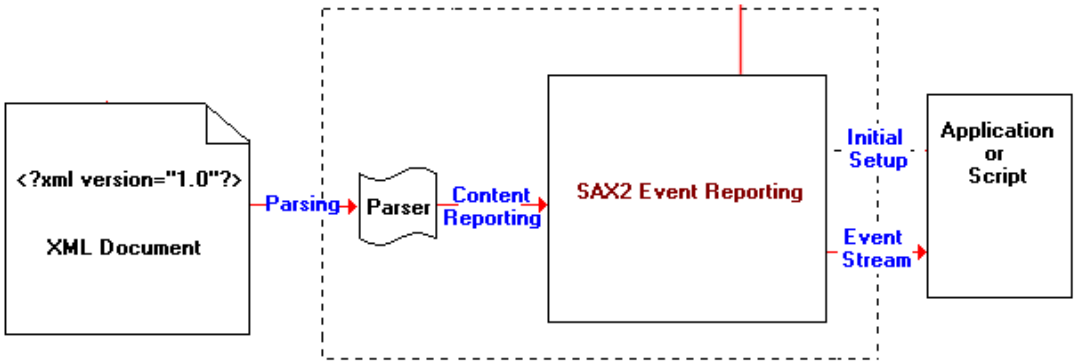
네임스페이스의 선언과 적용

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <person xmlns="http://sernaferna.com/pers">
3.    <name id="1">
4.      <title>Sir</title>
5.      <first>John</first>
6.      <middle>Fitzgerald Johansen</middle>
7.      <last>Doe</last>
8.    </name>
9.    <position>Vice President of Marketing</position>
10.   <resume>
11.     <html:html xmlns:html="http://www.w3.org/1999/xhtml">
12.     <html:head><html:title>Resume of John Doe</html:title></html:head>
13.     <html:body>
14.     <html:h1>John Doe</html:h1>
15.     <html:p html:style="FONT-FAMILY: Arial">
16.       John's a great guy, you know?
17.     </html:p>
18.     </html:body>
19.     </html:html>
20.   </resume>
21. </person>
```

03. XML Processing

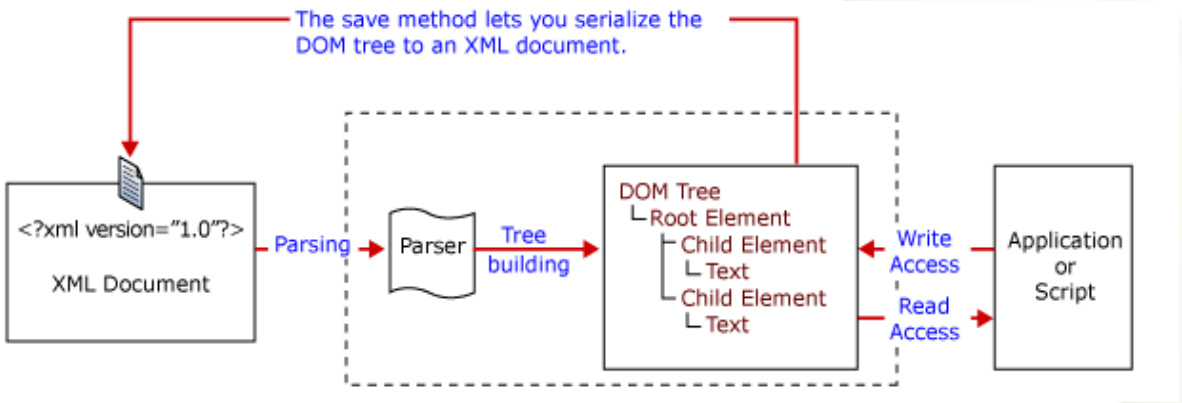
파싱을 통한 xml문서 처리 (SAX)

- 이벤트를 이용한 전진 / 읽기 전용 문서 접근
- 현재 노드에만 접근하기 때문에 성능이 좋은 장점 제공
