

제 6 장 XML 스키마 선언

- 1 절. 스키마 언어의 탄생 배경
- 2 절. 스키마 언어 권고안
- 3 절. 스키마 문서의 물리적 구성
- 4 절. 스키마 인스턴스의 유효성 검사
- 5 절. 스키마 문서의 루트 엘리먼트
- 6 절. 주석
- 7 절. 엘리먼트 선언
- 8 절. 속성 선언
- 9 절. 데이터 타입
- 10 절. 빌트인 심플 타입
- 11 절. 사용자 정의 심플 타입
- 12 절. 콤플렉스 타입
- 13 절. 콤플렉스 타입의 확장 및 제한
- 14 절. 모델 그룹과 속성 그룹
- 15 절. 유일 제약 정의
- 16 절. 임의의 엘리먼트 및 속성 사용
- 17 절. 노테이션 선언
- 18 절. 스키마 문서의 결합
- 19 절. 네임스페이스를 갖는 스키마

1. 스키마 언어의 탄생 배경

- 1998년 1월 Microsoft를 비롯한 몇몇 업체들이 XML-Data라는 제안서를 W3C에 제출.
- 1999년 1월 Document Definition Markup Language (DDML) Specification, Version 1.0의 수정된 제안서를 Note로 인정.
- 1999년 2월 스키마 언어 개발을 위한 설계목표를 XML Schema Requirements라는 Note로 발표.
- 2001년 5월 “XML Schema 1.1”이 W3C의 권고안으로 채택.

스키마 언어 설계 목표

1. XML DTD보다 표현력이 풍부해야 한다.
2. XML로 표현되어야 한다.
3. 자기기술적(self-describing)이어야 한다.
4. XML을 채택한 많은 종류의 응용프로그램에 의해 사용 가능해야 한다.
5. 인터넷에서 바로 사용할 수 있어야 한다.
5. 상호 운용성을 위해 최적화되어야 한다.
6. 설계가 복잡하지 않아야 하며, 실행 시 시스템 자원을 너무 많이 사용하지 않아야 한다.
7. 관련된 W3C 규격(XML Information Set, Links, Namespaces, Pointers, Style and Syntax, DOM, HTML, and RDF Schema)과 조화를 이루어야 한다.

2. 스키마 언어 권고안

< XML Schema Recommendation의 구성 >

XML Schema Part 0 : Primer

- 스키마에 대한 특징을 읽기 쉽게 기술해 놓은 입문서이다.

XML Schema Part 1 : Structures

- 스키마 언어의 골격에 대한 문법을 설명해 놓은 파트이다.

XML Schema Part 2 : Datatypes

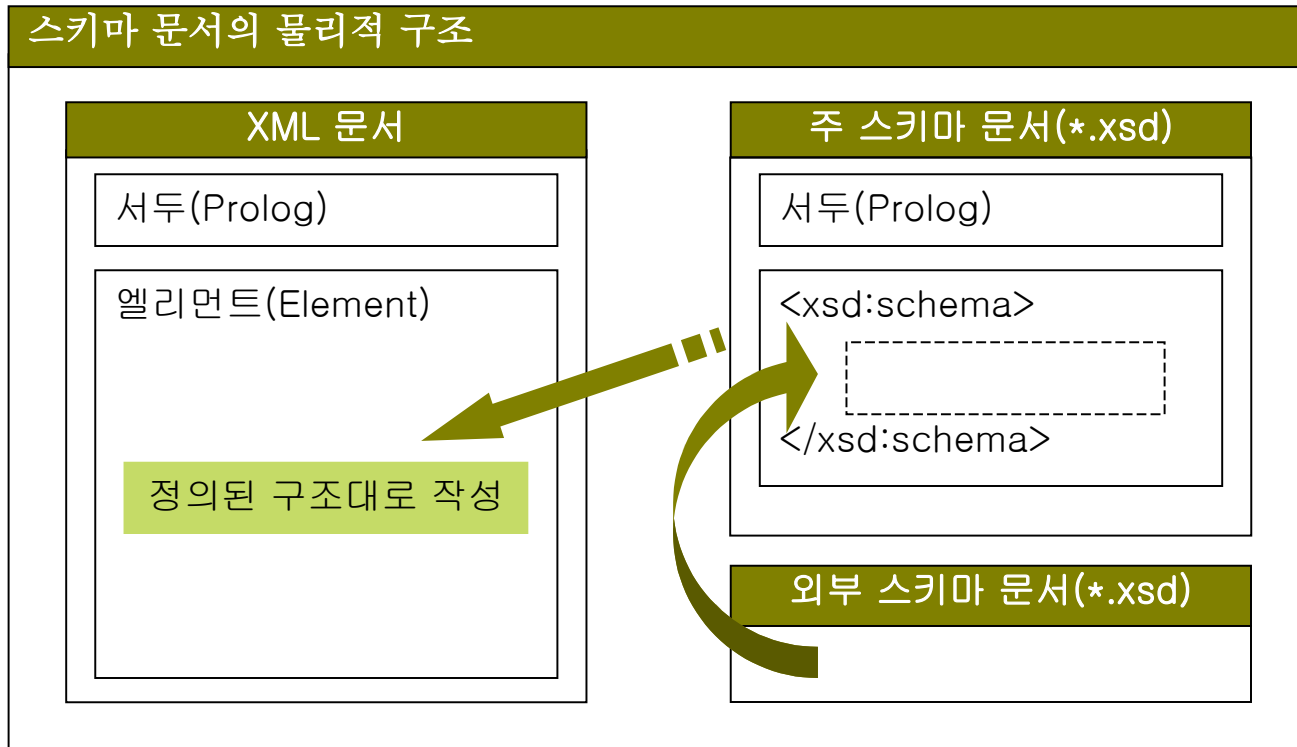
- XML 문서의 문자 데이터 및 속성값에 지정할 수 있는 데이터 타입을 설명해 놓은 파트이다.
- XML Schema 1.1 권고안 사이트

<http://www.w3.org/XML/Schema#dev>

3. 스키마 문서의 물리적 구성

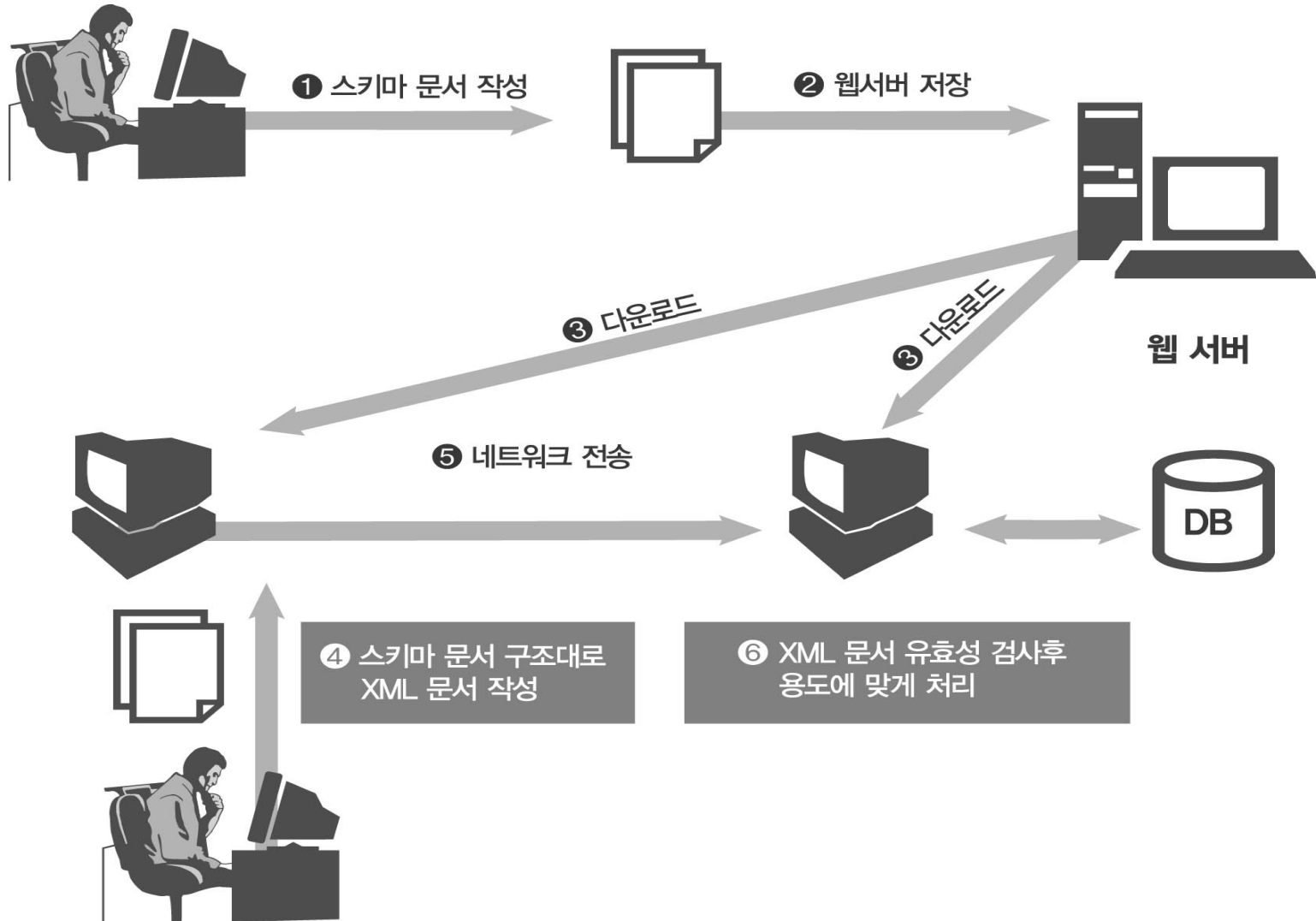
3-1. 주 스키마 문서 및 외부 스키마 문서

- 스키마 문서는 XML 문서와는 별도로 “xsd” 확장자를 가진 XML 문서로 작성된다.



3. 스키마 문서의 물리적 구성

3-2. 스키마 문서의 작성에서부터 사용까지



3. 스키마 문서의 물리적 구성

3-3. 스키마 인스턴스 지정 방법

- 스키마 인스턴스(Schema instance)란 스키마 문서에 정의된 구조대로 작성된 XML 문서이다.
- XML 문서는 특정 스키마 문서에 정의된 구조를 따르고 있다는 것을 루트 엘리먼트에 명시해야만 해당 스키마 문서의 인스턴스로 파서가 인식할 수 있고 유효성 검사도 할 수 있게 된다.

문법

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

스키마 인스턴스 네임스페이스 선언

```
<루트엘리먼트 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:noNamespaceSchemaLocation="스키마 문서 URI">
</루트엘리먼트>
```

- xsi:noNamespaceSchemaLocation 속성값은 스키마 문서의 URI 경로를 지정한다.

```
xsi:noNamespaceSchemaLocation="http://웹서버주소/경로/.../스키마문서파일명"
```

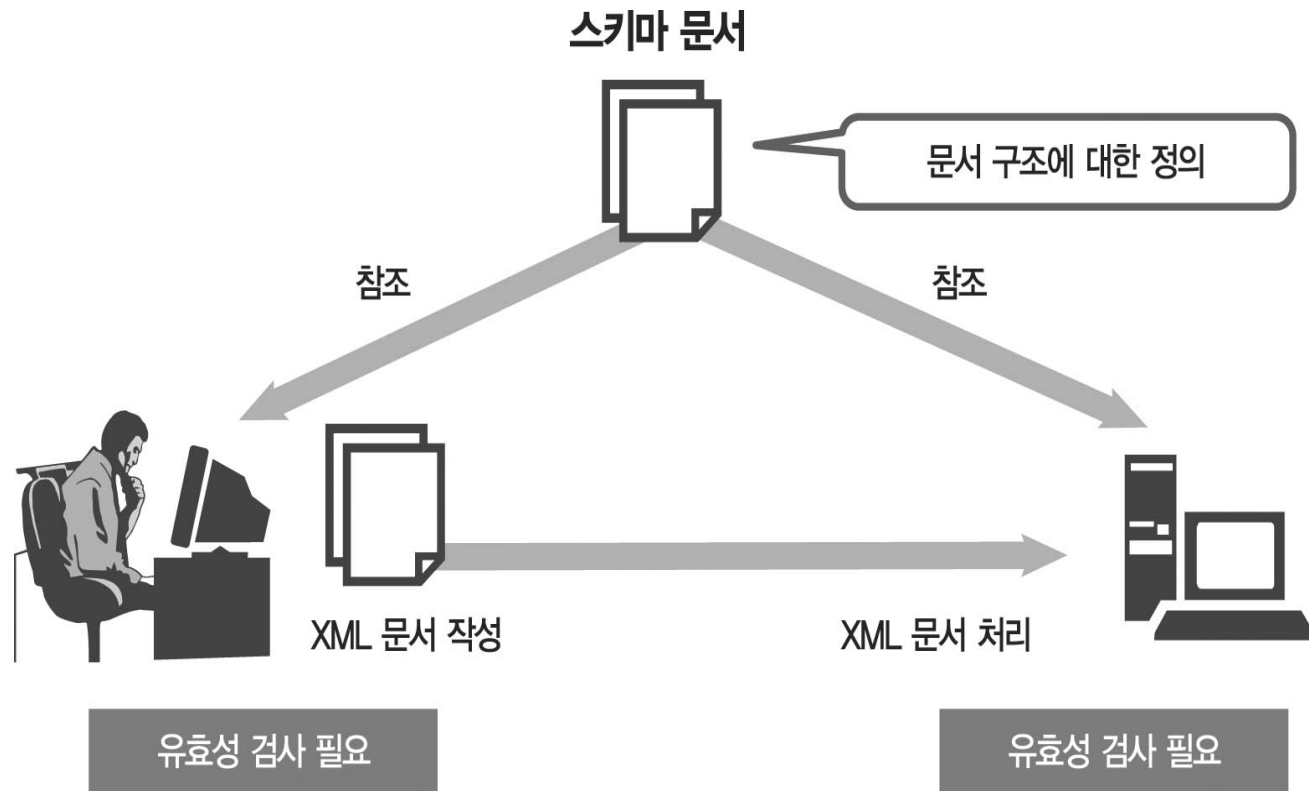
- XML문서와 스키마 문서가 동일한 로컬 하드디스크에 저장되어있을 경우.

```
xsi:noNamespaceSchemaLocation="스키마문서파일명"
```

