제 8 장 XSL로 XML 문서 구조 변경

1. XSL 소개
2. XSL 처리 과정 이해
3. XSL 프로세서(Processor) 설치
4. XSL 문서를 XML 문서에 적용하는 방법
5. XSL 문서의 루트 엘리먼트
6. 리절트 트리의 문서 종류 결정
7. 템플레이트 룰(Template Rule)
8. 원본 XML 문서의 내용 가져오기
9. 정렬 시키기
10. 번호 매기기
11. 제어 엘리먼트
12. 변수와 파라메터
13. XSL 문서의 결합

1. XSL 소개

1-1 스타일쉬트(CSS)의 문제점과 XSL 탄생

- 스타일쉬트를 XML 문서에 적용하는 데 따르는 제약 사항.

① 스타일쉬트는 출력되는 내용을 XML 문서의 원래 구조대로 출력하는 것을 원칙으로 한다. 입력 XML 문서와 전혀 다른 구조로 출력하는 것은 불가능 하다.

② 스타일쉬트는 XML 문서에 없는 새로운 내용이나 계산된 결과를 출력할 수 있는 기능을 제공하지 않는다.

③ 스타일쉬트는 XML 문서의 모든 내용을 출력할 수 없다.
- 속성값은 출력할 수 없다.

1. XML 문서 표현

1-2 XSL 구성 파트

- XSLT(XSL Transformation)
- XPath(XML Path Language)
- XSL-FO(XSL Formatting Objects)

1) XSLT(XSL Transformation)

XSLT는 XML 문서의 구조를 다른 구조로 변환시키기 위해 설계된 마크업 언어이다.
1999년 11월 16일에 발표한 XSLT Version 1.0이 최신 버전이다.

2) XPath(XML Path Language)

XML 문서 내에서 특정 파트(엘리먼트 또는 속성)를 찾아가기 위해 사용되는 경로 표기 언어이다.
XPath는 XSLT와 XPointer에서 사용하기 위해 설계된 언어이다.
현재는 1999년 11월 16일에 발표한 XPath Version 1.0이 최신 버전이다.

3) XSL-FO(XSL Formatting Objects)

XSL-FO 언어는 XML 문서를 비 XML 문서로 만들기 위해서 설계된 언어이다.
포맷터(Formatter)라는 프로그램을 통해서 XML 문서를 비 XML 문서로 변환하기 위해, 먼저 XML 문서는 XSL-FO 마크업 언어로 작성되어야 한다.

2. XSL 처리 과정 이해

2-1 XSL 처리 과정 종류

- XSL 처리 과정(XSL 프로세스)은 두 가지 처리 과정으로 나누어 진다.
  - ■ Transformation : XML 문서에서 다른 구조의 XML 문서로 구조를 변환하는 과정
  - ■ Formatting : XML 문서를 특정 S/W, H/W에 맞는 비 XML 문서로 변환하는 과정

2-2 Transformation 처리 과정

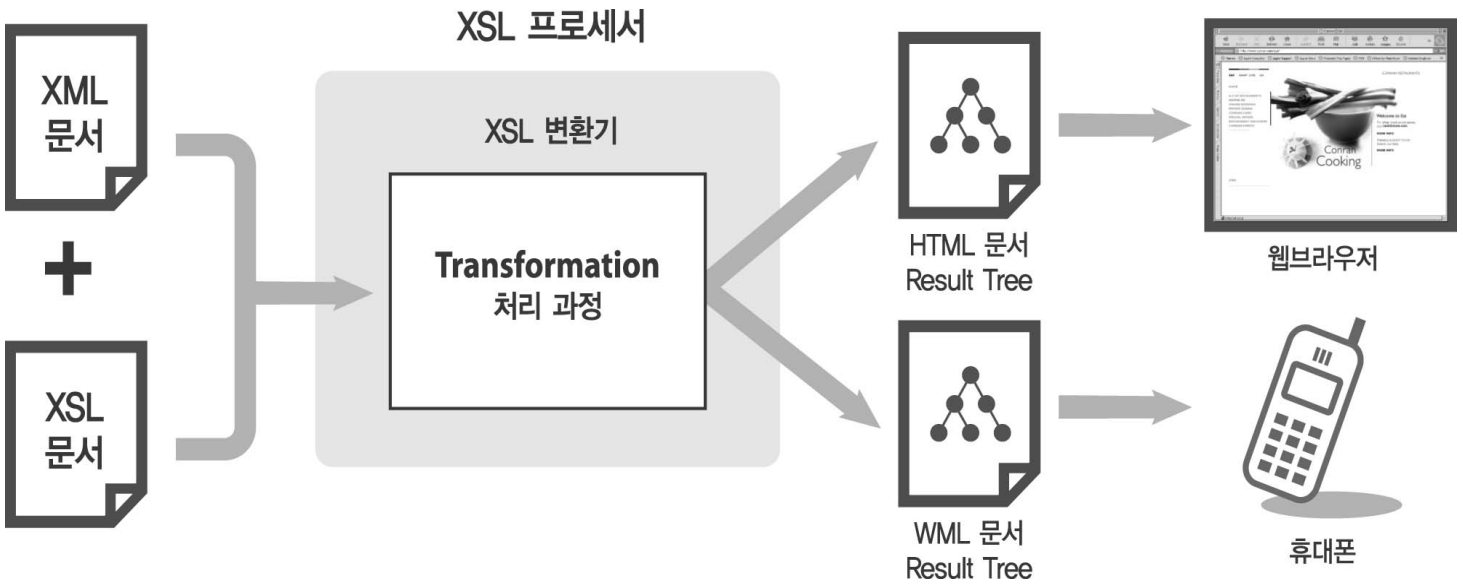
- XML 문서에서 다른 구조의 XML 문서로 구조를 변환하는 과정이다.
- 구조를 변경한다는 말은 특정 마크업 언어로 작성된 XML 문서를 다른 마크업 언어로 작성된 XML 문서로 바꾸는 것을 말한다.

  예) XML 문서 ➔ HTML(XHTML) 문서

  XML 문서 ➔ WML(Wireless Markup Language) 문서

2. XSL 처리 과정 이해

- Transformation 처리 과정



- XSL 변환기의 결과물은 메모리 상에서 트리를 이루는 DOM 객체들로 생성된다. (Result Tree)

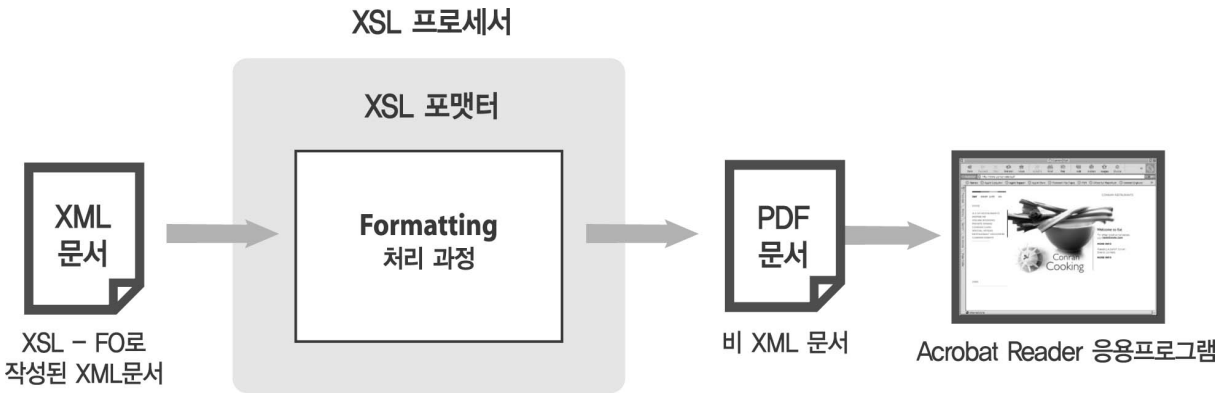
- Transformation 처리 과정을 담당하는 프로그램을 XSL 변환기(XSL Transformer)라고 한다. XSL 변환기는 XSL 프로세서(Processor) 프로그램에 포함이 된다.

