6. XML Schema

dtd에 대응하는 …

Contents

개요

- DTD의 단점
  - XML 형식이 아닌 EBNF 형식을 따름
    - 예: <!ELEMENT book (title, author+, publisher, price)>
  - 재사용성과 확장성이 떨어짐
  - 제한적인 데이터 타입만을 지원
    - 예: 문자열, 열거형, 토큰타입, ID, IDREF
    - 문서 내용을 정확하게 표현하기 어려움 → 숫자 타입 없음!
- XML Schema
  - Markup 언어 및 XML 문서 양식을 정의하기 위한 언어
    - DTD의 단점을 보완
  - W3C Standard
    - XML Schema (http://www.w3.org/TR/xmlschema-0/)
      - ✓ Part 0: Primer, Part 1: Structures, Part 2: Datatypes
    - W3C XML Schema Definition Language (XSD) 1.1 (https://www.w3.org/TR/xmlschema11-1/)
      - ✓ Part 1: Structures, Part 2: Datatypes

개요

- XML Schema의 장점
  - 데이터 타입(data type)을 이용해서 엘리먼트 및 속성 정의
  - SQL, Java 등과의 호환을 위해 byte, date, integer와 같은 다양한 기본(built-in) 데이터 타입 제공
  - 기존 데이터 타입을 기반으로 새로운 사용자 정의(user-defined) 데이터 타입을 정의할 수 있음
    - DTD보다 더 복잡한 엘리먼트 구조(content model) 정의 가능
  - 데이터 타입을 상속할 수 있는 방법을 제공
    - 객체지향 개념 사용
    - 확장성과 재사용성 향상
  - XML Namespace를 지원
    - Markup 언어의 namespace를 정의함으로써 이름 충돌 방지
  - XML 데이터를 관계형 데이터베이스 등을 통해 저장 및 검색 용이

개요

- XML Schema 사용 예

student.xml

```
<students>
  <student sno='s100'>
    <name>홍길동</name>
    <age>23</age>
    <phone>02-123-8989</phone>
    <address>서울시 성북구 월곡동</address>
  </student> …
</students>
```

student.dtd

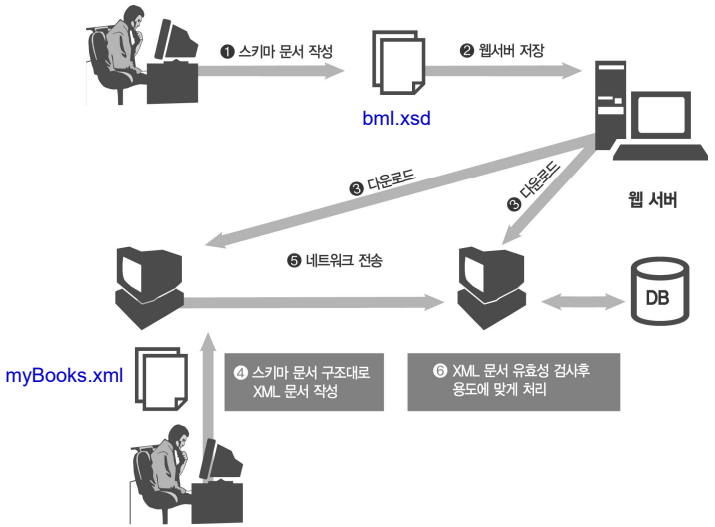
```
<!ELEMENT students (student*)>
<!ELEMENT student (name,age,phone,address)>
<!ATTLIST student sno ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT address (#PCDATA)>
```