4. 스키마 인스턴스의 유효성 검사

4-1. 유효성 검사의 필요성

- XML 문서(스키마 인스턴스)가 해당 스키마 문서에 정의되어 있는 구조대로 작성되었는지 검사하는 것을 유효성 검사라고 한다.



4. 스키마 인스턴스의 유효성 검사

4-2. XML 문서 작성시의 유효성 검사

- 보통 XML 문서를 작성하는 측은 사람 또는 응용프로그램이다. 사람이 수작업으로 XML 문서를 작성할 경우에는 반드시 유효성 검사를 해서 잘 작성되었는지 확인해야 한다.
- 유효성 검사를 하는 프로그램을 **밸리데이터**(Validator)라고 부른다.

4-3. XML 문서 처리시의 유효성 검사

- XML 문서를 처리하는 측은 응용프로그램이다. 응용프로그램은 로컬 영역에 파일 형태로 존재하는 XML 문서와 네트워크를 통해 전달되는 XML 문서를 읽고 적절히 처리하는 프로그램이다.
- 응용프로그램은 XML 문서를 처리하기 전에 반드시 유효성 검사를 해야 한다.

5. 스키마 문서의 루트 엘리먼트

5-1. 루트 엘리먼트

- 스키마 문서의 루트 엘리먼트는 “**schema**”라는 엘리먼트를 사용한다.
- 관례적으로 스키마 엘리먼트에 대한 네임스페이스 접두사는 xsd를 사용한다.

문법

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

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

자식 엘리먼트

```
</xsd:schema>
```

5. 스키마 문서의 루트 엘리먼트

5-2. 루트 엘리먼트의 자식 엘리먼트

- 루트 엘리먼트의 자식 엘리먼트들은 크게 두 개의 파트로 나뉜다.

문법	<pre><?xml version="1.0" encoding="euc-kr"?> <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"> 외부 XML Schema 문서의 참조에 관련된 엘리먼트 새로운 엘리먼트 및 속성을 선언하는 엘리먼트 </xsd:schema></pre>
----	--

- 외부 XML Schema 문서의 참조에 관련된 엘리먼트
 - include 엘리먼트
 - import 엘리먼트
 - redefine 엘리먼트
 - annotation 엘리먼트
- 새로운 엘리먼트 및 속성을 선언하는 엘리먼트
 - element 엘리먼트
 - group 엘리먼트
 - attribute 엘리먼트
 - attributeGroup 엘리먼트
 - simpleType 엘리먼트
 - notation 엘리먼트
 - complexType 엘리먼트
 - annotation 엘리먼트

6. 주석

6-1. XML 주석

- 스키마 문서도 하나의 XML 문서이므로 XML 문서 주석 표시 ‘<!-- 주석 -->’를 사용할 수 있다.

6-2. annotation 엘리먼트 사용하기

- annotation 엘리먼트로 주석을 기술하면 응용프로그램에서 스키마 문서를 처리할 때 주석의 내용을 유용하게 활용할 수 있다.

문법

```
<annotation>  
  (appinfo | documentation)*  
</annotation>
```

(1) appinfo 엘리먼트

- appinfo 엘리먼트는 XML 문서의 내용을 처리하는 응용프로그램을 위한 주석 부분으로 사용된다. source 속성에는 외부 참고 문서를 기술할 수 있지만 보통 source 속성은 생략된다.

문법

```
<appinfo source="외부참조문서경로"> 주석 내용 </appinfo>
```

6. 주석

(2) documentation 엘리먼트

- documentation 엘리먼트는 사람이 읽기 위한 서술형 주석으로 사용된다. source 속성에는 외부 참고 문서를 기술할 수 있고 xml:lang 속성에는 주석에 사용된 언어를 기술하지만 보통은 source 속성 및 xml:lang 속성은 생략된다.

문법	<pre><documentation source="외부참조문서경로" xml:lang="사용된 언어"> 주석 내용 </documentation></pre>
----	---

스키마 문서

```
<?xml version="1.0" encoding="euc-kr"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:annotation>
    <xsd:appinfo>BML 2.0</xsd:appinfo>
    <xsd:documentation>BML 1.0을 업그레이드</xsd:documentation>
  </xsd:annotation>

</xsd:schema>
```

7. 엘리먼트 선언

7-1. 데이터를 갖는 엘리먼트 선언

- XML 문서에서 엘리먼트의 내용으로 데이터만 갖게 하기 위한 선언.

문법

```
<element name="엘리먼트 이름"  
          minOccurs="횟수"  
          maxOccurs="횟수"  
          type="데이터 타입"/>
```

- name : 엘리먼트 이름(XML 권고안의 이름 작성 규칙 준수)
- minOccurs & maxOccurs : 엘리먼트의 빈도수 지정.
 - 최소수는 0 이며, 만약 생략되면 디폴트로 각각 1 값을 가진다.
- type 속성 : 데이터의 타입 기술.
- 데이터 타입
 - 빌트인 심플 타입 : 네임스페이스 선언에서 지정한 접두사 'xsd'를 붙여서 사용해야 된다.
 - 사용자정의 심플 타입

7. 엘리먼트 선언

7-1. 데이터를 갖는 엘리먼트 선언(사용 예)

①	스키마문서	<code><xsd:element name="제목" type="xsd:string"/></code>
	XML문서	<code><제목>Visual Programming</제목></code>
②	스키마문서	<code><xsd:element name="저자" maxOccurs="5" type="xsd:string"/></code>
	XML문서	<code><저자>Visual Programming</저자></code>
③	스키마문서	<code><xsd:element name="가격" type="xsd:int"/></code>
	XML문서	<code><가격>20000</가격></code>
④	스키마문서	<code><xsd:element name="책" minOccurs="0" maxOccurs="unbounded" type="xsd:string"/></code>
	XML문서	<code><책>해리포터</책></code> <code><책>사랑과 영혼</책></code>

7. 엘리먼트 선언

7-2. 자식 엘리먼트를 갖는 엘리먼트 선언

- 엘리먼트의 콘텐츠로 자식 엘리먼트를 포함하고 있는 엘리먼트를 작성하기 위한 선언.

문법	<pre><element name="엘리먼트 이름" minOccurs="횟수" maxOccurs="횟수"> <complexType> <sequence> 자식 엘리먼트 선언 </sequence> </complexType> </element></pre> <p>자식이나 속성을 갖는 엘리먼트 선언시 사용</p>
스키마 문서	<pre><xsd:element name="책" minOccurs="0" maxOccurs="unbounded"> <xsd:complexType> <xsd:sequence> <xsd:element name="제목" type="xsd:string"/> <xsd:element name="저자" type="xsd:string"/> <xsd:element name="가격" type="xsd:int"/> </xsd:sequence> </xsd:complexType> </xsd:element></pre>

7. 엘리먼트 선언

7-3. 자식 엘리먼트와 속성을 동시에 갖는 엘리먼트 선언

- 자식 엘리먼트를 가지면서 속성을 갖는 엘리먼트를 작성하기 위한 선언
 - 주의 사항** : 속성 선언은 자식 엘리먼트 선언 다음에 선언되어야 한다.

문법

```
<xsd:element name="책">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="제목" type="xsd:string"/>
      <xsd:element name="저자" type="xsd:string"/>
      <xsd:element name="가격" type="xsd:int"/>
    </xsd:sequence>

    <!-- 책 엘리먼트가 가질 속성 선언 -->
    <xsd:attribute name="종류" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

자식엘리먼트 선언

속성 선언

7. 엘리먼트 선언

7-4. 속성만 갖는 빈(empty) 엘리먼트 선언

- XML 문서에서 자식 엘리먼트는 없고 속성만 있는 엘리먼트를 빈(empty) 엘리먼트라고 한다.

스키마문서	<pre><xsd:element name="그림"> <xsd:complexType> <xsd:attribute name="소스" type="xsd:anyURI"/> </xsd:complexType> </xsd:element></pre> <div>속성 선언</div>
XML문서	<pre><그림 소스="book1.gif"/></pre>

7. 엘리먼트 선언

7-5. 데이터 및 속성을 동시에 갖는 엘리먼트 선언

- 데이터를 가지면서 속성도 갖는 엘리먼트 선언.

스키마문서	<pre><xsd:element name="가격"> <xsd:complexType> <xsd:simpleContent> <xsd:extension base="xsd:int"> <xsd:attribute name="단위" type="xsd:string"/> </xsd:extension> </xsd:simpleContent> </xsd:complexType> </xsd:element></pre>
XML문서	<pre><가격 단위="원">5000</가격></pre>

7. 엘리먼트 선언

7-5. 엘리먼트 선언의 종류

(1) 글로벌 엘리먼트 선언

- ‘schema’ 엘리먼트의 자식 엘리먼트로 엘리먼트를 선언하는 경우.
- 글로벌 엘리먼트 선언은 로컬 엘리먼트 선언시 ‘ref’ 속성에 의해 참조될 수 있다. (코드 재사용)
 - 주의 사항 : minOccurs 속성과 maxOccurs 속성은 사용할 수 없다.

스키마 문서

```
<?xml version="1.0" encoding="euc-kr"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <!-- 글로벌 엘리먼트 선언 -->
  <xsd:element name="엘리먼트명">
    ~~~
  </xsd:element>

</xsd:schema>
```

7. 엘리먼트 선언

(2) 로컬 엘리먼트 선언

- complexType 엘리먼트 내부에 엘리먼트를 선언하는 경우.
- 로컬로 선언된 엘리먼트는 다른 엘리먼트 선언시 ref 속성에 의해 참조될 수 없다.

스키마 문서

```
<?xml version="1.0" encoding="euc-kr"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="엘리먼트명">
    <xsd:complexType>
      <xsd:sequence>
        <!-- 로컬 엘리먼트 선언 -->
        <xsd:element name="엘리먼트명" type="데이터 타입"/>
        <xsd:element name="엘리먼트명" type="데이터 타입"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