2. XSL 처리 과정 이해

2-3 Formatting 처리 과정

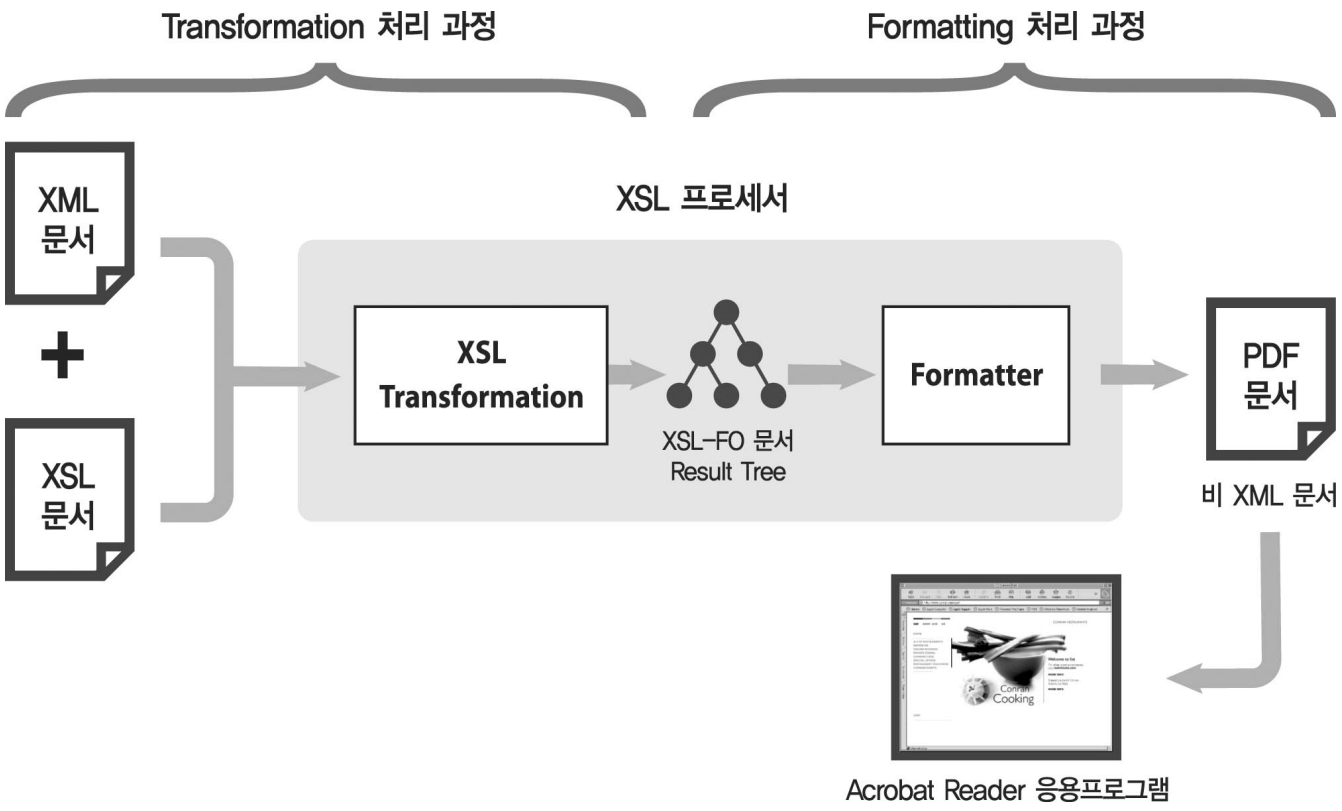
- XML 문서를 특정 S/W, H/W에 맞는 비 XML 문서로 변환하는 과정을 Formatting 처리 과정이라고 한다.
- 비 XML 문서는 XML 권고안 문법을 따라 작성된 텍스트 형식의 파일이 아닌 특정 포맷 형식으로 작성된 문서를 말한다.
- XML 문서가 비 XML 문서로 변환하기 위해서는, XML 문서는 XSL-FO 언어로 작성되어 져야 한다. XSL-FO 언어로 작성된 XML 문서를 포맷터(Formatter)가 비 XML 문서로 만든다.
- Formatting 처리 과정을 담당하는 프로그램을 포맷터(Formatter)라고 한다. 포맷터는 XSL 프로세서 프로그램에 포함이 된다.

- Formatting 처리 과정



2. XSL 처리 과정 이해

- XML 문서가 XSL-FO 언어로 작성되어 있지 않으면, XSL 변환기를 거쳐 XSL-FO 문서 구조로 변경하고 Formatting 처리 과정을 해야 한다.



2. XSL 처리 과정 이해

2-4 XSL 프로세서(Processor)

- 일반적으로 XSL 프로세서는 XSL 변환기와 포맷터 기능을 내장하고 있다. 하지만 현재 나와있는 XSL 프로세서는 XSL 변환기 기능만 구현하고 있다.

- XSL 프로세서가 XSL 변환기 기능만 구현한 이유.

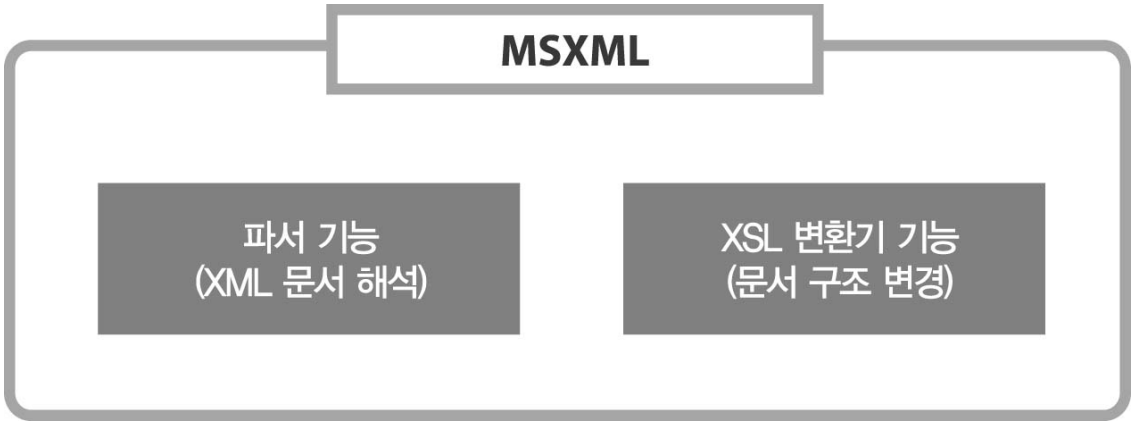
  ■ 포맷터에서 사용되는 XSL-FO 마크업 언어가 XSL 변환기에서 사용하는 XSLT 마크업 언어보다 늦게 권고안으로 채택되었다.