- 반면 문서 전체에 대한 Random Access는 지원하지 않습니다.



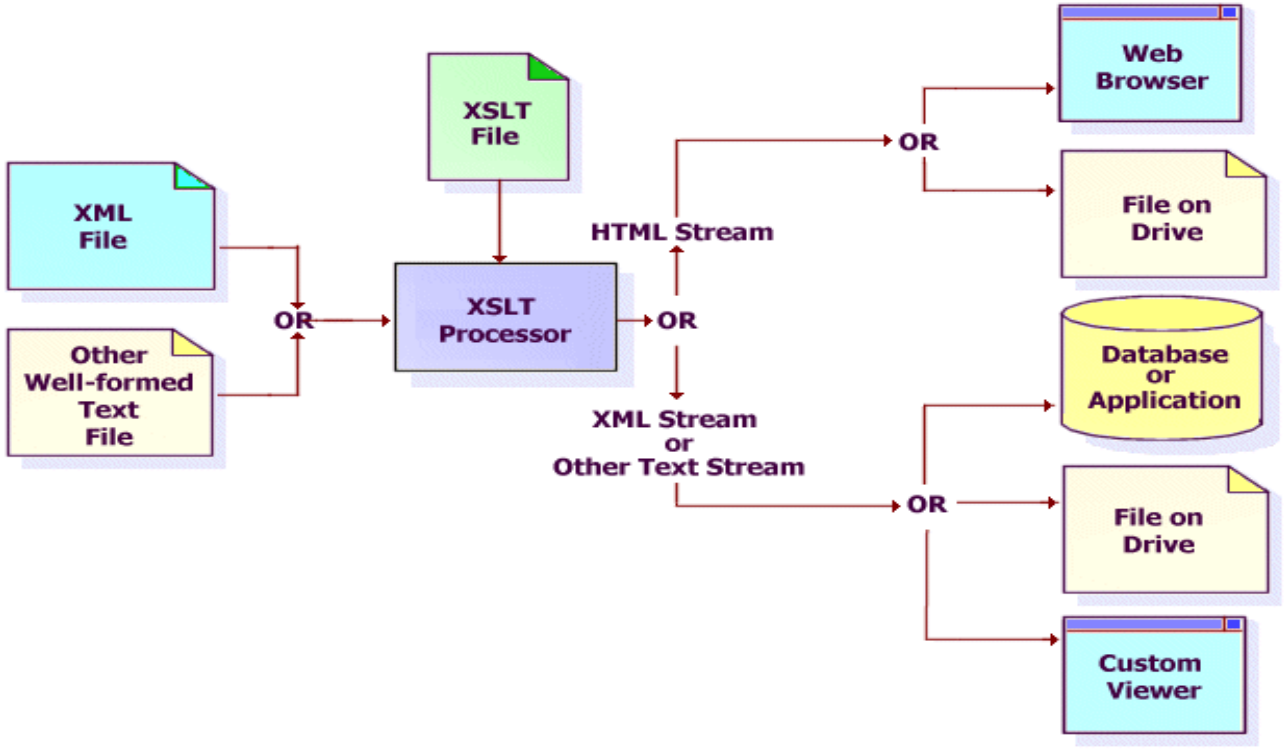
파싱을 통한 xml문서 처리 (DOM)

- 파서가 문서 전체 또는 일부를 읽어서 메모리에 객체의 트리를 구성해서 xml 어플리케이션에 제공
- SAX에 비해 읽기 성능은 낮지만 Random Access 및 쓰기 기능이 제공됨
- 표준 인터페이스를 구현한 객체 기반 접근



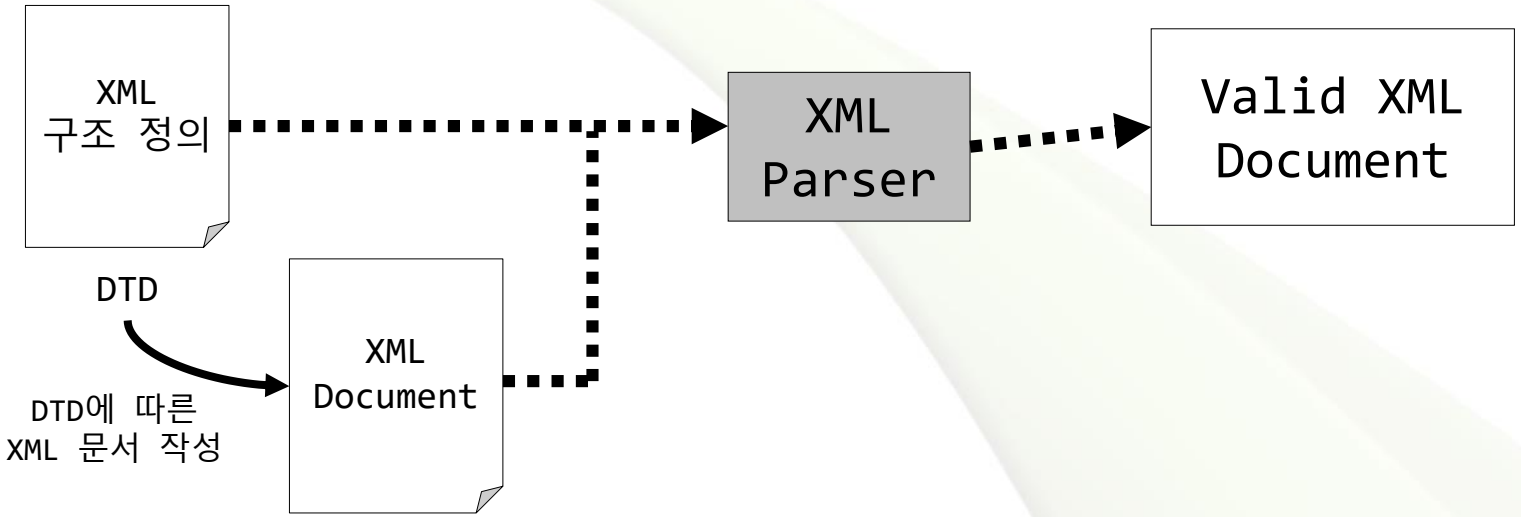
템플릿을 통한 xml문서 처리 (XSL)

- eXtensible Stylesheet Language
- Xml 문서의 변환을 지원
  - Xslt : xml, html, text 변환
  - Xsl-fo : pdf 등 바이너리 형식 변환



문서구조정의 및 유효성 검사 (DTD)

- DTD란?
  - XML 문서의 구조를 정의하기 위한 언어



- DTD의 특징
  - 간결한 문법으로 비교적 작성이 쉽다.
  - 그러나, XML과 다른 문법을 사용한다.

DTD (예제)

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

<!DOCTYPE Computer [
<!ELEMENT Computer (CPU, Memory, GraphicCard)>
<!ELEMENT CPU (#PCDATA)>
<!ELEMENT Memory (#PCDATA)>
<!ELEMENT GraphicCard (#PCDATA)>
]>

<Computer>
        <CPU>Pantium III</CPU>
        <Memory>512 MB</Memory>
        <GraphicCard>Radeon 9200</GraphicCard>
</Computer>
```