</xsd:schema>
```

7. 엘리먼트 선언

7-6. 글로벌 엘리먼트 참조

- 글로벌 엘리먼트는 컴플렉스 타입 정의 및 모델 그룹 내에서 ref 속성에 의해 참조될 수 있다.

스키마문서	<pre><!-- 글로벌 엘리먼트 선언 --> <xsd:element name="책" type="xsd:string"/> <!-- 루트 엘리먼트 선언 --> <xsd:element name="책목록"> <xsd:complexType> <xsd:sequence> <!-- 글로벌 엘리먼트 참조 --> <xsd:element ref="책" maxOccurs="unbounded"/> </xsd:sequence> </xsd:complexType> </xsd:element></pre>
XML문서	<pre><책목록> <책>시인과도둑</책> <책>모바일 프로그램</책> </책목록></pre>

7. 엘리먼트 선언

7-7. null 값을 가지는 엘리먼트

- 엘리먼트의 콘텐츠를 null로 만들 때는 nil 속성을 사용하는데, 이 속성을 사용하기 위해서는 해당 엘리먼트를 선언할 때 nillable 속성값을 true로 설정해야 한다.

스키마문서	<pre><xsd:element name="book" maxOccurs="unbounded"> <xsd:complexType> <xsd:sequence> <xsd:element name="title" type="xsd:string"/> <!-- nillable 속성 사용 --> <xsd:element name="image" type="xsd:anyURI" nillable="true"/> </xsd:sequence> </xsd:complexType> </xsd:element></pre>
XML문서	<pre><book> <title>Visual C++</title> <image xsi:nil="true"/> </book></pre>

7. 엘리먼트 선언

7-8. 엘리먼트의 다형성(polymorphism)

- **다형성(polymorphism)** : 종류는 한가지지만 여러 가지의 개별적인 형태를 가지고 있는 것.
- 스키마 문서에 동일한 종류의 여러 엘리먼트들을 선언하고, XML 문서 작성시 특정 위치에 동일 종류의 엘리먼트라면 어떤 것이 작성되더라도 상관없도록 하는 것이 엘리먼트의 다형성이다.
- 엘리먼트의 다형성을 지원하기 위해 섭스티튜션그룹(**substitutionGroup**) 속성을 이용한다. 엘리먼트 선언 시 substitutionGroup 속성값으로 글로벌 엘리먼트를 지정하면, 현재 선언하는 엘리먼트가 글로벌 엘리먼트와 같은 종류의 엘리먼트로 취급된다. 이때 글로벌 엘리먼트를 섭스티튜션 헤드(substitution head)라고 부른다. 그리고 같은 섭스티튜션 헤드를 갖는 엘리먼트 그룹을 섭스티튜션 그룹이라고 부른다.
- 글로벌 엘리먼트(섭스티튜션 헤드)가 사용되는 어떤 곳이든지 해당 섭스티튜션 그룹에 속하는 엘리먼트들은 모두 사용될 수 있다.

7. 엘리먼트 선언

스키마 문서: c6_0806.xsd

```
<?xml version="1.0" encoding="euc-kr"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="author" type="xsd:string"/>
  <!-- 섭스티튜션 그룹에 속하는 엘리먼트 -->
  <xsd:element name="writer" type="xsd:string" substitutionGroup="author"/>
  <xsd:element name="booklist">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="book" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="title" type="xsd:string"/>

              <xsd:element ref="author"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

섭스티튜션 헤드

루트엘리먼트 선언

섭스티튜션 헤드 엘리먼트 사용

7. 엘리먼트 선언

XML 문서: c6_0806.xml

```
<?xml version="1.0" encoding="euc-kr"?>
<booklist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="c6_0806.xsd">

  <book>
    <title>Visual Basic</title>
    <author>이규미</author>
  </book>

  <book>
    <title>Java Programming</title>
    <writer>채규태</writer>
  </book>
</booklist>
```

8. 속성 선언

8-1. 속성 선언 문법

- 속성 선언을 하기 위해서 attribute 엘리먼트를 사용한다.

문법	<pre><attribute name="속성명" use="생략 여부" default="디폴트값" ref="참조할 글로벌 속성명" type="데이터 타입"/></pre>
----	---

- use** : 속성을 생략할 수 있는지의 여부를 결정한다.
 - 속성값으로 optional(생략 가능-default)과 required(필수)를 줄 수 있다.
- default** : 속성이 생략되었을 경우 파서에 의해 자동적으로 속성값으로 인식된다.
 - 단, use 속성값이 optional일 경우에만 적용할 수 있다.
- ref** : 글로벌 속성 선언을 참조할 경우 사용된다.
- type** : 속성값으로 가질 수 있는 데이터의 타입을 기술한다.

8. 속성 선언

8-2. 속성 선언 예제

①	스키마 문서	<code><xsd:attribute name="종류" use="required" type="xsd:string"/></code>
②	스키마 문서	<code><xsd:attribute name="단위" use="optional" default="원" type="xsd:string"/></code>
	XML 문서	<pre><책 종류="소설"> <제목>시인과도둑</제목> <가격 단위="원">9000</가격> </책> <책 종류="컴퓨터"> <제목>모바일 프로그램</제목> <가격>5000</가격> <가격 단위="원">5000</가격> </책></pre>

8. 속성 선언

8-3. 속성 선언의 종류

- 속성 선언은 위치에 따라 글로벌 속성 선언과 로컬 속성 선언으로 나누어 진다.

(1) 글로벌 속성 선언

- schema 엘리먼트의 바로 밑 자식 엘리먼트로 오는 속성 선언
- 글로벌 속성 선언은 로컬 속성 선언에서 ref 속성에 의해 참조될 수 있다.
 - 주의 사항 : 글로벌 속성 선언에서는 use 속성을 사용하면 안된다.

< 글로벌 속성 선언 위치 >

스키마
문서

```
<?xml version="1.0" encoding="euc-kr"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <!-- 글로벌 속성 선언 -->
  <xsd:attribute name="속성명" type="데이터 타입"/>

</xsd:schema>
```

8. 속성 선언

(2) 로컬 속성 선언

- **complexType** 엘리먼트의 자손 엘리먼트로 오는 속성 선언을 로컬 속성 선언이라고 부른다. 로컬 속성 선언은 다른 속성 선언에서 참조 할 수 없다.