  ■ XML 문서를 비 XML 문서로 변환하는 기능은 특별한 경우를 제외하고는 거의 사용되지 않는다.

3. XSL 프로세서 설치

3-1 MSXML에 포함되어 있는 XSL 프로세서 이용

- 인터넷 익스플로러 6.0에 있는 MSXML에는 파서 기능과 XSL 프로세서(Processor) 기능이 포함되어 있다. 그러나 Transformation 처리를 하는 XSL 변환기 기능만 포함되어 있다.

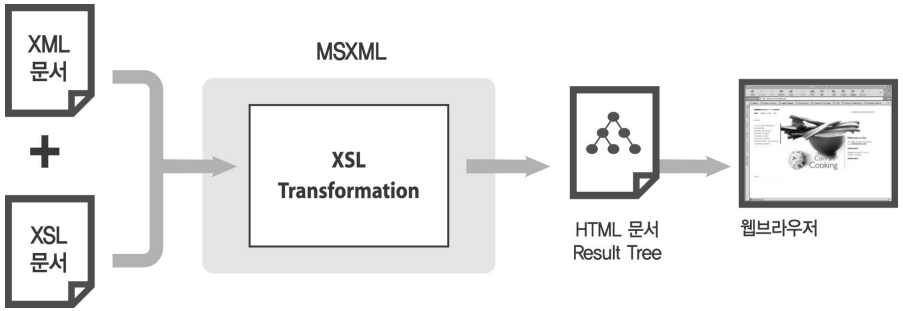


- MSXML의 XSL 변환 기능 테스트

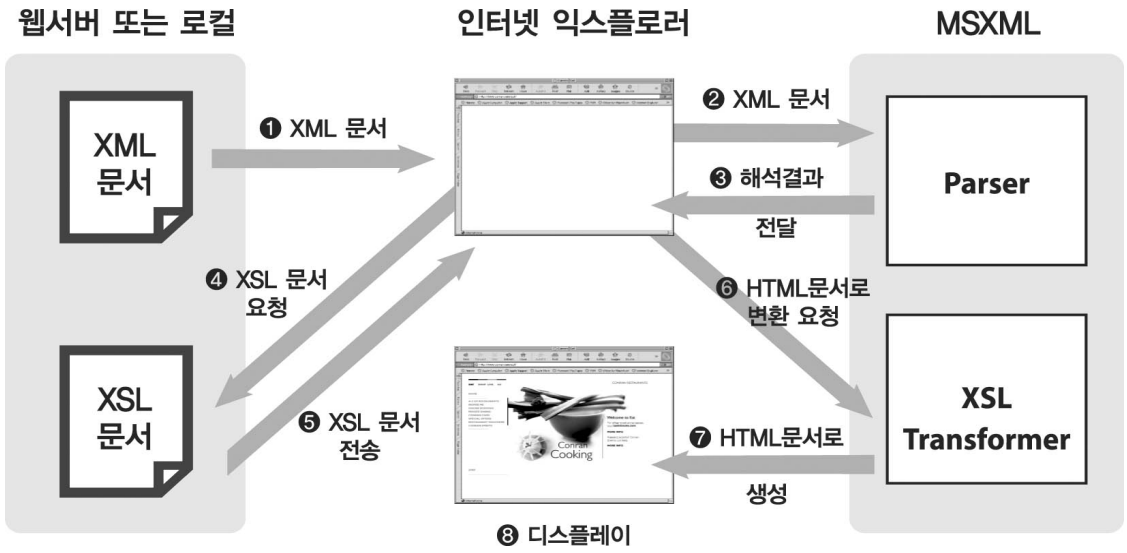
XML 문서 : c8_0301.xml
CSS 문서 : c8_0301.xsl

3. XSL 프로세서 설치

- MSXML의 XSL 변환 과정.



- XSL 프로세서 내부 처리 과정.



3. XSL 프로세서 설치

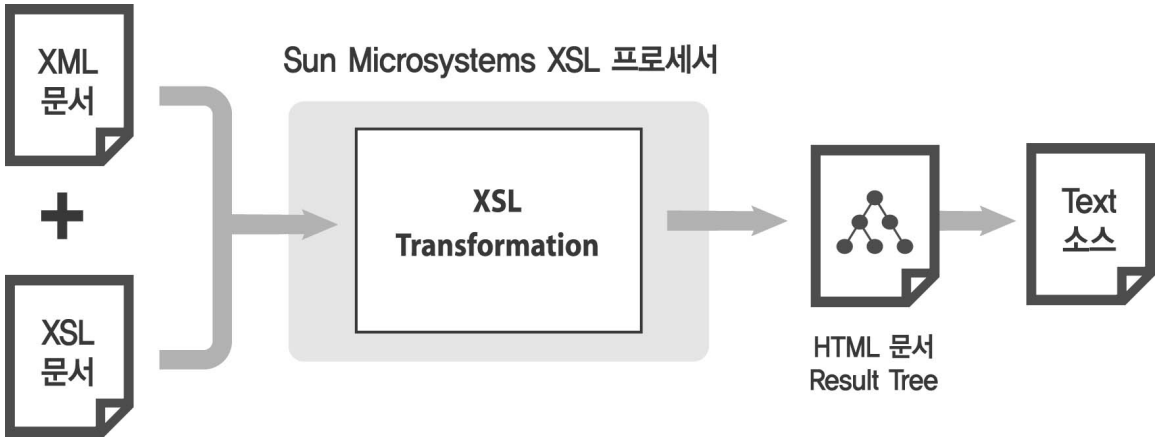
3-2 인터넷 익스플로러에서 변환된 HTML 문서 보기

- MSXML의 Transformer는 변환된 HTML 문서를 리절트 트리 형태의 DOM 개체로 생성한다.
- XSL 변환기를 거쳐 나온 리절트 트리 형태의 결과물을 텍스트 형태의 소스로 확인할 수 있다.

  ➔ 446페이지 절차대로 iexmltls.exe 설치

3-3 다른 XSL 프로세서를 이용

- MSXML 3.0에 있는 XSL 변환기는 W3C XSLT(XSL Transformation) Version 1.0 권고안의 내용을 완벽하게 따르지 않고있다.
- Sun Microsystems에서 배포하는 XSL 프로세서를 이용해서 XML 문서를 변환시켜, 변환된 리절트 트리를 텍스트 형태의 소스로 출력해서 확인할 수 있다.