문서구조정의 및 유효성검사(XML Schema)

- DTD와 함께 XML 문서의 구조와 내용을 정의하는 언어

- DTD보다 구체적인 XML 문서의 구조를 정의 할 수 있습니다.
  - XML 문법을 사용

<!ELEMENT title (#PCDATA)>	<xsd:element name="title" type="xsd:string" />

  - 광범위한 자료형 제공

<!ELEMENT title (#PCDATA)> <!ELEMENT number (#PCDATA)> <!ELEMENT date (#PCDATA)>	<xsd:element name="title" type="xsd:string" /> <xsd:element name="number" type="xsd:integer" /> <xsd:element name="title" type="xsd:date" />

XML Schema의 데이터 타입

- 기본적으로 제공되는 데이터 타입

데이터 타입	설명	예시
string	문자열 값	Hello, World
boolean	참, 거짓	true | false
decimal	소수	7.08
time	시간 데이터 타입	hh:mm:ss.sss
date	날짜 데이터 타입	CCYY-MM-DD
duration	기간을 위한 데이터 타입	P1Y2M3DT10H30M12.3S
integer	정수 값	51
ID	유일한 값을 위한 데이터 타입 (어트리뷰트에서만 사용 가능)	

- 확장 데이터 타입
  - Complex Type 을 이용한 사용자 정의 타입 정의 (구조에 대한 정의)
  - Simple Type을 이용한 사용자 정의 타입 정의 (값에 대한 정의)

간단한 XML Schema 작성 (예제)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                elementFormDefault="qualified">
    <xsd:element name="car">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="name" type="xsd:string" />
                <xsd:element name="CC" type="xsd:integer" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

문서내 요소 접근 표준 (XPath)

- XML 문서 내의 요소와 어트리뷰트를 지정하고 접근하기 위한 표준 문법
- 트리 구조를 기반으로 한 경로 표현 방법 제공
- DOM, XSLT 등의 표준과 함께 사용