스키마
문서

```
<?xml version="1.0" encoding="euc-kr"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="엘리먼트명">
    <xsd:complexType>
      <xsd:sequence>
        <!-- 로컬 엘리먼트 선언 -->
        <xsd:element name="엘리먼트명" type="데이터 타입"/>
        <xsd:element name="엘리먼트명" type="데이터 타입"/>
      </xsd:sequence>

      <!-- 로컬 속성 선언 -->
      <xsd:attribute name="속성명" type="데이터 타입"/>
      <xsd:attribute name="속성명" type="데이터 타입"/>

    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

9. 데이터 타입

9-1. 데이터 타입이란?

- 엘리먼트의 내용(컨텐츠) 또는 속성값으로 어떤 종류의 정보가 어떤 형태로 작성되어 질 것인지를 결정하는 것.
- 스키마 언어는 다양한 데이터 타입을 지원함으로써 XML 문서 구조를 더욱 정밀하게 정의할 수 있게 되었다. (사용자 정의 데이터 타입)

9-2. 데이터 타입의 분류

(1) 사용 용도에 따른 분류

종류	설명	용도
빌트인(built-in) 심플 타입	• 스키마 언어에 이미 정의되어 있는 심플 타입	• 데이터 값의 종류 지정 • 사용자 정의 심플타입의 베이스 데이터 타입으로 사용
사용자 정의 심플 타입	• 사용자가 새로 정의하는 심플 타입	• 데이터 값의 종류 지정
사용자 정의 컴플렉스 타입	• 사용자가 새로 정의하는 컴플렉스 타입	• 자식 엘리먼트 및 속성을 가지는 엘리먼트 선언시 사용

9. 데이터 타입

< simpleType >

- **빌트인(built-in) 심플 타입**

- 별도의 정의 없이 사용가능한 데이터 타입이다. 'xsd' 네임스페이스 접두사를 붙여 사용하며, 엘리먼트의 데이터 값의 종류 및 속성값의 종류를 지정할 때 사용.

- **사용자 정의 심플 타입**

- 빌트인 심플 타입으로 표현하기 어려운 데이터 타입을 사용자가 직접 정의해서 사용하는 데이터 타입이다.
 - 빌트인 심플 타입과는 달리 'xsd' 접두사를 붙이면 안된다.

- 속성의 속성값의 종류를 지정하는 type 속성에는 반드시 심플 타입 이름이 와야 한다.

스키마문서	<pre><xsd:attribute name="kind" type="xsd:string"/> <xsd:attribute name="tel" type="stTel"/></pre>
-------	--

- 엘리먼트의 type 속성값에 심플 타입이 사용 되면 데이터만 가지는 엘리먼트가 된다.

스키마문서	<pre><xsd:element name="title" type="xsd:string"/> <xsd:element name="tel" type="stTel"/></pre>
XML문서	<pre><title>Visual Programming</title> <tel>010-234-6789</tel></pre>

9. 데이터 타입

- **컴플렉스 타입**
 - 컴플렉스 타입은 속성이나 자식 엘리먼트를 가지는 엘리먼트를 선언할 경우에 사용된다.
- 컴플렉스 타입은 사용자가 정의해서 사용하는 사용자 정의 컴플렉스 타입만 있다.

스키마문서	<pre><xsd:element name="book"> <xsd:complexType> <xsd:sequence> <xsd:element name="title" type="xsd:string"/> <xsd:element name="author" type="xsd:string" /> <xsd:element name="price" type="xsd:int"/> </xsd:sequence> </xsd:complexType> </xsd:element></pre>
XML문서	<pre><book> <title>시인과도독</title> <author>이문열</author> <price>9000</price> </book></pre>

9. 데이터 타입

9-2. 데이터 타입의 분류

(2) 정의되는 위치에 따른 분류

- 데이터 타입이 정의되는 위치에 따라 여러 번 참조해서 사용할 수 있느냐 없느냐가 결정 된다.

종류	설명
글로벌 데이터 타입	<ul style="list-style-type: none">• 루트 엘리먼트인 schema 엘리먼트의 자식 엘리먼트로 정의된 데이터 타입• 로컬 데이터 타입 정의시 참조해서 사용할 목적
로컬 데이터 타입	<ul style="list-style-type: none">• 엘리먼트 선언 또는 속성 선언 내에서 정의된 데이터 타입• 해당 엘리먼트 선언 및 해당 속성 선언 내에서만 사용

10. 빌트인 심플 타입

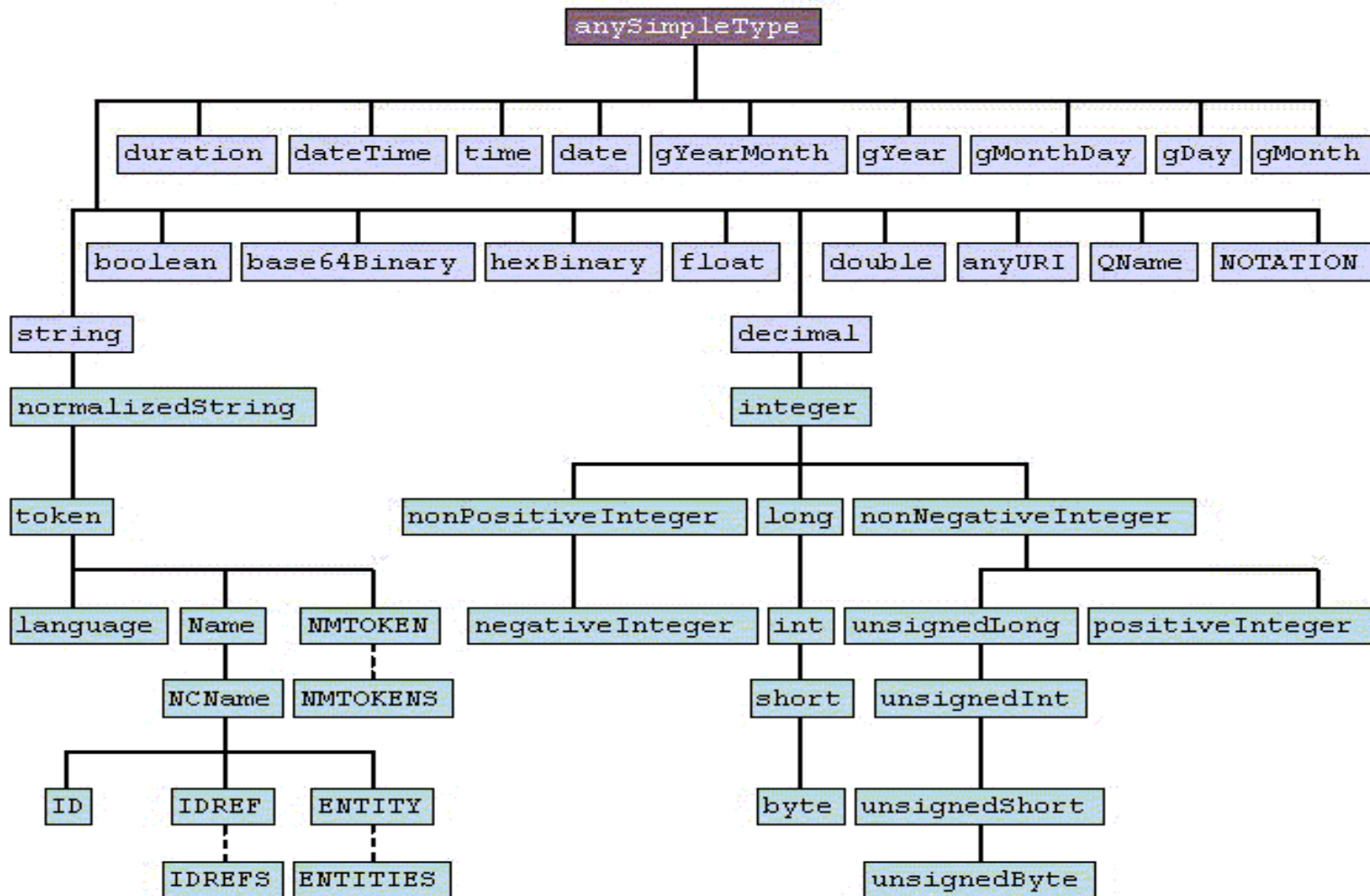
10-1. 빌트인 심플 타입 용도

- 빌트인(Built-in) 심플 타입은 스키마 언어에 이미 정의되어 있는 심플 타입이다.
- 빌트인 심플 타입은 반드시 스키마 언어의 네임스페이스 접두사(xsd)를 붙여서 사용해야 된다.
- 빌트인 심플 타입의 사용
 - 속성 선언에서 속성 값의 종류를 지정할 경우.
 - 속성과 자식 엘리먼트가 없고 데이터만 가지는 엘리먼트 선언에서 데이터 값의 종류 지정.
 - 사용자 정의 심플 타입의 베이스 타입으로 사용되는 경우.

10. 빌트인 심플 타입

10-2. 빌트인(Built-in) 심플 타입 계층도

- 빌트인 심플 타입은 기본(Primitive) 데이터 타입과 파생된(Derived) 데이터 타입으로 구분된다.



10. 빌트인 심플 타입

10-3. 기본 데이터 타입 (Primitive Datatype)

타입 이름	설명	사용예
string	문자열 데이터	서울시 역삼동
boolean	참 또는 거짓	true, false, 1, 0
decimal	유한소수	-1.23, 1266.543, +100.00, 210
float	$m * 2^e$ 로서 표현할 경우 m 범위: $ m < 2^{24}$ e 범위: $-149 \leq e \leq 104$	1, 12, 0.1234, -9.123, -1E4, 1266.43233E12, 11.78e-2, INF
double	$m * 2^e$ 로서 표현할 경우 m 범위: $ m < 2^{53}$ e 범위: $-1075 \leq e \leq 970$	1, 12, 0.1234, -9.123, -1E4, 1266.43233E12, 11.78e-2, INF
duration	기간 P: 기간(Period)의 시작 표시 nY: 해수 nM: 달수 nD: 일수 T: 일과 시의 구분자 nH: 시간 nM: 분 nS:초	P1347Y, P1347M, P1Y2MT2H, P0Y1347M, P0Y1347M0D, P1Y5M11DT16H10M30S -P120D
dateTime	CCYY-MM-DDThh:mm:ss	2003-10-09T12:10:30
time	hh:mm:ss	12:10:30
date	CCYY-MM-DD	2003-10-09

10. 빌트인 심플 타입

타입 이름	설명	사용예
gYearMonth	CCYY-MM	2003-11, 2002-05
gYear	CCYY	2003
gMonthDay	--MM-DD	--10-09
gDay	---DD	---09
gMonth	--MM--	--10--
hexBinary	0~9 범위 숫자, a~f 범위 영문 소문자 A~F 범위 영문 대문자로 이루어진 16진 데이터	1FF, 3E8, 0FB7
base64Binary	Base64 방식으로 인코딩된 이진 데이터	GpM7
anyURI	절대 또는 상대 URI	http://www.w3c.org http://www.w3c.org/#part1 /image1.gif image/background.gif
QName	“Namespaces in XML”의 문법을 따르는 XML qualified name	xsd:element, bml:book
NOTATION	속성 타입으로 바로 사용할 수 없고, 이뉴머 레이션 타입 패킷을 적용한 사용자 정의 심 플 타입을 만들어 사용	img.gif img.jpeg kind:doc

10. 빌트인 심플 타입

10-4. 파생된 데이터 타입 (Derived Datatype)

- 기본 데이터 타입(Primitive Datatype)으로부터 파생된 데이터 타입이다.

타입 이름	설명	사용예
normalizedString	캐리지 리턴(#xD), 라인 피드(#xA), 탭(#x9)을 포함하지 않는 문자열 데이터	XML과 JAVA
token	라인 피드(#xA), 탭(#x9)를 포함하지 않으며, 제일 앞과 뒤에 공백 문자열이 없고, 문자열 중간에 하나의 공백 문자(#x20)로만 이루어진 문자열	자바 XML Schema XML문법
language	RFC 1766에 따라 정의된 자연어 식별자	ko, en, fr, de, da, el, it
NMTOKEN	NMTOKEN 속성 타입	컴퓨터, 오디오
NMTOKENS	NMTOKENS 속성 타입	컴퓨터 오디오 자동차
Name	XML 권고안의 이름 정의 규칙을 따르는 이름, 첫글자는 숫자가 되어서는 안 된다.	xml_book music
NCName	“Namespaces in XML”의 문법을 따르는 NCName	math, bml

10. 빌트인 심플 타입

10-4. 파생된 데이터 타입 (Derived Datatype)-계속

기본 타입	설명	사용예
ID	유일한 값을 가질경우 사용	b1, b100, k5
IDREF	ID 값을 참조	b1, k5
IDREFS	여러 개의 ID값을 참조	b1 b2
ENTITY	ENTITY 참조	head_image
ENTITIES	여러 개의 ENTITY 참조	head_image body_image
integer	decimal 기본 데이터 타입에서 소수 부분이 없는 정수	-1, 0, 12678967, +100000
nonNegativeInteger	0과 음의 정수	0, 1, 12345
positiveInteger	양의 정수	1, 12345
nonPositiveInteger	0과 양의 정수	-1, -1267896, -100000, 0
negativeInteger	음의 정수	-1, -12678967, -100000

10. 빌트인 심플 타입

10-4. 파생된 데이터 타입 (Derived Datatype)-계속

기본 타입	설명	사용예
long	-9223372036854775808 부터 9223372036854775807 까지	-9223372036854775808, -1, 0, 1, 100
int	-2147483648 부터 2147483647 까지	-2147483648, -1, 0, 1, 100
short	-32768 부터 32767 까지	-500, -1, 0, 1, 500
byte	-128 부터 127 까지	-128, -100, 0, 100, 127
unsignedLong	0 부터 18446744073709551615 까지	0, 1, 12345, 37493730473932047
unsignedInt	0 부터 4294967295 까지	0, 1, 12345, 3456723
unsignedShort	0 부터 65535 까지	0, 1, 65535
unsignedByte	0 부터 255 까지	0, 100, 255

10. 빌트인 심플 타입

10-5. 빌트인 심플 타입 사용 방법

- 빌트인 심플 타입을 사용할 때에는 반드시 Namespace 접두사를 붙여서 사용해야 된다.

①	스키마문서	<code><xsd:element name="제목" type="xsd:string"/></code>
	XML문서	<code><제목>소설쓰는 교수</제목></code>
②	스키마문서	<code><xsd:element name="사이즈" type="xsd:double"/></code>
	XML문서	<code><사이즈>90.5</사이즈></code>
③	스키마문서	<code><xsd:element name="img"> <xsd:complexType> <xsd:attribute name="src" type="xsd:anyURI"/> </xsd:complexType> </xsd:element></code>
	XML문서	<code></code>

11. 사용자 정의 심플 타입

11-1. 사용자 정의 심플 타입 용도

- 스키마 언어에서는 빌트인 심플 타입이나 사용자 정의 심플 타입을 기반으로 새로운 사용자 정의 심플 타입(simpleType)을 정의하는 방법을 제공한다.

11-2. 사용자 정의 심플 타입 정의 문법

- 사용자 정의 심플 타입을 정의하기 위해서는 simpleType 엘리먼트를 사용한다.

(1) 글로벌 심플 타입 정의 문법

- 루트 엘리먼트인 schema 엘리먼트의 자식 엘리먼트로 정의 되는 심플 타입.
- 글로벌 심플 타입 정의는 엘리먼트 선언시나 속성 선언시에 심플 타입 이름으로 참조 적용 된다.

문법	<pre><simpleType name="심플 타입 이름"> (restriction list union) </simpleType></pre>
----	--

11. 사용자 정의 심플 타입

- 글로벌 심플 타입의 정의 위치 및 적용 방법

스키마
문서

```
<?xml version="1.0" encoding="euc-kr"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <!-- 글로벌 심플 타입 정의 -->
  <xsd:simpleType name="stTest">
    ~~~
  </xsd:simpleType>

  <!-- 글로벌 심플 타입 적용 -->
  <xsd:element name="엘리먼트명" type="stTest"/>
  <xsd:attribute name="속성명" type="stTest"/>
</xsd:schema>
```

11. 사용자 정의 심플 타입

(2) 로컬 심플 타입 정의 문법

- 엘리먼트 선언 및 속성 선언 내부에서 정의되어지는 심플 타입.
- 글로벌 심플 타입과는 달리, name 속성이 없어 다른 엘리먼트 선언 및 속성 선언에서 참조 적용될 수 없고 해당 엘리먼트 선언 및 속성 선언에서만 적용된다.

문법

```
<simpleType>  
  (restriction | list | union)  
</simpleType>
```

11. 사용자 정의 심플 타입

- 로컬 심플 타입의 정의 위치 및 적용 방법

스키마
문서

```
<?xml version="1.0" encoding="euc-kr"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="test">
    <!-- 로컬 심플 타입 정의 -->
    <xsd:simpleType>
      ~~~
    </xsd:simpleType>
  </xsd:element>

  <xsd:attribute name="속성명">
    <!-- 로컬 심플 타입 정의 -->
    <xsd:simpleType>
      ~~~
    </xsd:simpleType>
  </xsd:attribute>

</xsd:schema>
```

11. 사용자 정의 심플 타입

(3) 심플 타입 정의 시 사용되는 자식 엘리먼트

- simpleType 엘리먼트의 자식 엘리먼트로 올 수 있는 엘리먼트의 종류
- 주의 사항** : 여러 개를 동시에 기술 할 수 없다.

엘리먼트명	설명
restriction	빌트인 심플 타입 또는 이미 정의된 사용자 정의 심플 타입을 제한하여 새로운 심플 타입을 정의
list	공백 문자열로 분리된 토큰들의 리스트를 값으로 갖는 타입 정의
union	여러 개의 심플 타입을 결합하여 여러 종류의 데이터 값을 갖는 타입 정의

11-3 심플 타입을 제한하여 새로운 심플 타입 정의

- 값의 범위를 제한하기 위해서 사용되는 엘리먼트는 restriction 엘리먼트이다.

문법	<code><restriction base="기반이 되는 심플 타입 이름"></code> 패시 엘리먼트 <code></simpleType></code>
----	--

11. 사용자 정의 심플 타입

- 패싯(facet) 엘리먼트란 제한할 내용을 담고 있는 엘리먼트이다.

패싯이름	설명
minExclusive	포함이 되지 않는 하한값 지정
minInclusive	포함이 되는 하한값 지정
maxExclusive	포함이 되지 않는 상한값 지정
maxInclusive	포함이 되는 상한값 지정
totalDigits	수를 구성하는 숫자(digit)들의 개수 지정
fractionDigits	소수부를 구성하는 숫자(digit)들의 개수 지정
length	문자열의 길이 또는 리스트 항목의 개수 지정
minLength	문자열의 최소 길이 또는 리스트 항목의 최소수 지정
maxLength	문자열의 최대 길이 또는 리스트 항목의 최대수 지정
enumeration	이뉴머레이션 타입이 가질 수 있는 값 지정
pattern	문자 데이터의 포맷을 나타내는 정규식 지정

- 패싯(facet) 엘리먼트의 사용 문법.

문법	<패싯이름 value="값"/>
----	-------------------

11. 사용자 정의 심플 타입

(1) 특정 범위의 값을 가지는 심플 타입 정의

- 정수(int)를 기반으로 특정 범위를 가지는 새로운 사용자 정의 심플 타입을 정의해 보기.

스키마 문서	<pre><!-- 사용자 정의 심플 타입 정의 --> <xsd:simpleType name="stPrice"> <xsd:restriction base="xsd:int"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value="100000"/> </xsd:restriction> </xsd:simpleType> <!-- 사용자 정의 심플 타입 적용 --> <xsd:element name="price" type="stPrice"/></pre>
XML 문서	<pre><!-- 옳은 작성 방법 --> <price>20000</price> <!-- 잘못된 작성 방법 --> <price>3500000</price> <price>-10000</price></pre>

< 실습 >
C6_1201.xsd
C6_1201.xml

11. 사용자 정의 심플 타입

(2) 이뉴머레이션 값을 가지는 심플 타입 정의

- 이뉴머레이션이란 값으로 가질 수 있는 목록이다.

스키마
문서

```
<!-- 사용자 정의 심플 타입 정의 -->
<xsd:simpleType name="stUnit">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="원"/>
    <xsd:enumeration value="달러"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<!-- 사용자 정의 심플 타입 적용 -->
<xsd:element name="price">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:int">
        <xsd:attribute name="unit" type="stUnit"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

XML
문서

```
<!-- 옳은 작성 방법 -->
<price unit="원">25000</price>
```

```
<!-- 잘못된 작성 방법 -->
<price unit="마르크">25000</price>
```

< 실습 >

C6_1202.xsd
C6_1202.xml

11. 사용자 정의 심플 타입

(3) 고정된 패턴을 값으로 가지는 심플 타입 정의

- pattern 엘리먼트의 속성 value에 정규식을 기술함으로서 고정된 패턴을 만들 수 있다.