4. XSL 문서를 XML 문서에 적용하는 방법

- XSL 문서를 XML 문서에 적용하는 방법.

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

- type 속성값에는 "text/xsl"을, href 속성값으로 XSL 문서의 경로를 입력한다.

XML 문서 : c8_0301.xml

```
1  <?xml version="1.0" encoding="euc-kr"?>
2
3  <!-- 프로세싱 지시자 -->
4  <?xml:stylesheet type="text/xsl" href="c8_0301.xsl"?>
5
6  <booklist>
7      <book id="b1" kind="computer">
8          <title>XML 중급</title>
9          <author>이 규미</author>
10         <publisher>Freelec</publisher>
11         <price>25000</price>
12     </book>
13 </booklist>
```

5. XSL 문서의 루트 엘리먼트

5-1 루트 엘리먼트

- XSL 문서의 루트 엘리먼트 : stylesheet or transform 사용.
- 모든 XSLT 언어의 엘리먼트들은 관례적으로 접두사로 xsl을 사용한다.

문법	xmlns:xsl=http://www.w3c.org/1999/XSL/Transform

- XSL 문서의 루트 엘리먼트 작성 문법.

문법

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

            자식 엘리먼트 (최상위 엘리먼트)


</xsl:stylesheet >
```

5. XSL 문서의 루트 엘리먼트

5-2 루트 엘리먼트의 자식 엘리먼트

- 최상위 엘리먼트(Top-level 엘리먼트)의 종류.

■ XSL 문서의 결합과 관련된 최상위 엘리먼트
- Import, include

■ 리절트 트리의 문서 종류를 결정하는 최상위 엘리먼트
- output

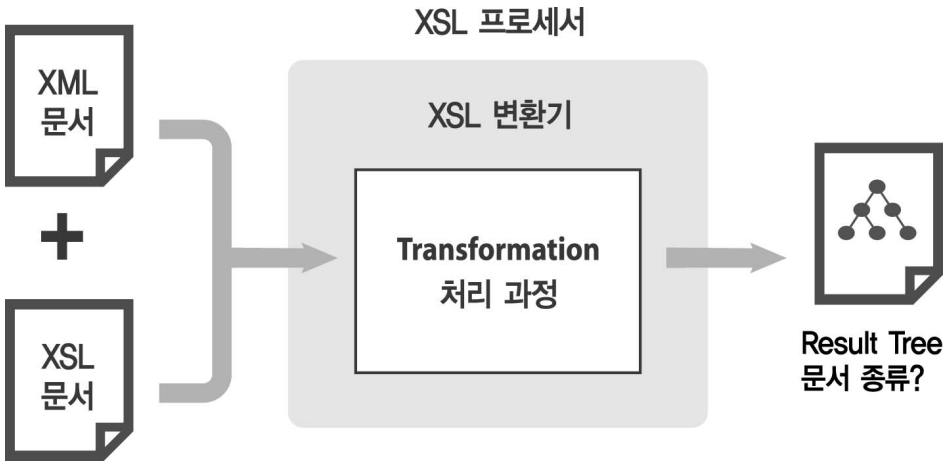
■ 리절트 트리의 엘리먼트 내용을 담고있는 최상위 엘리먼트
- template

■ 기타 최상위 엘리먼트
- strip-space, preserve-space, key, decimal-format,
- namespace-alias, attribute-set, variable, param

6. 리절트 트리의 문서 종류 결정

6-1 output 엘리먼트

- 리절트 트리 문서의 종류를 기술한다.



- output 엘리먼트로 리절트 트리의 XML 선언과 문서 유형 선언에 대해서도 지정할 수 있다.

문법	<xsl:output 속성명="속성값 속성명="속성값" … />

6. 리절트 트리의 문서 종류 결정

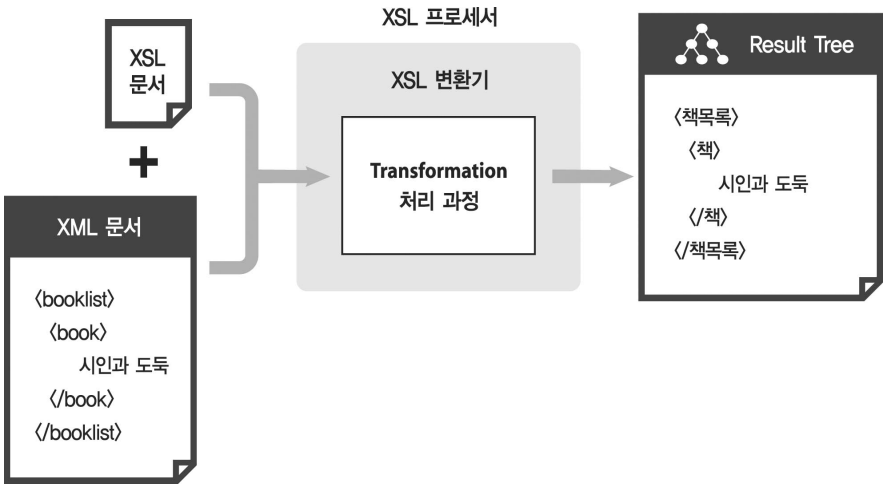
- output 엘리먼트가 가질 수 있는 속성

속성명	설명	속성값 유형
method	▪ 리절트 트리의 문서 종류를 지정	xml, html, text
version	▪ 리절트 트리가 XML 문서일 경우 XML 권고안 버전	1.0
encoding	리절트 트리의 인코딩 방식을 지정	string
omit-xml-declaration	▪ 리절트 트리의 XML 선언 생략 여부	yes, no
standalone	▪ 리절트 트리의 외부 문서 참조 여부	yes, no
doctype-public	▪ 리절트 트리가 문서유형선언을 포함할 경우 PUBLIC식별자 지정	string
doctype-system	▪ 리절트 트리가 문서유형선언을 포함할 경우 SYSTEM식별자 지정	string
cdata-section-elements	▪ 리절트 트리에서 CDATA 섹션으로 사용되어야 할 노드 지정	QName
indent	▪ 리절트 트리에 화이트스페이스를 포함할 것인지 여부	yes, no
media-type	▪ media type 지정	string

6. 리절트 트리의 문서 종류 결정

6-2 다른 구조의 XML 문서로 변환

- 영어 엘리먼트로 작성된 XML 문서를 한글 엘리먼트로 구성된 XML 문서 리절트 트리로 변환.



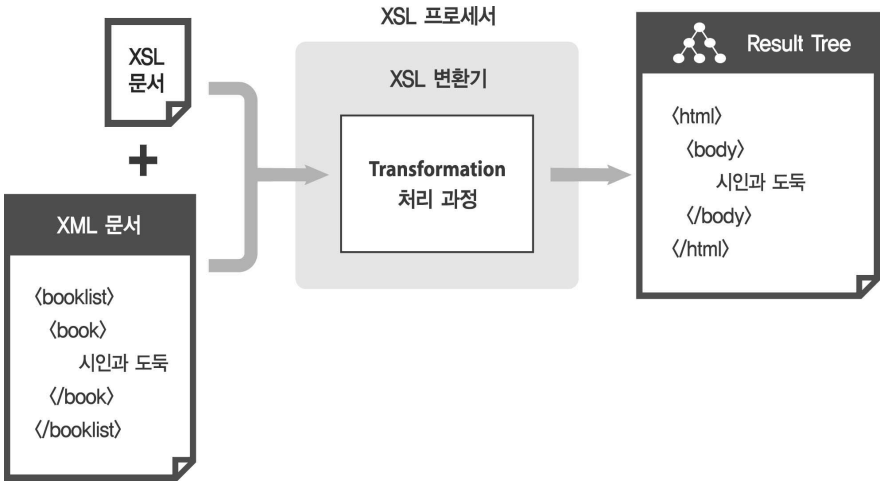
- 실습 예제.

XML 문서 : c8_0601.xml
XSL 문서 : c8_0601.xsl

6. 리절트 트리의 문서 종류 결정

6-3 HTML 문서로 변환

- 영어 엘리먼트로 작성된 XML 문서를 HTML 문서 리절트 트리로 변환.



- 실습 예제.

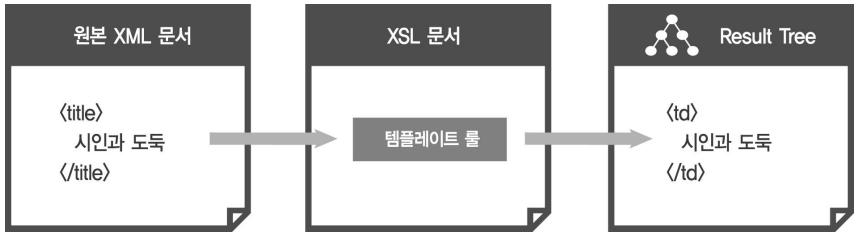
XML 문서 : c8_0602.xml
XSL 문서 : c8_0602.xsl

7. 템플레이트 룰(Template Rule)

7-1 템플레이트 룰의 개념

- Template Rule은 어떤 구조를 다른 구조로 바꾸는 방법을 말한다.

- XSL 문서에서의 템플레이트 룰에는 원본 XML 문서의 특정 노드(엘리먼트)를 다른 구조의 노드(엘리먼트)로 바꾸는 내용이 기술된다.

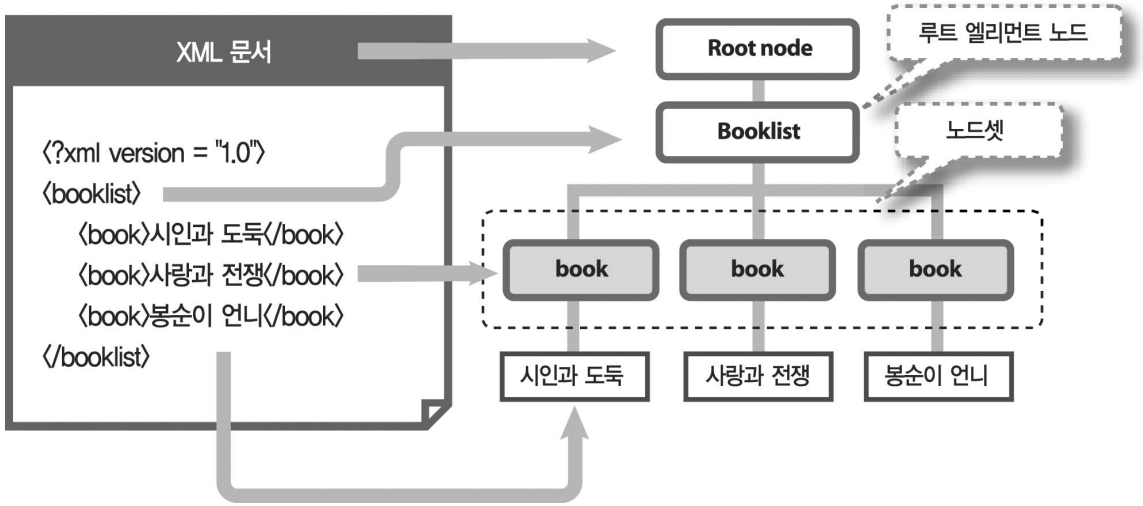


- XSL 문서를 작성하는 것은 XML문서에 적용할 템플레이트 룰을 작성하는 과정이다.

7. 템플레이트 룰(Template Rule)

7-2 노드 및 노드셋의 개념

- 노드 : 트리 구조에서 사용되는 용어로 매듭 부분.



기호 구분	설명
루트 노드	XML 문서 자체를 뜻하는 노드
엘리먼트노드	루트 엘리먼트를 포함해서 자식 엘리먼트를 뜻하는 노드
텍스트노드	데이터 부분을 뜻하는 노드
노드셋	같은 이름을 가지고 있는 노드들

7. 템플레이트 룰(Template Rule)

7-3 템플레이트 룰의 정의

- 템플레이트 룰은 원본 XML 문서에서 변환될 대상 노드의 숫자 만큼 정의되어야 한다.

- XSL 문서에서 템플레이트 룰 정의하기.

문법

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

<!-- 템플레이트 룰 -->
<xsl:template  match="원본 XML 문서의 대상 노드 지정">
    대상 노드 대신에 대치되는 새로운 노드 또는 내용

</xsl:template>

<!-- 템플레이트 룰 -->
<xsl:template  match="원본 XML 문서의 대상 노드 지정">
    대상 노드 대신에 대치되는 새로운 노드 또는 내용

</xsl:template>
  ..
  ..
</xsl:stylesheet >
```