student.xsd

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" …>
  <xsd:element name="students">
    <xsd:sequence>
      <xsd:element name="student" type="studentType" maxOccurs="unlimited">
    </xsd:sequence>
  </xsd:element>
  <xsd:complexType name="studentType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/><xsd:element name="age" type="xsd:int"/>
      <xsd:element name="phone" type="xsd:string"/><xsd:element name="address" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="sno" type="xsd:id"/>
  </xsd:complexType>
</xsd:schema>
```

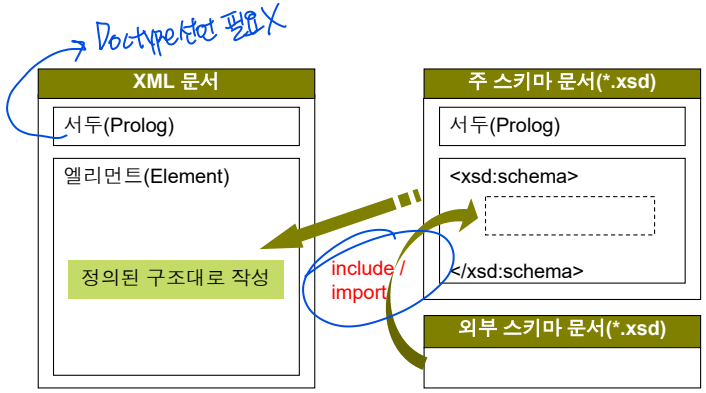
개요

- 스키마 문서의 작성 및 사용



개요

- 주 스키마 문서 및 외부 스키마 문서
  - 스키마 문서는 "xsd" 확장자를 가진 XML 문서로 작성됨



개요

- XML Schema를 이용한 XML 문서 작성
  - 이용할 스키마 파일을 지정
  - Namespace("*targetNamespace*" 속성)가 정의되지 않은 스키마
    - "*noNamespaceSchemaLocation*" 속성을 이용해서 지정

```
<student xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="student.xsd">
```

  - Namespace가 정의된 스키마
    - "*schemaLocation*" 속성을 이용해서 지정

```
<student xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.dongduk.ac.kr/student
                            student.xsd">
```

스키마 파일의 이름(경로) 스키마에 정의된 namespace 이름

  - 지정된 스키마를 적용하여 XML 문서 작성 및 유효성 검사 실시

스키마 문서의 구성

- XML Schema 문서의 주요 구성 요소
  - Type definitions
    - simple types
    - complex types
  - Declarations
    - element
    - attribute
    - notation
  - Group definitions
    - model group, attribute group
  - Constraint definitions
    - unique, key, keyref
  - Annotations

스키마 문서 구조

- DTD에 대한 기본적인 대응

DTD	XML Schema
<!ELEMENT>	<element />
<!ATTLIST>	<attribute />
,	<sequence> ...</sequence>
|	<choice> ... </choice>
+	minOccurs ="1" maxOccurs="*unbounded*"
?	minOccurs ="0" maxOccurs="1"
*	minOccurs ="0" maxOccurs="*unbounded*"

스키마 문서 구조

- <schema> 엘리먼트
  - XML Schema 문서의 루트 엘리먼트
  - 속성
    - Namespace 선언
      - ✓ Target namespace 선언
      - ✓ Default namespace 선언
      - ✓ Namespace prefixes 선언
    - Element 및 attribute에 대한 qualification 관련 선언

```
<xsd:schema
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.dongduk.ac.kr/student"
        xmlns="http://www.dongduk.ac.kr/student"     ← default namespace 선언
        elementFormDefault="qualified"
        attributeFormDefault="qualified">
  ...
</xsd:schema>
```


스키마 문서 구조

- <schema>의 자식 엘리먼트

문법

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    외부 XML Schema 문서의 참조에 관한 엘리먼트
    새로운 엘리먼트 및 속성, 타입 등을 선언하는 엘리먼트
</xsd:schema>
```

  - 외부 XML Schema 문서의 참조에 관한 엘리먼트
    - ✓ <include>, <import>, <redefine>
  - 새로운 엘리먼트 및 속성, 타입 등을 선언하는 엘리먼트
    - ✓ <element>, <attribute>, <simpleType>, <complexType>
    - ✓ <group>, <attributeGroup>
    - ✓ <notation>, <annotation>