정규식	설명	가능한 값
BookWd	Wd : 0~9 임의의 숫자	Book0, Book1, ..., Book9
BookWd*	* : zero or more	Book, Book0, ..., Book100, ...
BookWd+	+ : one or more	Book0, Book1, ..., Book9
a?x	? : zero or one	x , ax
(a b)x	: 선택	ax, bx
(a b)+x	() : 블록	ax, bx, aax, abx, bbbx, ...
[ab]Wd	[ab] : a 또는 b가 와야 됨	a1, b5
Book[^0]Wd*	^0 : 0은 제외	Book1, ..., Book100, ...
Wd{3}-Wd{4}	{3} : 반드시 3개가 와야 됨	110-1111, 123-1234
Wd{2,3}-Wd{1,4}	{1,4} : 1개~4개 올 수 있음	10-111, 123-1234
Wd{2,}	{2,} : 최소 2개가 와야 됨	10, 100, 1110, ...
(ab){2}x	{2} : 반드시 2개가 와야 됨	ababx
[a-e]x	[시작문자-끝문자]	ax, bx, cx, dx, ex

11. 사용자 정의 심플 타입

- 고정된 패턴을 값으로 가지는 심플 타입 정의-예제

스키마
문서

```
<!-- 사용자 정의 심플 타입 정의 -->
<xsd:simpleType name="stSsn">
  <xsd:restriction base="xsd:string">
    <xsd:length value="14"/>
    <xsd:pattern value="Wd{6}-Wd{7}"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<!-- 사용자 정의 심플 타입 적용 -->
<xsd:element name="author">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="ssn" type="stSsn"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

XML
문서

```
<!-- 옳은 작성 방법 -->
<ssn>123455-1234567</ssn>
```

```
<!-- 잘못된 작성 방법 -->
<ssn>1234561234567</ssn>
```

11. 사용자 정의 심플 타입

(3) 사용자 정의 심플 타입을 다시 제한하여 새로운 심플 타입 정의

- 빌트인 심플 타입을 제한하여 만든 사용자 정의 심플 타입을 기반으로 하여, 이를 다시 제한해서 새로운 심플 타입을 정의할 수도 있다.

스키마
문서

```
<!-- 사용자 정의 심플 타입 정의 -->
<xsd:simpleType name="stPrice">
  <xsd:restriction base="xsd:int">
    <xsd:minInclusive value="0"/>
    <xsd:maxInclusive value="100000"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="stPrice2">
  <xsd:restriction base="stPrice">
    <xsd:minInclusive value="1000"/>
  </xsd:restriction>
</xsd:simpleType>

<!-- 사용자 정의 심플 타입 적용 -->
<xsd:element name="price" type="stPrice2"/>
```

XML
문서

```
<!-- 옳은 작성 방법 -->
<price>20000</price>
```

```
<!-- 잘못된 작성 방법 -->
<price>500</price>
```

11. 사용자 정의 심플 타입

11-4 공백으로 분리된 여러 데이터를 가지는 심플 타입 정의

- list 엘리먼트는 공백으로 분리된 여러 데이터 아이템들의 리스트를 표현한다.

문법	<code><list itemType="심플 타입 이름"/></code>
스키마문서	<pre><!-- 사용자 정의 심플 타입 정의 --> <xsd:simpleType name="stAuthor"> <xsd:list itemType="xsd:string"/> </xsd:simpleType> <!-- 사용자 정의 심플 타입 적용 --> <xsd:element name="author" type="stAuthor"/></pre>
XML문서	<code><author>신민철 채규태 이규미</author></code>

11. 사용자 정의 심플 타입

11-5 여러 개의 심플 타입을 결합한 심플 타입 정의

- union 엘리먼트는 하나 이상의 심플 타입을 결합하여 새로운 심플 타입을 정의하는 엘리먼트이다.

문법	<union memberTypes="심플타입1 심플타입2 ..."/>	
스키마 문서	<pre><!-- 사용자 정의 심플 타입 정의 --> <xsd:simpleType name="stAuthor"> <xsd:restriction base="xsd:string"> <xsd:length value="3"/> </xsd:restriction> </xsd:simpleType> <xsd:simpleType name="stAuthor2"> <xsd:restriction base="xsd:string"> <xsd:length value="4"/> </xsd:restriction> </xsd:simpleType> <xsd:simpleType name="stAuthor3"> <xsd:union memberTypes="stAuthor stAuthor2"/> </xsd:simpleType> <!-- 사용자 정의 심플 타입 적용 --> <xsd:element name="author" type="stAuthor3"/></pre>	
	XML 문서	<author>선우해경</author>

12. 콤플렉스 타입

12-1 콤플렉스 타입 용도

- 콤플렉스 타입(complexType)은 속성이나 자식 엘리먼트를 가지는 엘리먼트 선언에 필요한 타입이다.

12-2 콤플렉스 타입 문법

- 콤플렉스 타입을 정의하기 위해서는 complexType 엘리먼트를 사용한다.

(1) 글로벌 콤플렉스 타입 정의 문법

- 루트 엘리먼트인 schema 엘리먼트의 자식 엘리먼트로 콤플렉스 타입을 정의할 경우 글로벌 콤플렉스 타입 정의가 된다. 글로벌 콤플렉스 타입 정의는 엘리먼트 선언시 **type** 속성값에서 콤플렉스 타입 **name**으로 참조 적용 된다.

문법

```
<complexType name="콤플렉스 타입 이름">
    ~~~
</complexType>
```

12. 컴플렉스 타입

- 글로벌 컴플렉스 타입의 정의 위치 및 적용 방법.

스키마
문서

```
<?xml version="1.0" encoding="euc-kr"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <!-- 글로벌 컴플렉스 타입 정의 -->
  <xsd:complexType name="ctTest">
    ~~~
  </xsd:complexType>

  <!-- 글로벌 컴플렉스 타입 적용 -->
  <xsd:element name="엘리먼트명" type="ctTest"/>
```


12. 콤플렉스 타입

(2) 로컬 콤플렉스 타입 정의 문법

- 엘리먼트 선언 내부에서 정의되는 콤플렉스 타입.
- 로컬 콤플렉스 타입은 name 속성이 없으므로 다른 엘리먼트 선언에서 참조 적용될 수 없고 해당 엘리먼트 선언에만 적용된다.

문법	<pre><complexType> ~~~ </ complexType></pre>
스키마 문서	<pre><?xml version="1.0" encoding="euc-kr"?> <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <xsd:element name="test"> <!-- 로컬 콤플렉스 타입 적용 --> <xsd: complexType> ~~~ </xsd: complexType> </xsd:element></pre>

12. 컴플렉스 타입

12-3 순차적으로 자식 엘리먼트들이 오는 컴플렉스 타입 정의

- complexType의 자식 엘리먼트로 sequence 엘리먼트를 사용하여 정의한다.
- **주의 사항** : 자식 엘리먼트가 하나만 오더라도 sequence 엘리먼트를 사용해야 한다.

스키마
문서

```
<!-- 루트 엘리먼트 선언 -->
<xsd:element name="booklist">
  <!-- 로컬 컴플렉트 타입 정의 -->
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="book" type="ctBook" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!-- 글로벌 컴플렉스 타입 정의 -->
<xsd:complexType name="ctBook">
  <xsd:sequence>
    <xsd:element name="title" type="xsd:string"/>
    <xsd:element name="author" type="xsd:string"/>
    <xsd:element name="publisher" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

XML
문서

```
<book>
  <title>XML Programming</title>
  <author>신민철</author>
  <publisher>프리렉</publisher>
</book>
```

12. 컴플렉스 타입

12-4 선택적으로 자식 엘리먼트들을 사용할 수 있는 컴플렉스 타입 정의

- complexType 엘리먼트 및 sequence 엘리먼트에서 choice 엘리먼트를 사용하여 엘리먼트를 선언하게 되면 선택적으로 엘리먼트를 사용할 수 있게 된다.

문법(1)	<pre> <complexType name="컴플렉스 타입 이름"> <choice minOccurs="최소선택횟수" maxOccurs="최대선택횟수"> 엘리먼트 선언들 </choice> </complexType> </pre>	
문법(2)	<pre> <complexType name="컴플렉스 타입 이름"> <sequence> 엘리먼트 선언들 <choice minOccurs="최소선택횟수" maxOccurs="최대선택횟수"> 엘리먼트 선언들 </choice> 엘리먼트 선언들 </sequence> </complexType> </pre>	
스키마 문서	<pre> <xsd:choice maxOccurs="unbounded"> <xsd:element name="author" type="xsd:string"/> <xsd:element name="writer" type="xsd:string"/> </xsd:choice> </pre>	<div>< 실습 ></div> <div>C6_1302.xsd</div> <div>C6_1302.xml</div>

12. 컴플렉스 타입

12-5 자식 엘리먼트와 속성을 동시에 갖는 컴플렉스 타입 정의

- complexType 엘리먼트에서 attribute 엘리먼트가 선언되는 위치는 sequence 엘리먼트 또는 choice 엘리먼트 다음이다.

스키마 문서

```
<!-- 글로벌 컴플렉스 타입 정의 -->
<xsd:complexType name="ctBook">
  <xsd:sequence>
    <xsd:element name="title" type="xsd:string"/>
    <xsd:element name="author" type="xsd:string"/>
    <xsd:element name="publisher" type="xsd:string"/>
    <xsd:element name="price" type="xsd:int"/>
  </xsd:sequence>
  <!--속성 선언-->
  <xsd:attribute name="id" type="xsd:ID"/>
  <xsd:attribute name="kind" type="xsd:string"/>
</xsd:complexType>
```

```
<!-- 글로벌 컴플렉스 타입 적용 -->
<xsd:element name="book" type="ctBook" maxOccurs="unbounded"/>
```

XML 문서

```
<book id="b1" kind="computer">
  <title>XML And Java</title>
  <author>신민철</author>
  <publisher>프리렉</publisher>
  <price>25000</price>
</book>
```

12. 컴플렉스 타입

12-6 속성만 갖는 컴플렉스 타입 정의

- 빈(empty) 엘리먼트를 선언할 때에도 컴플렉스 타입이 사용된다.

문법	<pre><complexType name="컴플렉스 타입 이름"> 속성 선언들 </complexType></pre>
스키마문서	<pre><!-- 글로벌 컴플렉스 타입 정의 --> <xsd:complexType name="ctImg"> <attribute name="src" type="xsd:anyURI" use="required"/> </xsd:complexType> <!-- 글로벌 컴플렉스 타입 적용 --> <xsd:element name="img" type="ctImg" minOccurs="0"/></pre>
XML문서	<pre></pre>

12. 컴플렉스 타입

12-7 데이터 및 속성을 동시에 갖는 컴플렉스 타입 정의

- 속성은 있지만, 자식 엘리먼트 없이 심플 타입 데이터만 오는 엘리먼트 선언하기.

문법	<pre> <complexType name="컴플렉스 타입 이름"> <simpleContent base="데이터 타입"> <extension> 속성 선언 </extension> </simpleContent> </complexType> </pre>
스키마문서	<pre> <!-- 글로벌 컴플렉스 타입 정의 --> <xsd:complexType name="ctPrice"> <xsd:simpleContent> <xsd:extension base="xsd:int"> <xsd:attribute name="unit" type="xsd:string"/> </xsd:extension> </xsd:simpleContent> </xsd:complexType> <!-- 글로벌 컴플렉스 타입 적용 --> <xsd:element name="price" type="ctPrice" /> </pre>
XML문서	<pre> <price unit="원">25000</price> </pre> <p>심플 타입 콘텐츠를 갖는 컴플렉스 타입</p>

13. 컴플렉스 타입의 확장 및 제한

13-1 컴플렉스 타입의 확장 및 제한의 의미

- 컴플렉스 타입은 다른 컴플렉스 타입을 확장 및 제한할 수 있도록 상속의 개념을 적용할 수 있을 뿐만 아니라, 상속된 타입의 다형성(polymorphism)을 이용할 수 있는 방법도 제공하고 있다.
 - **컴플렉스 타입의 확장** : 기반이 되는 컴플렉스 타입에 속성이나 엘리먼트를 추가하는 것.
 - **컴플렉스 타입의 제한** : 기반이 되는 컴플렉스 데이터값 및 속성값의 범위를 제한하거나 엘리먼트 일부를 제거하는 것.