7. 템플레이트 룰(Template Rule)

- match 속성에는 원본 XML 문서에서 변환되어야 할 대상 노드를 지정하는 패턴(Pattern)이 온다.
- 패턴에는 XPath 표현식 중에서 한 개의 노드 또는 노드셋을 가리키는 표현식이 와야 한다.

자주 사용되는 패턴

패턴	설명
/	문서 자체를 뜻하는 루트 노드를 지정
/노드명/.../노드명	특정 경로를 가지는 노드 또는 노드들을 지정
//노드명 또는 노드명	경로와는 상관없이 동일한 이름의 노드 또는 노드들을 지정
/노드명/노드명[@속성명=”속성값”]	특정 속성값을 가지는 노드 또는 노드들을 지정
/노드명/노드명[@속성명!=”속성값”]	특정 속성값과 다른 속성값을 가지는 노드 또는 노드들을 지정

7. 템플레이트 룰(Template Rule)

- 템플레이트 룰(Template Rule) 정의 예제.

XSL 문서	<xsl:template match="/"> ~~ </xsl:template>
XSL 문서	<xsl:template match="//book"> ~~ </xsl:template>
XSL 문서	<xsl:template match="book"> ~~ </xsl:template>
XSL 문서	<xsl:template match="/booklist/book/title"> ~~ </xsl:template>
XSL 문서	<xsl:template match="/booklist/book[@kind='computer']"> ~~ </xsl:template>

7. 템플레이트 룰(Template Rule)

7-4 템플레이트 룰의 적용

- XSL 변환기는 템플레이트 룰의 match 속성에 루트 노드("/")가 지정된 템플레이트 룰을 찾아서 적용시킨다.