Element 선언

- <element> 형식

```
<element
    name = NCName      ← Noncolonized Name
    ref = QName        ← Qualified Name
    type = QName
    maxOccurs = (nonNegativeInteger | unbounded) : 1   (default 값)
    minOccurs = nonNegativeInteger : 1
    default = string   fixed = string
    abstract = boolean : false
    block = (#all | List of (extension | restriction | substitution))
    final = (#all | List of (extension | restriction))
    form = (qualified | unqualified)
    id = ID
    nillable = boolean : false
    substitutionGroup = QName
    {any attributes with non-schema namespace . . .} >
    Content: (annotation?, ((simpleType | complexType)?, (unique | key | keyref)*))
</element>
```

Element 선언

- 엘리먼트 선언 방법
  - 명시적으로 정의하거나 또는 다른 엘리먼트를 참조
    - *name* 또는 *ref* 속성 이용
  - 엘리먼트 타입 지정
    - 미리 정의된 타입(global type)을 *type* 속성을 통해 지정하거나,
    - 명시적으로 local type을 정의(<simpleType>, <complexType> 이용)
  - 빈도수 제한(occurrence constraints)
    - *minOccurs, maxOccurs* 속성 이용
  - 기본값(default value) 또는 고정값(fixed value) 설정
    - *default* 또는 *fixed* 속성 이용
  - 관련된 엘리먼트 및 속성들 사이에 유일성 또는 참조 관계 제약을 설정
    - <unique>, <key>, <keyref> 이용
  - 엘리먼트 치환 그룹(substitution group)을 통해 엘리먼트의 치환 제어
    - *substitutionGroup* 속성 이용

Element 선언

1. 데이터(값)를 갖는 엘리먼트 선언
   - 엘리먼트의 타입으로 simple type을 사용
     - built-in simple type이나 사용자 정의 simple type
       - ✓ 예: xsd:string, xsd:int, xsd:double 등
   - 빈도수 지정
     - *minOccurs, maxOccurs* 속성 사용
       - ✓ 생략 시 default 값은 1 (즉, 반드시 한 번 사용해야 함)
       - ✓ *maxOccurs* = *unbounded*: 회수의 제한 없이 사용 가능

```
<!ELEMENT order (item*, …)>
<!ELEMENT item (#PCDATA)>
```

order를 위한 local element type 선언

```
<element name="order">
  <complexType><sequence>
    <element name="item" type="xsd:string"
             minOccurs="0" maxOccurs="unbounded" />
    …
</element>
```


Element 선언

- 사용 예

스키마 문서	<xsd:element name="제목" type="**xsd:string**"/>
XML 문서	<제목>XML in a Nutshell</제목>

스키마 문서	<xsd:element name="가격" type="**xsd:int**"/>
XML 문서	<가격>25000</가격> <!-- 숫자가 아닌 문자를 사용하면 오류 -->

스키마 문서	<xsd:element name="저자" type="xsd:string" **maxOccurs="5"** />
XML 문서	<저자>Elliotte Harold</저자> <!-- 1~ 5개 사용 가능 --> <저자>Scott Means</저자> <저자>Erik Ray</저자>

스키마 문서	<xsd:element name="분야" type="xsd:string" **minOccurs="0" maxOccurs="unbounded"** />
XML 문서	<분야>프로그래밍 언어</분야> <!– 생략가능, 회수의 제한 없음 --> <분야>데이터베이스</분야>

Element 선언

2. 자식 엘리먼트나 속성을 갖는 엘리먼트 선언
   - 엘리먼트의 타입으로 complex type 사용

2.1 자식 엘리먼트만 갖는 엘리먼트 선언

문법

```
<element name="엘리먼트 이름" minOccurs="횟수" maxOccurs="횟수">
  <complexType>
    <sequence>
      자식 엘리먼트들을 선언
    </sequence>
  </complexType>
</element>
```

스키마 문서

```
<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>
```


Element 선언

2.2 자식 엘리먼트는 없고 속성만 갖는 empty 엘리먼트 선언

스키마 문서

```
<xsd:element name="그림">
  <xsd:complexType>
    <!-- 속성들만 선언 -->
    <xsd:attribute name="src" type="xsd:anyURI"/>
    …
  </xsd:complexType>
</xsd:element>
```

XML 문서

```
<그림 src="book1.gif" />
```


Element 선언

2.3 자식 엘리먼트와 속성을 동시에 갖는 엘리먼트 선언

원래 순서가 없음!