13-2 심플 타입 콘텐츠를 갖는 컴플렉스 타입을 확장

- 심플 타입 콘텐츠를 갖는 컴플렉스 타입은 데이터와 속성을 동시에 갖는 컴플렉스 타입을 말한다.
 - **심플 타입 콘텐츠를 갖는 컴플렉스 타입의 확장** : 속성을 추가하여 새로운 심플 타입 콘텐츠를 갖는 컴플렉스 타입을 정의하는 것.

13. 컴플렉스 타입의 확장 및 제한

- 심플 타입 콘텐츠를 갖는 컴플렉스 타입을 확장하는 컴플렉스 타입을 정의하는 문법.

문법

```
<xsd:complexType name="컴플렉스 타입 이름">  
  <xsd:simpleContent>  
    <xsd:extension base="확장할 컴플렉스 타입 이름">  
      엘리먼트 선언들  
    </xsd:extension>  
  </xsd:simpleContent>  
</xsd:complexType>
```

- 주의 사항 :** base 속성값으로 심플 타입 콘텐츠를 갖지 않는, 즉 자식 엘리먼트나 속성만 가지고 있는 컴플렉스 타입 이름은 올 수 없다.

13. 컴플렉스 타입의 확장 및 제한

- 심플 타입 콘텐츠를 갖는 컴플렉스 타입을 확장하는 컴플렉스 타입 예제.

스키마 문서	<pre><!-- 글로벌 컴플렉스 타입 정의 --> <xsd:complexType name="ctPrice2"> <!-- 심플 콘텐츠를 갖는 컴플렉스 타입 --> <xsd:simpleContent> <!-- 심플 콘텐츠를 갖는 컴플렉스 타입인 ctPrice를 확장 --> <xsd:extension base="ctPrice"> <xsd:attribute name="card" type="xsd:boolean"/> </xsd:extension> </xsd:simpleContent> </xsd:complexType> <!-- 글로벌 컴플렉스 타입 적용 --> <xsd:element name="price" type="ctPrice2"/></pre>
XML 문서	<pre><price unit="원" card="true">25000</price></pre>

13. 컴플렉스 타입의 확장 및 제한

13-3 심플 타입 콘텐츠를 갖는 컴플렉스 타입을 제한

- simpleContent 엘리먼트의 자식 엘리먼트인 restriction 엘리먼트의 base 속성값에 심플 콘텐츠를 갖는 컴플렉스 타입의 이름이 올 수 있다.
- 이 경우 해당 컴플렉스 타입을 제한하는 새로운 심플 타입 콘텐츠를 갖는 컴플렉스 타입을 만들 수 있다.
- 주로 제한되는 내용은 기반이 되는 컴플렉스 타입의 데이터값의 범위이다.

문법

```
<xsd:complexType name="컴플렉스 타입 이름">  
  <xsd:simpleContent>  
    <xsd:restriction base="제한할 컴플렉스 타입 이름">  
      패킷 엘리먼트들  
    </xsd:restriction>  
  </xsd:simpleContent>  
</xsd:complexType>
```

13. 컴플렉스 타입의 확장 및 제한

- restriction 엘리먼트의 base 속성값으로 심플 타입 콘텐츠를 갖지 않는, 즉 자식 엘리먼트나 속성만 가지고 있는 컴플렉스 타입은 올 수가 없다.
- 패싯(facet) 엘리먼트란 제한할 내용을 담고 있는 엘리먼트를 뜻한다.

< 패싯 엘리먼트로 올 수 있는 것 >

패싯이름	설명
minExclusive	포함이 되지 않는 하한값 지정
minInclusive	포함이 되는 하한값 지정
maxExclusive	포함이 되지 않는 상한값 지정
maxInclusive	포함이 되는 상한값 지정
totalDigits	수를 이루는 숫자(digit)의 수 지정
fractionDigits	소수부를 이루는 숫자(digit)의 수 지정
length	문자열의 길이 또는 리스트 항목의 수 지정
minLength	문자열의 최소 길이 또는 리스트 항목의 최소수 지정
maxLength	문자열의 최대 길이 또는 리스트 항목의 최대수 지정
enumeration	이뉴머레이션 타입이 가질 수 있는 값 지정
pattern	문자 데이터의 포맷을 나타내는 정규식 지정

13. 컴플렉스 타입의 확장 및 제한

- 패킷 엘리먼트의 사용 문법 및 사용 예

문법	<패킷이름 value="값"/>
스키마문서	<pre><!-- 글로벌 컴플렉스 타입 정의 --> <xsd:complexType name="ctPrice2"> <!-- 심플 콘텐츠를 갖는 컴플렉스 타입 --> <xsd:simpleContent> <!-- 심플 콘텐츠를 갖는 컴플렉스 타입인 ctPrice를 제한 --> <xsd:restriction base="ctPrice"> <xsd:minInclusive value="1000"/> <xsd:maxInclusive value="100000"/> </xsd:restriction> </xsd:simpleContent> </xsd:complexType> <!-- 글로벌 컴플렉스 타입 적용 --> <xsd:element name="price" type="ctPrice2"/></pre>
XML문서	<pre><price unit="원" card="true">100000</price></pre>

13. 컴플렉스 타입의 확장 및 제한

13-4 자식 엘리먼트와 속성을 갖는 컴플렉스 타입의 확장

- 자식 엘리먼트와 속성을 가지고 있는 기존의 컴플렉스 타입에 자식 엘리먼트들과 속성들을 추가하여 새로운 컴플렉스 타입을 정의할 수 있다.

문법

```
<xsd:complexType name="컴플렉스 타입 이름">  
  <xsd:complexContent>  
    <xsd:extension base="확장할 컴플렉스 타입 이름">  
      <xsd:sequence>  
        추가할 엘리먼트 선언들  
      </xsd:sequence>  
      추가할 속성 선언들  
    </xsd:extension>  
  </xsd:complexContent>  
</xsd:complexType>
```

- 주의 사항 :** base 속성값으로 심플 타입 콘텐츠를 갖지 않는, 즉 자식 엘리먼트 및 속성만 가지고 있는 컴플렉스 타입 이름만 올 수 있다.

13. 컴플렉스 타입의 확장 및 제한

- 기존의 컴플렉스 타입에 속성 및 자식 엘리먼트를 추가 확장하는 새로운 컴플렉스 타입 정의 예.

스키마
문서

```
<!-- 글로벌 컴플렉스 타입 정의 -->
<xsd:complexType name="ctBook2">
  <xsd:complexContent>
    <xsd:extension base="ctBook">
      <xsd:sequence>
        <xsd:element name="publisher" type="xsd:string"/>
        <xsd:element name="price" type="xsd:int"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:ID"/>
      <xsd:attribute name="kind" type="xsd:string"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

XML
문서

```
<book id="b1" kind="컴퓨터">
  <title>XML And Java</title>
  <author>신민철</author>
  <publisher>프리렉</publisher>
  <price>25000</price>
</book>
```

```
<!-- 글로벌 컴플렉스 타입 적용 -->
<xsd:element name="book" type="ctBook2" maxOccurs="unbounded"/>
```

13. 컴플렉스 타입의 확장 및 제한

13-5 자식 엘리먼트와 속성을 갖는 컴플렉스 타입의 제한

- 자식 엘리먼트의 갯수를 줄인다거나, 속성값을 제한하여 사용할 경우.

문법

```
<xsd:complexType name="컴플렉스 타입 이름">  
  <xsd:complexContent>  
    <xsd:extension base="제한할 컴플렉스 타입 이름">
```

```
      <xsd:restriction>
```

제한할 컴플렉스 타입이 가지고 있는 엘리먼트 중에
실제 사용할 엘리먼트만 재 선언

```
    </xsd:restriction>
```

제한할 컴플렉스 타입에서 속성값의 범위를 제한하는 속성만 선언

```
      </xsd:extension>
```

```
    </xsd:complexContent>
```

```
</xsd:complexType>
```

- 주의 사항(1)** : base 속성값으로 심플 타입 콘텐츠를 갖지 않는, 즉 자식 엘리먼트 및 속성만 가지고 있는 컴플렉스 타입 이름만 올 수 있다.
- 주의 사항(2)** : 컴플렉스 타입을 제한하고자 할 경우, 제한하고자 하는 엘리먼트를 제외한 나머지 엘리먼트들을 모두 기술하여야 한다. 또한 새로운 엘리먼트가 아닌, 컴플렉스 타입이 가지고 있는 엘리먼트가 와야 한다. 속성값의 제한은 적은 범위로의 제한만 가능하다.

13. 컴플렉스 타입의 확장 및 제한

스키마문서	<pre><!-- 글로벌 컴플렉스 타입 정의 --> <xsd:complexType name="ctBook2"> <xsd:complexContent> <xsd:restriction base="ctBook"> <xsd:sequence> <xsd:element name="title" type="xsd:string"/> <xsd:element name="author" type="xsd:string"/> </xsd:sequence> <xsd:attribute name="kind"> <xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:enumeration value="컴퓨터"/> <xsd:enumeration value="소설"/> </xsd:restriction> </xsd:simpleType> </xsd:attribute> </xsd:restriction> </xsd:complexContent> </xsd:complexType> <!-- 글로벌 컴플렉스 타입 적용 --> <xsd:element name="book" type="ctBook2" maxOccurs="unbounded"/></pre>
XML문서	<pre><xsd:element name="book" type="ctBook2" maxOccurs="unbounded"/></pre>

13. 컴플렉스 타입의 확장 및 제한

13-6 컴플렉스 타입의 다형성(polymorphism)

- 다형성이란 동일 종류지만 여러 가지의 개별적인 형태를 가지고 있는 것을 말한다. XML Schema에서는 엘리먼트의 다형성도 지원하지만 컴플렉스 타입의 다형성도 지원한다.
- 엘리먼트 선언시 type 속성에 특정 종류의 컴플렉스 타입이 온다는 것만 지정해 두면, 실제 XML 문서 작성시, 해당 컴플렉스 타입의 형태는 물론이고, 해당 컴플렉스 타입을 확장한 다른 컴플렉스 타입의 형태도 올 수 있다는 것을 의미한다.

문법	<엘리먼트 이름 xsi:type="컴플렉스 타입 이름">
스키마 문서	<!-- 글로벌 컴플렉스 타입 적용 --> <xsd:element name="book" type="ctBook" maxOccurs="unbounded"/>
XML 문서	<book id="b1" xsi:type="ctBook2"> <title>사랑과 전쟁</title> <author>이사랑</author> <publisher>사랑문화사</publisher> </book>

XML 문서에서 어떤 타입을
적용할지 결정한다.

< 실습 >
C6_1405.xsd
C6_1405.xml

14. 모델 그룹과 속성 그룹

14-1 모델 그룹과 속성 그룹의 필요성

- 스키마 문서에서 여러 가지 컴플렉스 타입 정의시 중복해서 들어가는 엘리먼트나 속성선언들이 있을 수 있다.
 - 모델 그룹(model group)** : 중복되는 엘리먼트 선언들을 모아서 만든 그룹.
 - 속성 그룹(attribute group)** : 중복되는 속성들을 모아서 만든 그룹.
- 정의의 재사용으로 인해 스키마 문서 내용도 짧아지고, 단순한 구조로 인해 가독성이 좋아 진다.

14-2 모델 그룹

- 모델 그룹(model group)이란 순서와 반복 횟수가 지정된 엘리먼트들의 그룹이다.
- 모델 그룹은 글로벌 위치에서 정의되고, 컴플렉스 타입 정의시 참조된다.
- 모델 그룹의 정의 및 참조는 group 엘리먼트를 사용한다.

문법
(정의)

```
<group name="그룹이름">  
  (all | choice | sequence)  
</group>
```

문법
(참조)

```
<group ref="참조할 그룹이름"/>
```

14. 모델 그룹과 속성 그룹

- 모델 그룹 정의 예

스키마
문서

```
<!-- 모델 그룹 정의 -->
<xsd:group name="gTel">
  <xsd:sequence>
    <xsd:element name="home" type="xsd:string" minOccurs="0"/>
    <xsd:element name="office" type="xsd:string" minOccurs="0"/>
    <xsd:element name="mobile" type="xsd:string" minOccurs="0"/>
  </xsd:sequence>
</xsd:group>

<!-- 모델 그룹 참조 -->
<xsd:complexType name="ctPhone">
  <xsd:group ref="gTel"/>
</xsd:complexType>
```

14. 모델 그룹과 속성 그룹

14-3 속성 그룹

- 속성 그룹(attribute group)이란 속성 선언들의 모임을 말한다.
- 글로벌 위치에서 정의되고, 컴플렉스 타입 정의시 참조 된다.
- 속성 그룹의 정의와 참조는 attributeGroup 엘리먼트를 사용한다.

문법
(정의)

```
<attributeGroup name="속성 그룹 이름">  
  속성 선언들  
</attributeGroup>
```

문법
(참조)