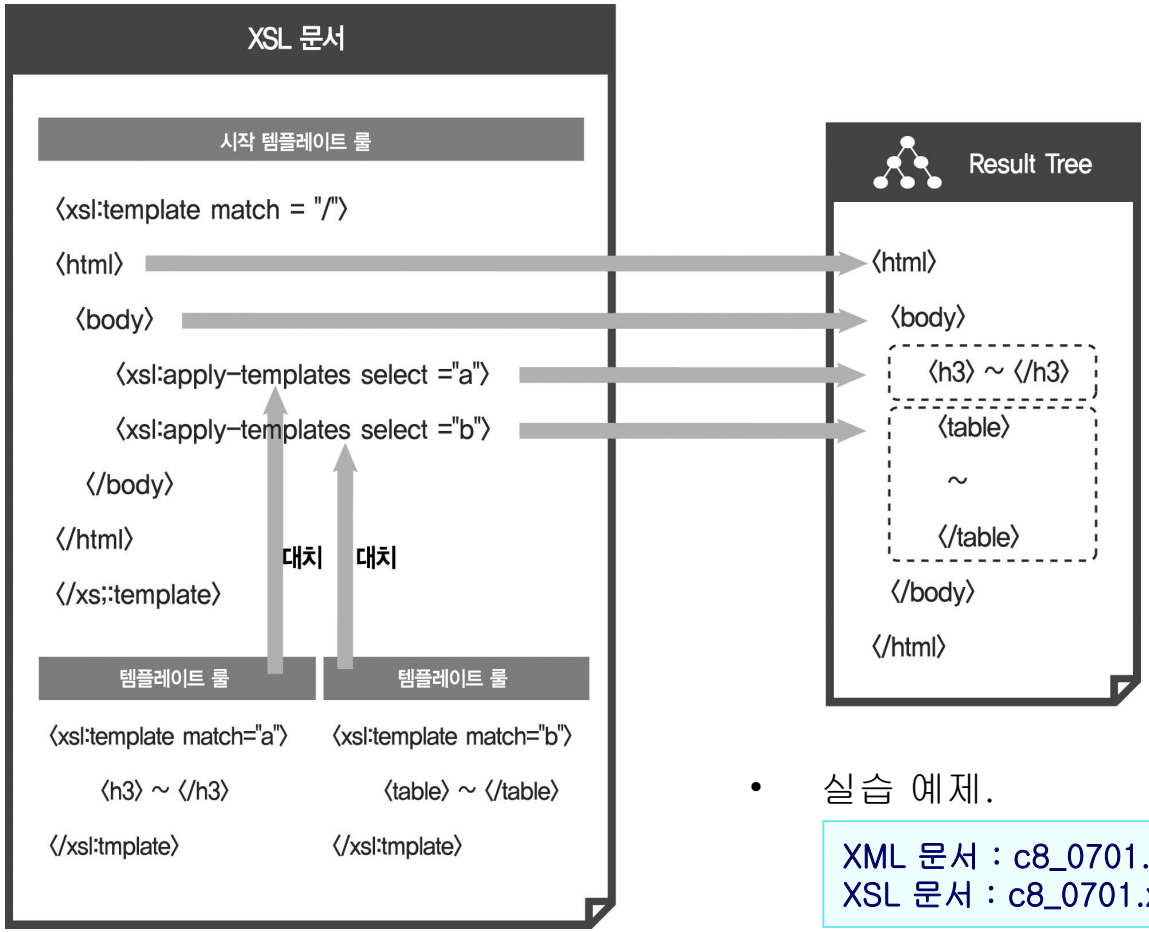
```
<xsl:template match="/">  ← 시작 템플레이트 룰
    ~~
</xsl:template>
```

문법

```
<!-- 시작 템플레이트 룰 -->
<xsl:template match="/">
    <루트엘리먼트>
        <엘리먼트>
            <xsl:apply-templates select="원본 XML 문서의 대상 노드 지정"/>
        </엘리먼트>
    </루트엘리먼트>
</xsl:template>
```

7. 템플레이트 룰(Template Rule)

7-4 템플레이트 룰의 적용



- 실습 예제.

XML 문서 : c8_0701.xml
XSL 문서 : c8_0701.xsl

7. 템플레이트 룰(Template Rule)

7-5 템플레이트 룰의 name 속성

- name 속성이 지정된 템플레이트 룰은 XSL 문서 내에서 반복해서 자주 적용되는 내용을 한 개의 템플레이트 룰로 만들고, 다른 템플레이트 룰에서 참조해서 적용할 경우 사용된다.

- 템플레이트 룰에는 사용 용도에 따라 match 속성 또는 name 속성 둘 중 하나를 기술해야 한다.

문법	<xsl:template name="템플레이트 룰명"> 반복해서 자주 적용되는 내용 </xsl:template >

- name 속성을 갖는 템플레이트 룰 적용하기.

문법	<xsl:call-template name="템플레이트 룰명"/>

- 실습 예제.

XML 문서 : c8_0702.xml
XSL 문서 : c8_0702.xsl

7. 템플레이트 룰(Template Rule)

7-6 템플레이트 룰의 priority 속성

- 원본 XML 문서의 대상 노드를 변환시키는 템플레이트 룰이 여러 개가 있을 경우 우선순위가 높은 템플레이트 룰이 적용될 수 있도록 할 수 있다.
- 템플레이트 룰을 정의할 때 priority 속성에 우선순위를 지정할 수 있다.

문법	<xsl:template match="원본 XML 문서의 대상 노드 지정" priority="우선순위번호"> ~~ </xsl:template> 번호가 클수록 우선순위가 높다.

- 우선순위번호에는 음수 또는 양수의 실수값이 온다.
- priority 속성은 name 속성을 갖는 템플레이트 룰에는 사용할 수 없다.
- 실습 예제.

XML 문서 : c8_0703.xml
XSL 문서 : c8_0703.xsl

7. 템플레이트 룰(Template Rule)

7-7 템플레이트 룰의 mode 속성

- 템플레이트의 mode 속성은 match 속성값이 동일한 템플레이트 룰을 여러 개 만들 수 있도록 해 주고, mode 이름으로 구분해서 각각의 템플레이트 룰을 적용시킬 수 있다.

문법	<xsl:template match="원본 XML 문서의 대상 노드 지정" mode="모드명"> ~~ </xsl:template>

- mode 속성은 name 속성을 갖는 템플레이트 룰에는 기술하지 않고 반드시 match 속성을 갖는 템플레이트 룰에만 기술된다.

- xsl:apply-templates 엘리먼트 작성 방법

문법	<xsl:apply-templates select="대상 노드 지정" mode="모드명"/>

- 실습 예제.

XML 문서 : c8_0704.xml
XSL 문서 : c8_0704.xsl

8. 원본 XML 문서의 내용 가져오기

8-1 엘리먼트의 컨텐트 내용 및 속성값 가져오기

- 가져오는 컨텐트 내용이 숫자일 경우 .