- **주의: 속성 선언은 반드시 자식 엘리먼트 선언 다음에 와야 함**

스키마 문서

```
<xsd:element name="책">
  <xsd:complexType>
    <xsd:sequence>
      <!-- 자식 엘리먼트들을 선언 -->
      <xsd:element name="제목" type="xsd:string"/>
      <xsd:element name="저자" type="xsd:string"/>
      <xsd:element name="가격" type="xsd:int"/>
    </xsd:sequence>
    <!-- 속성들을 선언 -->
    <xsd:attribute name="분야" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

minOccurs=0 이면 생략 가능!

Element 선언

2.4 데이터(값)와 속성을 동시에 갖는 엘리먼트 선언

- Complex type
- Simple content 및 속성 포함

스키마 문서

```
<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>
```

content가 simple type(값)임을 의미

content의 data type 지정

속성 선언

XML 문서

```
<가격 단위="원">5000</가격>
```

속성 데이터(값)

Attribute 선언

- <attribute> 형식

```
<attribute
    name = NCName
    ref = QName
    type = QName
    use = (optional | required | prohibited) : optional    ← default 값
    default = string  fixed = string
    form = (qualified | unqualified)
    id = ID
    {any attributes with non-schema namespace . . .} >
    Content: (annotation?, (simpleType?)
</attribute>
```

- *name* 또는 *ref* : 이름을 명시적으로 정의하거나, 다른 속성을 참조
- *type* : 속성 값의 데이터 타입 지정 (simple type만 가능)

Attribute 선언

- *use* : 속성을 생략할 수 있는지 여부를 나타냄
  - "required", "optional"(default)
- *default* : 속성이 생략될 경우 자동적으로 그 값으로 인식됨
  - 단, use = "optional" 일 경우에만 적용 가능
- *fixed* : 속성의 값을 고정 (다른 값 사용 불가)

```
<xsd:element  name="book">
  <xsd:complexType>
    <xsd:attribute  name="title"  type="xsd:string"  use="required" />
  </xsd:complexType>
</xsd:element>
```

동일한 의미

```
<!ELEMENT  book  EMPTY>
<!ATTLIST  book  title  CDATA  #REQUIRED>
```


Attribute 선언

- 예 1

```
<element name="student">
  <complexType>
    <sequence>
      <element name="sname" type="string"/>
    </sequence>
    <attribute name="sid" … use="required" />
    <attribute name="alias" … use="optional" />
        → 속성이 사용되지 않을 수 있음 ( == #IMPLIED in DTD)
    <attribute name="grade" … default="1" (use="optional") />
        → 속성이 생략되면, default 값 1을 갖는 grade 속성이 있는 것으로 간주
        → DTD에서 default 값을 명시하는 것과 동일
    <attribute name="major" … fixed="computer science" (use="optional") />
        → default와 유사하나 다른 값 사용 불가능 ( == #FIXED in DTD)
  </complexType>
</element>
```


Attribute 선언

- 예 2

스키마 문서

```
<xsd:attribute name="분야" type="xsd:string" use="required" />
<xsd:attribute name="단위" type="xsd:string"
        use="optional" default="원" />
```

XML 문서