```
<attributeGroup ref="참조할 속성 그룹 이름"/>
```

14. 모델 그룹과 속성 그룹


- 속성그룹 정의 예

스키마
문서

```
<!-- 속성 그룹 정의 -->
<xsd:attributeGroup name="gInfo">
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="tel" type="xsd:string"/>
</xsd:attributeGroup>

<!-- 속성 그룹 참조 -->
<xsd:complexType name="ctAuthor">
  <xsd:attributeGroup ref="gInfo"/>
</xsd:complexType>

<xsd:complexType name="ctPublisher">
  <xsd:sequence>
    <xsd:element name="address" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="gInfo"/>
</xsd:complexType>
```



15. 유일 제약 정의

14-1 유일 제약이란?

- 중복된 값을 가지지 않도록 제한하는 것.
예) 주민등록번호와 같이 유일한 값을 가지는 곳에 유일 제약을 걸어두면 중복된 데이터가 입력되는 것을 방지할 수 있다.
- DTD 내에서 ID 속성 타입으로 유일 제약을 만들수 있다.
- 스키마에서는 ID 속성 타입 뿐만 아니라 **key** 및 **unique** 엘리먼트를 사용해서 유일 제약을 정의할 수 있다.
- **ID** 속성 타입으로 지정된 속성은 **IDREF** 속성 타입으로 지정된 속성에서 참조 가능하다. 그리고 **key** 엘리먼트로 지정된 속성은 **keyref** 엘리먼트로 지정된 속성에서 참조 가능하다.

15. 유일 제약 정의

14-2 ID 속성 타입을 이용한 유일 제약 구현

- ID 속성 타입은 빌트인 심플 타입으로 별도의 정의 없이 사용될 수 있다. ID 속성 타입을 사용하면 XML 문서 전체에서 유일한 값을 가지도록 속성을 만들 수 있다.

스키마 문서	<pre><!-- ID 타입 지정 --> <xsd:attribute name="id" type="xsd:ID" use="required"/></pre>
XML 문서	<pre><kind id="k1">컴퓨터</kind> <kind id="k2">소설</kind> <kind id="k3">잡지</kind> <book id="b1">XML Fundamentals</book> <book id="b2">.NET And XML</book> <book id="b3">Java And XML</book></pre> <p>문자로 시작해야 하며, 공백을 포함할 수 없다.</p>

15. 유일 제약 정의

- ID 속성 타입으로 선언된 속성값은 IDREF 속성 타입으로 선언된 속성에서 참조할 수 있다.

스키마 문서	<code><!-- IDREF 타입 지정 --></code> <code><xsd:attribute name="kind" type="xsd:IDREF" use="required"/></code>
XML 문서	<code><book id="b1" kind="k1">XML 기초서</book></code> <code><book id="b2" kind="k2">.NET And XML</book></code> <code><book id="b3" kind="k3">Java And XML</book></code>

- IDREF 속성 타입은 ID 속성 타입으로 지정된 속성값을 모두 가질 수 있다.

15. 유일 제약 정의

14-3 key 엘리먼트를 이용한 유일 제약 구현

- 유일 제약이 적용될 대상 엘리먼트를 먼저 결정하고, 그 다음 실제로 유일 제약이 적용될 부분을 언급해야 된다. 이 때 유일 제약이 적용될 부분은 보통 대상 엘리먼트의 속성명이 온다.

문법

```
<key name="키 이름">  
  <selector xpath="대상 엘리먼트의 xpath 경로"/>  
  <field xpath="적용할 속성 이름"/>  
</key>
```

- **주의 사항 :** key 엘리먼트의 name 속성은 필수 속성이므로 생략하면 안된다. 키 이름은 keyref 엘리먼트에서 참조 할 경우 사용된다.
- selector 엘리먼트의 xpath 속성값으로는 대상이 되는 엘리먼트의 문서 내 경로를 xpath 표현식으로 나타내면 된다. 보통 key 정의는 루트 엘리먼트 선언 내에서 만들어 지므로 xpath 속성값은 루트 엘리먼트에서 부터의 상대 경로로 표현하면 된다.

15. 유일 제약 정의

- key 엘리먼트를 이용한 유일 제약 구현 예제

문법	<code><selector xpath="자식엘리먼트/자손엘리먼트/.../대상 엘리먼트"></code> or <code><selector xpath="./자식엘리먼트/자손엘리먼트/.../대상 엘리먼트"></code>
스키마 문서	<code><selector xpath="kinds/kind"></code> <code><selector xpath="./kinds/kind"></code> <code><selector xpath="book"/></code> <code><selector xpath="./book"/></code>

문법	<code><field xpath="@대상엘리먼트의 속성명"/></code> or <code><field xpath="./@대상엘리먼트의 속성명"/></code>
스키마 문서	<code><field xpath="@id"/></code> <code><field xpath="./@kind"/></code>

15. 유일 제약 정의

- key 엘리먼트로 지정된 유일 제약이 적용되는 값은 keyref 엘리먼트로 지정된 부분에서 참조 가능하다.

스키마
문서

```
<xsd:key name="keyKind">
  <!-- 상대 경로를 이용한 대상 엘리먼트 지정 -->
  <xsd:selector xpath="kinds/kind"/>
  <!-- 유일 제약이 적용될 대상 엘리먼트의 부분 -->
  <xsd:field xpath="@id"/>
</xsd:key>

<xsd:key name="keyBook">
  <!-- 상대 경로를 이용한 대상 엘리먼트 지정 -->
  <xsd:selector xpath="./book"/>
  <!-- 유일 제약이 적용될 대상 엘리먼트의 부분 -->
  <xsd:field xpath="./@id"/>
</xsd:key>
```

15. 유일 제약 정의

- keyref가 IDREF 속성 타입과 다른 점
 - keyref 엘리먼트에 의해 속성이 지정이 되면 참조하는 해당 key 속성값만 올 수 있고, 다른 key 속성값은 올 수 없다.

문법	<pre><keyref name="키참조 이름" refer="참조할 키 이름"> <selector xpath="대상 엘리먼트의 xpath 경로"/> <field xpath="적용할 속성 이름"/> </keyref></pre>
----	---

- keyref 엘리먼트의 name 속성 및 refer 속성은 필수 속성이므로 생략하면 안된다.

스키마 문서	<pre><xsd:keyref name="keyrefKind" refer="keyKind"> <!-- 대상 엘리먼트 지정 --> <xsd:selector xpath="book"/> <!-- 참조할 대상 엘리먼트의 속성 --> <xsd:field xpath="@kind"/> </xsd:keyref></pre>
-----------	--

< 실습 >

C6_1604.xsd
C6_1604.xml

15. 유일 제약 정의

14-4 unique 엘리먼트를 이용한 유일 제약 구현

- 속성값이나 엘리먼트 콘텐츠 데이터 값의 유일성을 보장할 필요성만 있고, 그 값을 참조할 필요가 없을 경우 사용한다.

문법

```
<unique name="유일제약 이름">  
  <selector xpath="대상 엘리먼트의 xpath 경로"/>  
  <field xpath="적용할 속성 이름"/>  
</unique>
```

- unique 엘리먼트의 name 속성은 필수 속성이므로 생략하면 안된다.

스키마
문서

```
<xsd:unique name="uniqueBook">  
  <xsd:selector xpath="book"/>  
  <xsd:field xpath="@id"/>  
</xsd:unique>
```

16. 임의의 엘리먼트 및 속성 사용

17. 노테이션 선언

17-1 노테이션이란?

- 그림 파일이나 동영상 파일, 음악 파일의 포맷을 식별하기 위해서 사용되어지는 특별한 엘리먼트이다.
- XML 파서가 해석할 수 없는 비 문자 데이터가 어떤 종류의 포맷을 가지고 있고, 이 데이터를 처리할 수 있는 헬퍼 프로그램은 어떤 것인지를 응용프로그램에게 알려준다.

17-2 노테이션 선언 문법

문법

```
<notation  
  name="노테이션 이름"  
  public="공개 식별자(보통 MIME 타입을 지정)"  
  system="헬퍼 프로그램"/>
```

- **주의 사항 :** 선언된 노테이션을 속성값 또는 데이터 값으로 사용할 때는 반드시 빌트인 심플 타입인 노테이션을 제한하는 사용자 정의 심플 타입을 만들어서 사용해야 된다.

17. 노테이션 선언

- Notation 예제

스키마
문서

<!-- NOTATION 선언 -->

<xsd:notation name="bmp" public="image/bmp" system="mspaint.exe"/>

<xsd:notation name="gif" public="image/gif" system="photoshop.exe"/>

<xsd:notation name="jpeg" public="image/jpeg" system="photoshop.exe"/>

<!-- 글로벌 심플 타입 정의 -->

<xsd:simpleType name="stType">

 <xsd:restriction base="xsd:NOTATION">

 <xsd:enumeration value="bmp"/>

 <xsd:enumeration value="gif"/>

 <xsd:enumeration value="jpeg"/>

 </xsd:restriction>

</xsd:simpleType>

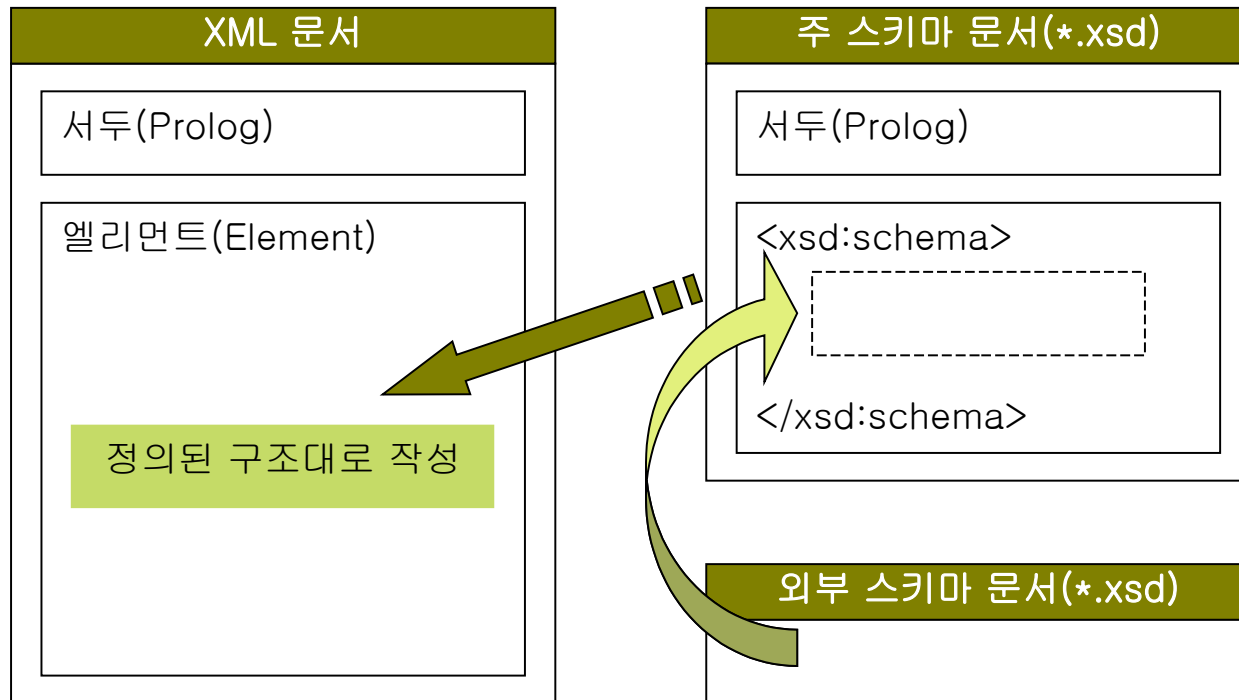
<!-- 글로벌 심플 타입 적용 -->

<xsd:attribute name="type" type="stType"/>

18. 스키마 문서의 결합

18-1 스키마 문서의 결합이란?

- 공통적으로 사용되는 정의 및 선언들을 별도의 스키마 문서에서 작성하고, 주 스키마 문서에서는 같은 내용이 작성될 곳에 중복 작성을 하지 않고 참조를 통해서 동일한 효과를 거둘 수 있다.



18. 스키마 문서의 결합

18-2 같은 네임스페이스를 가지는 스키마 문서들의 결합

- 네임스페이스가 같은 스키마 문서 또는 네임스페이스가 없는 스키마 문서를 결합하기 위해서 사용되는 엘리먼트는 **include** 엘리먼트 이다.

문법	<code><xsd:include schemaLocation="포함시킬 스키마 문서"/></code>
----	--

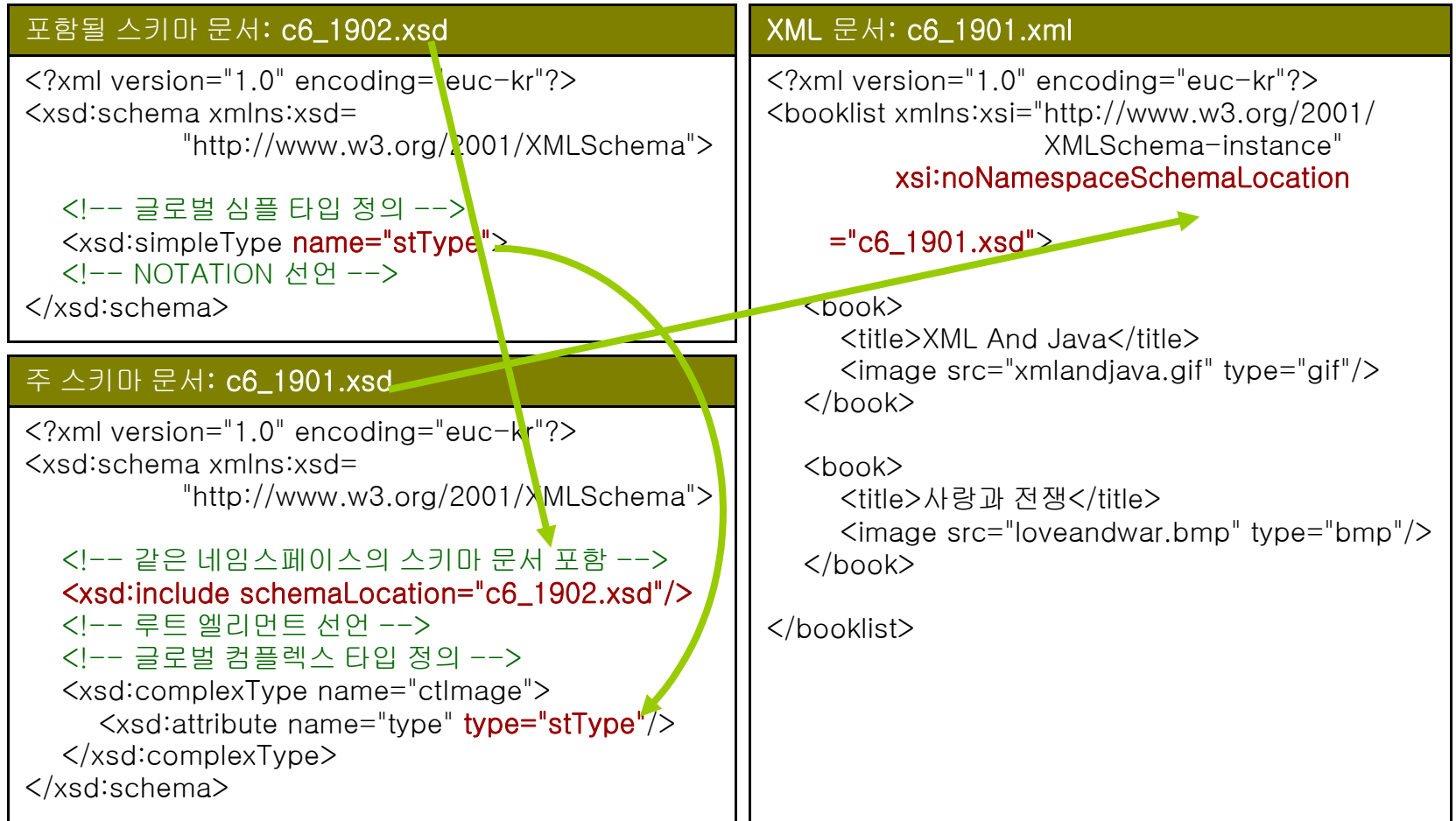
- 포함시킬 스키마 문서는 anyURI 타입으로 기술하면 된다.

<code>schemaLocation="~.xsd"</code> <code>schemaLocation="/경로명/.../~.xsd"</code> <code>schemaLocation="http://웹서버주소/경로명/.../~.xsd"</code>

- include 엘리먼트는 글로벌 엘리먼트 선언이나 속성선언보다도 먼저 작성되어야 한다.

18. 스키마 문서의 결합

- 같은 네임스페이스를 가지는 스키마 문서들의 결합-예제



18. 스키마 문서의 결합

18-3 다른 네임스페이스를 가지는 스키마 문서들의 결합

- 네임스페이스가 다른 스키마 문서들을 결합하기 위해서 사용되는 엘리먼트는 **import** 엘리먼트이다. 다음은 import 엘리먼트의 작성 문법이다.

문법