문법	<xsl:templates match="원본 XML 문서의 대상 노드 지정"> **<xsl:value-of select="format-number(자식 노드명, '포맷 형식')"/>** </xsl:apply-tempaltes>
XSL 문서	<xsl:templates match="/booklist/book"> <xsl:value-of select="format-number(price, '###,###')"/> </xsl:apply-tempaltes>

- 가져오는 데이터가 속성값일 경우.

문법	<xsl:templates match="원본 XML 문서의 대상 노드 지정"> **<xsl:value-of select="@속성명"/>** </xsl:apply-tempaltes>
XSL 문서	<xsl:templates match="/booklist/book"> <xsl:value-of select="@kind"/> </xsl:apply-tempaltes>

- 실습 예제.

XML 문서 : c8_0801.xml
XSL 문서 : c8_0801.xsl

8. 원본 XML 문서의 내용 가져오기

8-2 엘리먼트 및 속성을 복사해서 가져오기

- 원본 XML 문서의 일부분을 복사해서 변환될 XML 문서에 붙여 사용할 수 있다.(속성도 복사 가능)

문법	<xsl:templates match="원본 XML 문서의 대상 노드 지정"> <xsl:copy-of select="복사할 자식 노드명 및 @속성명"/> </xsl:apply-tempaltes>

- 실습 예제.

XML 문서 : c8_0802.xml
XSL 문서 : c8_0802.xsl

9. 정렬 시키기

- xsl:sort 엘리먼트를 이용해서 정렬하기.

문법	<xsl:apply-templates select=”원본 XML 문서의 대상 노드 지정”> <xsl:sort select=대상 노드를 정렬시킬 자식 노드명 또는 속성명” data-type=”데이터 유형("text" | "number") default : text order=정렬방식("ascending" | "descending")/> </xsl:apply-tempaltes>

- xsl:sort는 xsl:apply-templates 엘리먼트와 xsl:for-each 엘리먼트의 자식 엘리먼트로 작성된다.

<< 예제 >>

XSL 문서	<xsl:apply-templates select="/booklist/book"> <xsl:sort select="title"/> </xsl:apply-templates>

XSL 문서	<xsl:apply-templates select="/booklist/book"> <xsl:sort select="@kind"/> </xsl:apply-templates>

- 실습 예제.

XML 문서 : c8_0901.xml
XSL 문서 : c8_0901.xsl

9. 정렬 시키기

- 2차 정렬이란 특정 필드를 먼저 정렬시키고(1차정렬), 다시 다른 필드를 정렬시키는 것이다.
- 2차 정렬을 하기 위해서 xsl:sort 엘리먼트를 2번 기술한다.

XSL 문서

```
<xsl:apply-templates select="/booklist/book">
        <!-- 1차 정렬 -->
        <xsl:sort select="@kind"/>
        <!-- 2차 정렬 -->
        <xsl:sort select="price" data-type="number" order="ascending"/>
</xsl:apply-templates>
```

- 실습 예제.

XML 문서 : c8_0902.xml
XSL 문서 : c8_0902.xsl

10. 번호 매기기

- XSL 변환기는 리절트 트리를 생성할 때 자동적으로 번호를 매겨 삽입할 수 있는 기능이 있다.

문법	<xsl:templates match="원본 XML 문서의 대상 노드 지정"> <xsl:number format="번호형식"/> </xsl:apply-tempaltes>

- 번호형식에는 '1', 'a', 'A', 'i', 'I', '(1)' 와 같은 형식들이 올 수 있다.
- 실습 예제.

XML 문서 : c8_1001.xml
XSL 문서 : c8_1001.xsl

11. 제어 엘리먼트

11-1 xsl:if 엘리먼트

- xsl:if 엘리먼트는 조건식을 평가하여 변환 내용을 선택적으로 적용시킨다.

문법	<xsl:templates match="원본 XML 문서의 대상 노드 지정">
	  <xsl:if test="조건식">
	  변환 내용
	  </xsl:if>
	</xsl:apply-tempaltes>

- 조건식에 자주 오는 XPath 표현식

조건식	설명
노드명="문자열"	컨텐트 내용이 주어진 문자열 가지면 참
@속성명='속성값'	주어진 속성값을 가지면 참
@속성명!='속성값'	주어진 속성값이 아닌 다른 속성값을 가지면 참
starts-with(노드명, '문자열')	컨텐트 내용이 주어진 문자열로 시작하면 참
contains(노드명,'문자열')	컨텐트 내용이 주어진 문자열을 포함하면 참

- 실습 예제.

XML 문서 : c8_1101.xml
XSL 문서 : c8_1101.xsl

11. 제어 엘리먼트

11-2 xsl:choose 엘리먼트

- xsl:choose 엘리먼트는 여러 조건식을 평가해서 참을 돌려주는 조건식을 포함하는 변환 내용만 적용할 때 사용한다.

문법

```
<xsl:templates match="원본 XML 문서의 대상 노드 지정">
    <xsl:choose>
        <xsl:when test="조건식1">
          변환 내용1
        </xsl:when>
        <xsl:when test="조건식2">
          변환 내용2
        </xsl:when>
            :
        <xsl:otherwise>
          변환 내용
        </xsl:otherwise>
    </xsl:choose>
</xsl:apply-tempaltes>
```

참과 거짓을 돌려주는 XPath 표현식이 온다.

생략가능

- 실습 예제.

XML 문서 : c8_1102.xml
XSL 문서 : c8_1102.xsl

11. 제어 엘리먼트

11-3 xsl:for-each 엘리먼트

- for-each 엘리먼트는 대상 노드들의 숫자만큼 반복해서 변환시키는 내용이 있을 경우 사용된다.
- for-each 엘리먼트는 반드시 템플레이트 룰 내에서 작성되어야 한다.
- 템플레이트 룰은 최상위 엘리먼트이므로 XSL 문서의 루트 엘리먼트인 xsl:stylesheet의 자식 엘리먼트로만 작성되어야 한다.

문법	<xsl:templates match="원본 XML 문서의 대상 노드 지정"> <xsl:for-each select="원본 XML 문서의 대상 노드 지정"> <!-- 대상 노드를 정렬하기 --> <xsl:sort ~/> 변환 내용 기술 </xsl:for-each> </xsl:apply-tempaltes>

- xsl:for-each 엘리먼트는 자식 엘리먼트로 xsl:sort 엘리먼트를 여러 개 가질 수 있다.

- 실습 예제.

XML 문서 : c8_1103.xml
XSL 문서 : c8_1103.xsl

12. 변수와 파라메터

12-1 변수

- XSL 문서에서 변수에 해당되는 엘리먼트는 xsl:variable 이다.

- **(1) 변수 선언**

문법	<xsl:variable name="변수명" select="변수값"> <xsl:variable name="변수명">변수값</xsl:valiable>

- 변수명은 QName(접두사:변수명) 형태로 기술한다.
- 변수값은 select 속성값으로 기술할 수도 있고, 컨텐트 내용으로 기술할 수도 있다.
  - 변수값이 문자열인 경우, ' '로 감싸지 않으면 노드명으로 인식하고 노드의 컨텐트 내용을 값으로 가진다.
- 변수값이 빈 문자인 경우.