```
<책 분야="소설">
  <제목>시인과 도둑</제목>
  <가격 단위="원">9000</가격>
</책>
<책 분야="컴퓨터">
  <제목>모바일 프로그래밍</제목>
  <가격>5000</가격>
</책>
```

속성 **단위="원"** 이 존재하는 것으로 간주

Attribute 선언

- 참고: 엘리먼트 선언 vs. 속성 선언
  - ***default***
    - 속성 선언 시
      - ✓ 속성이 생략되면 그 속성은 default 값을 가짐
    - 엘리먼트 선언 시
      - ✓ 엘리먼트가 content를 갖지 않는 empty element이면, content에 default 값을 가짐
        - **주의: 엘리먼트 자체가 생략되면 그 엘리먼트는 존재하지 않음**
  - ***fixed***
    - 엘리먼트 선언과 속성 선언에서 모두 사용됨
      - ✓ 속성 값과 엘리먼트의 값(content)을 특정 값으로 설정
  - 빈도수 제한
    - 엘리먼트 선언
      - ✓ ***minOccurs*** (최소값)
      - ✓ ***maxOccurs*** (최대값)
    - 속성 선언
      - ✓ ***use*** (0 또는 1번)

전역(global) 선언 vs. 지역(local) 선언

- Global element/attribute declaration
  - <schema> 엘리먼트의 자식으로 선언
  - 전역 선언된 엘리먼트는 XML 문서의 최상위 엘리먼트 (즉, 루트 엘리먼트)로 사용 가능
  - 다른 엘리먼트/속성 선언에서 ref 속성을 통해 참조될 수 있음 (재사용 개념)
  - minOccurs, maxOccurs, use 속성을 이용한 빈도수 지정 **불가**
- Local element/attribute declaration
  - 타입 정의 안에서 선언됨
  - 다른 선언에서 참조될 수 없음 (local scope를 가짐)
  - minOccurs, maxOccurs, use 속성을 사용하여 빈도수 지정 **가능**

전역(global) 선언 vs. 지역(local) 선언

- 엘리먼트 선언 예

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <!-- global element 선언 -->
 <xsd:element name="purchaseOrder“ type="PurchaseOrderType“/>
 <xsd:element name="comment" type="xsd:string"/>
 <xsd:complexType name="PurchaseOrderType">
    <xsd:sequence>
       <!-- local element 선언 -->
      <xsd:element name="shipTo" type="USAddress"/>
      <xsd:element name="billTo" type="USAddress"/>
       <!-- global element 참조 -->
      <xsd:element ref="comment" minOccurs="0" />
      <xsd:element name="item" type="Item" maxOccurs=“unbounded" />
    </xsd:sequence>
    <xsd:attribute name="orderDate" type="xsd:date"/>
 </xsd:complexType>
 ...
</xsd:schema>
```

참조

전역(global) 선언 vs. 지역(local) 선언

- 속성 선언 예

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <!-- global 속성 선언 -->
<xsd:attribute name=“분야” type=“xsd:string”/>
<xsd:element name=“책목록”>
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name=“책” maxOccurs=“unbounded”>
     <xsd:complexType>
       …
       <!-- local 속성 선언 (global 속성 참조) -->
       <xsd:attribute ref=“분야” use=“required” />
     </xsd:complexType>
    </xsd:element>
   </xsd:sequence>
    <!-- local 속성 선언 -->
   <xsd:attribute name=“작성일" type="xsd:date" use=“optional” />
  </xsd:complexType>
</xsd:element>
</xsd:schema>
```

참조

Element 치환 그룹

- 치환 그룹을 이용한 다형성 구현
  - 스키마 문서에 같은 종류의 엘리먼트들을 여러 개 선언하고, XML 문서에서 특정 위치에 같은 종류의 엘리먼트라면 어떤 것이라도 사용 가능하도록 함
  - 엘리먼트 선언 시 *substitutionGroup* 속성을 이용하여 특정 전역 엘리먼트(→ "**substitution head**")의 이름을 지정
    - 그 엘리먼트가 substitution head 엘리먼트와 같은 종류의 엘리먼트로 간주됨
  - 같은 substitution head를 갖는 엘리먼트들의 집합을 **치환 그룹** (**substitution group**)이라 함
  - substitution head 엘리먼트가 사용되는 모든 곳에서 해당 치환 그룹에 속한 엘리먼트를 대신 사용할 수 있음

Element 치환 그룹

- 사용 예

substitution head

스키마 문서: c6_0806.xsd