```
<xsd:import namespace="포함할 스키마 문서의 네임스페이스"  
             schemaLocation="포함할 스키마 문서"/>
```

- namespace 속성에는 참조할 스키마 문서의 네임스페이스 이름을 지정한다.
- schemaLocation 속성에는 포함시킬 스키마 문서에 대한 anyURI 타입 경로명을 기술한다.

```
schemaLocation="~.xsd"
```

```
schemaLocation="/경로명/.../~.xsd"
```

```
schemaLocation="http://웹서버주소/경로명/.../~.xsd"
```

- import 엘리먼트는 글로벌 엘리먼트 선언이나 속성선언 보다도 먼저 작성되어야 한다.

18. 스키마 문서의 결합

- 다른 네임스페이스를 가지는 스키마 문서들의 결합-예제

포함될 스키마 문서: c6_1904.xsd

```
<?xml version="1.0" encoding="euc-kr"?>
<xsd:schema ~ ~ ~
    xmlns="http://www.minchoel.com/impl">

    <!-- 글로벌 심플 타입 정의 -->
    <xsd:simpleType name="stType">

    <!-- NOTATION 선언 -->
    ~ ~ ~
</xsd:schema>
```

주 스키마 문서: c6_1903.xsd

```
<!-- 다른 네임스페이스의 스키마 문서 참조 -->
<xsd:import
    namespace="http://www.minchoel.com/impl"
    schemaLocation="c6_1904.xsd"/>

<!-- 글로벌 콤플렉스 타입 정의 -->
<xsd:complexType name="ctlImage">
    <xsd:attribute name="type" type="stType"/>
</xsd:complexType>
```

XML 문서: c6_1902.xml

```
<?xml version="1.0" encoding="euc-kr"?>
<booklist xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance"
    xsi:noNamespaceSchemaLocation=
        "c6_1903.xsd"
    xmlns:impl="http://www.minchoel.com/impl">
    <book>
        <title>XML And Java</title>
        <author>
            <impl:name>신민철</impl:name>
            <impl:ssn>123455-1234567</impl:ssn>
            <impl:tel>02-234-6789</impl:tel>
            <impl:address>삼성멀티캠퍼스</impl:address>
        </author>
        <image src="xmlandjava.gif" type="impl:gif"/>
    </book>
</booklist>
```

18. 스키마 문서의 결합

18-4 외부 스키마 문서 일부 재 정의 후 결합

- 같은 네임스페이스를 가지거나 네임스페이스가 없는 스키마 문서를 결합할 때 포함될 외부 스키마 문서의 일부를 재 정의해서 결합할 수 있다.
- 외부 스키마 문서 자체를 수정하는 것이 아니라 포함하는 과정에서 동적으로 재 정의를 한 후 주 스키마 문서에 포함하는 것이다.

문법

```
<xsd:redefine schemaLocation="포함할 스키마 문서">  
    재 정의 내용  
</xsd:redefine>
```

18. 스키마 문서의 결합

- 외부 스키마 문서 일부 재 정의 후 결합-예제

포함될 스키마 문서: c6_1905.xsd	
1	<?xml version="1.0" encoding="euc-kr"?>
2	<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3	<!-- 글로벌 심플 타입 정의 -->
4	<xsd:complexType name="ctPerson">
5	<xsd:sequence>
6	<xsd:element name="name" type="xsd:string"/>
7	<xsd:element name="tel" type="xsd:string"/>
8	<xsd:element name="address" type="xsd:string"/>
9	</xsd:sequence>
10	</xsd:complexType>
11	</xsd:schema>
스키마 문서	<xsd:redefine schemaLocation="c6_1905.xsd"> <xsd:complexType name="ctPerson"> <xsd:complexContent> <xsd:restriction base="ctPerson"> <xsd:sequence> <xsd:element name="name" type="xsd:string"/> <xsd:element name="tel" type="xsd:string"/> </xsd:sequence> </xsd:restriction> </xsd:complexContent> </xsd:complexType> </xsd:redefine>

19. 네임스페이스를 갖는 스키마

19-1 네임스페이스를 갖는 스키마 문서

- 스키마 문서가 특정 네임스페이스를 갖기 위한 선언방법.

문법

```
<?xml version="1.0" encoding="euc-kr"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="네임스페이스 이름"
  elementFormDefault="엘리먼트의 네임스페이스 포함 여부"
  attributeFormDefault="속성의 네임스페이스 포함 여부"
  xmlns="네임스페이스 이름">
  ~~~
</xsd:schema>
```

- targetNamespace : 스키마 문서의 새로운 마크업 언어가 가질 네임스페이스 지정.
- elementFormDefault : 스키마 문서의 엘리먼트가 타겟 네임스페이스에 속할 것이지를 결정.
 - **qualified** : 주어진 타겟 네임스페이스에 속하게 되고, XML 문서에서 엘리먼트를 사용할 경우 반드시 네임스페이스 접두사를 붙여서 사용해야 한다.
 - **unqualified** : 스키마 문서에 선언된 엘리먼트들은 타겟 네임스페이스에 속하지 않게 되며, XML 문서에서 엘리먼트를 사용할 경우 네임스페이스 접두사 없이 사용될 수 있다.
- attributeFormDefault : 스키마 문서에서 선언된 속성들이 주어진 타겟 네임스페이스에 속할 것이지를 언급한다. (보통은 unqualified가 사용된다.)

19. 네임스페이스를 갖는 스키마

- 네임스페이스를 갖는 스키마 문서-예제

스키마 문서: c6_2001.xsd

```
1 <?xml version="1.0" encoding="euc-kr"?>
2 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
3             targetNamespace="http://blueskii.homeip.net/xml"
4             elementFormDefault="qualified"
5             attributeFormDefault="unqualified"
6             xmlns="http://blueskii.homeip.net/xml">
7     <!-- 루트 엘리먼트 선언 -->
8     <xsd:element name="booklist" type="ctBooklist"/>
9     <!-- 글로벌 컴플렉스 타입 정의 -->
10    <xsd:complexType name="ctBooklist">
11        <xsd:sequence>
12            <xsd:element name="book" type="ctBook" minOccurs="0" maxOccurs="unbounded"/>
13        </xsd:sequence>
14    </xsd:complexType>
15    <xsd:complexType name="ctBook">
16        <xsd:sequence>
17            <xsd:element name="title" type="xsd:string"/>
18            <xsd:element name="author" type="xsd:string" minOccurs="1" maxOccurs="unbounded"/>
19            <xsd:element name="publisher" type="xsd:string"/>
20            <xsd:element name="price" type="xsd:int"/>
21        </xsd:sequence>
22    </xsd:complexType>
23 </xsd:schema>
```

19. 네임스페이스를 갖는 스키마

19-2 스키마 인스턴스 지정 방법

- XML 문서가 특정 네임스페이스를 가지고 있는 스키마 문서의 인스턴스임을 지정하기.

문법	<pre><?xml version="1.0" encoding="euc-kr"?> <접두사:루트엘리먼트 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="네임스페이스이름 스키마 문서 URI" xmlns:접두사="네임스페이스이름"> ~~~ </접두사:루트엘리먼트></pre>
----	---

- 네임스페이스가 없는 스키마 문서 URI를 지정하는 `xsi:noNamespaceSchemaLocation` 속성 대신에 네임스페이스가 있는 스키마 문서를 지정하는 속성인 `xsi:schemaLocation`을 언급하고, 해당 스키마 문서의 네임스페이스 이름과 스키마 문서 URI를 공백 문자열로 구분해서 기술해 주면 된다.

<code>xsi:schemaLocation="네임스페이스이름 http://웹서버주소/경로명/.../~.xsd"</code>

19. 네임스페이스를 갖는 스키마

- 스키마 인스턴스 지정하기-예제

XML 문서: c6_2001.xml

```
1 <?xml version="1.0" encoding="euc-kr"?>
2 <booklist
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://blueskii.homeip.net/xml c6_2001.xsd"
5   xmlns="http://blueskii.homeip.net/xml">
6
7   <book>
8     <title>JSP And Servlet</title>
9     <author>Mincheol Shin</author>
10    <publisher>InfoBook</publisher>
11    <price>25000</price>
12  </book>
13
14  <book>
15    <title>사랑과 전쟁</title>
16    <author>이사랑</author>
17    <publisher>전쟁문화사</publisher>
18    <price>15000</price>
19  </book>
20
21 </booklist>
```