문법	<xsl:variable name="변수명"/> 또는 <xsl:variable name="변수명" select="''"/>

12. 변수와 파라메터

(2) 변수의 사용

- 변수값이 단순한 문자열이고, 리절트 트리 엘리먼트의 컨텐트 내용으로 사용될 경우에 다음과 같이 변수값을 참조한다.

문법	<엘리먼트> <xsl:value-of select="$변수명"/> </엘리먼트>

문법	속성명="{$변수명}"

문법	[노드명=$변수명] 또는 [@속성명=$변수명]

문법	<엘리먼트> <xsl:copy-of select="$변수명"/> </엘리먼트>

12. 변수와 파라메터

(3) 변수의 사용 범위

- xsl:variable 엘리먼트를 어디에 작성하느냐에 따라 사용할 수 있는 범위가 제한된다.
  - ■ 글로벌 변수 – xsl:stylesheet 엘리먼트의 자식 엘리먼트로 작성
  - ■ 로컬 변수 – 템플레이트 룰 내에서 작성

(4) 변수 사용 예제

- 실습 예제.

  XML 문서 : c8_1201.xml
  XSL 문서 : c8_1201.xsl

12. 변수와 파라메터

12-2 파라메터

- Parameter는 현재 템플레이트 룰에서 다른 템플레이트 룰을 적용할 때 필요한 정보를 전달한다.

(1) 파라메터 선언

문법	<xsl:param name="파라메터명" select="파라메터값"/>

- 파라메터명은 QName(접두사:변수명) 형태로 기술한다.
- 파라메터값이 문자열인 경우는 반드시 ' '로 감싸야 한다. 만약 ' '로 감싸지 않으면 노드명으로 인식하고 노드의 컨텐트 내용을 값으로 가진다.

- 파라메터값으로 빈 문자값 주기.

문법	<xsl:param name="변수명"/> 또는 <xsl:param name="변수명" select="''"/>

12. 변수와 파라메터

(2) 파라메터값 전달

- 템플레이트 룰에서 다른 템플레이트 룰로 파라메터값을 전달할 경우에는 다음과 같이 xsl:with-param 엘리먼트를 사용한다.

문법	<xsl:apply-templates match="대상 노드명"> <xsl:with-param name="파라메터명" select="전달할 데이터"/> </xsl:apply-templates>

문법	<xsl:call-template name="템플레이트 룰명"> <xsl:with-param name="파라메터명" select="전달할 데이터"/> </xsl:call-template>

- xsl:with-param 엘리먼트는 xsl:apply-templates 엘리먼트와 xsl:call-template 엘리먼트의 자식 엘리먼트로 작성된다.
- xsl:apply-templates 엘리먼트는 match 속성을 갖는 다른 템플레이트 룰을 적용시키는 것이고, xsl:call-template 엘리먼트는 name 속성을 갖는 다른 템플레이트 룰을 적용시키는 것이다.

12. 변수와 파라메터

(3) 파라메터값 사용

- 템플레이트 룰이 넘겨 받은 파라메터값을 사용하기 위해서는 다음과 같은 문법을 사용해야 한다.

문법	<xsl:template ~> <!-- 파라메터 선언 --> <xsl: param name="파라메터명"/> <!-- 파라메터 사용 --> 파라메터 사용 </xsl:call-template>

- 파라메터를 사용하려면 먼저 파라메터 선언을 해야한다. 파라메터 사용 방법은 변수 사용 방법과 동일하다.

(4) 파라메터 사용 예제

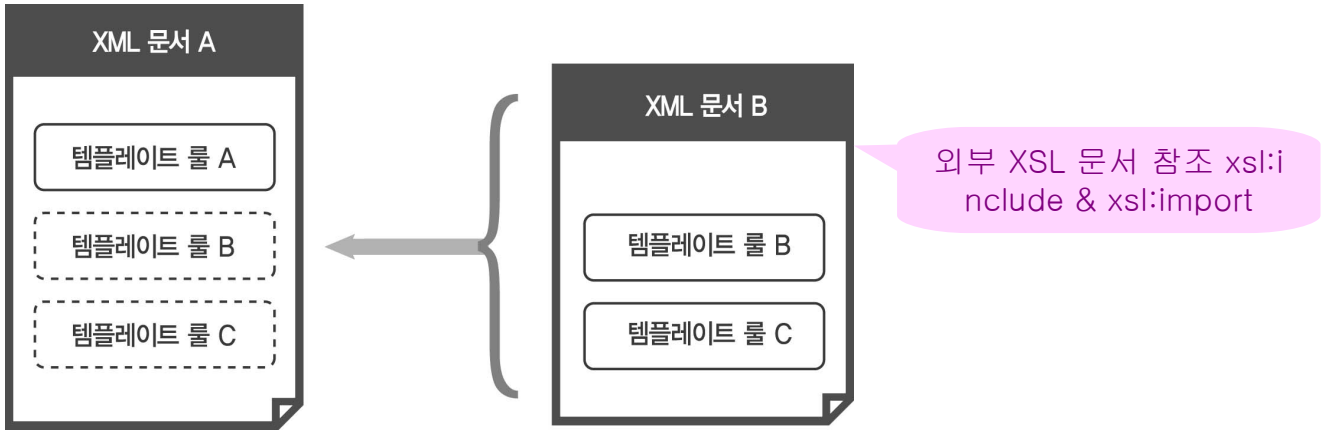
- 실습 예제.

XML 문서 : c8_1202.xml
XSL 문서 : c8_1202.xsl

13. XSL 문서의 결합

13-1 XSL 문서 결합의 의미

- XSL 문서는 다른 XSL 문서의 내용을 참조하여 사용될 수 있다. (XSL 문서의 결합)



13-2 xsl:include 엘리먼트

- xsl:import 엘리먼트는 xsl:stylesheet 엘리먼트의 자식 엘리먼트로 작성된다.

문법	<xsl:include href="참조할 외부 XSL 문서 URL 경로"/>

- 실습 예제.

XML 문서 : c8_1301.xml
XSL 문서 : c8_1301.xsl

13. XSL 문서의 결합

13-3 xsl:import 엘리먼트

- xsl:import 엘리먼트는 다른 최상위 엘리먼트보다 먼저 작성 되어야 한다.
- 외부 XSL 문서를 여러 개 참조하기 위해서, xsl:import 엘리먼트는 여러 개 작성할 수 있다.

문법	<xsl:import href="참조할 외부 XSL 문서 URL 경로"/>

- **주의 사항 :** 변환 대상 노드가 같은, 즉 macth 속성값이 같은, 템플레이트 룰이 참조하는 XSL 문서와 참조될 XSL 문서에 모두 정의되어 있을 경우에는, 참조하는 XSL 문서의 템플레이트 룰이 적용된다.
- 실습 예제.

XML 문서 : c8_1301.xml
XSL 문서 : c8_1302.xsl