```
<xsd:element name="author" type="xsd:string"/>
<xsd:element name="writer" type="xsd:string" substitutionGroup="author"/>
<xsd:element name="editor" 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>
```

치환 그룹(substitution group)

위 author 엘리먼트를 참조

Element 치환 그룹

- 사용 예

XML 문서: c6_0806.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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>
  <book>
    <title>XML Programming</title>
    <editor>홍길동</editor>
  </book>
</booklist>
```

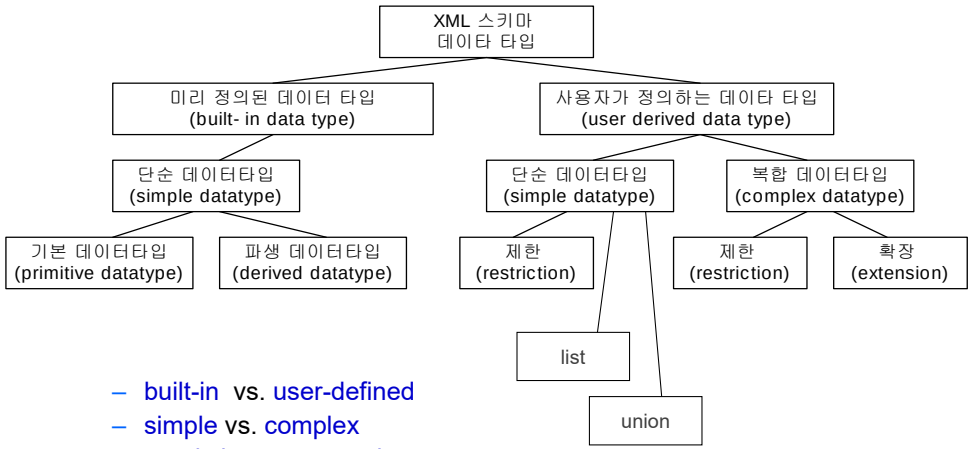
같은 치환 그룹에 속하는 엘리먼트들 중 선택

Data Types in XML Schema

- 데이터 타입이란?
  - 엘리먼트의 content 또는 속성의 값으로 어떤 종류(구조)의 정보가 어떤 형태로 작성될 수 있는지 정의
  - XML Schema는 다양한 데이터 타입을 제공함으로써 XML 문서의 구조를 더 세밀하게 정의할 수 있음
  - XML Schema는 사용자가 새로운 데이터 타입을 정의하는 것을 지원함

Data Types in XML Schema

- XML Schema 데이터 타입



- built-in vs. user-defined
- simple vs. complex
- restriction vs. extension

Data Types in XML Schema

- 데이터 타입의 분류

1. 사용 용도에 따른 분류

종류	설명	용도
빌트인(built-in) simple type	• 스키마 언어에 이미 정의되어 있는 타입	• 속성이나 엘리먼트의 content의 종류를 지정 • 사용자 정의 simple type의 base type으로 사용 • 일반적으로 XML Schema의 namespace prefix로 'xsd'를 선언하여 사용하거나 default namespace로 지정하여 사용
사용자 정의 simple type	• 사용자가 새로 정의하는 simple type	• 속성이나 엘리먼트 content의 종류를 지정
사용자 정의 complex type	• 사용자가 새로 정의하는 complex type	• 자식 엘리먼트 또는 속성을 갖는 엘리먼트 선언 시 사용

Data Types in XML Schema

- 속성 값은 반드시 simple type이어야 함

스키마 문서	<xsd:attribute name="kind" **type="xsd:string"**/> <xsd:attribute name="tel" **type="stTel"**/>

- 엘리먼트의 타입이 simple type이면 데이터만 갖는 엘리먼트가 됨

스키마 문서	<xsd:element name="title" **type="xsd:string"**/> <xsd:element name="tel" **type="stTel"**/>
***XML* 문서**	<title>Visual Programming</title> <tel>010-234-6789</tel>

Data Types in XML Schema

- 사용자 정의 complex type의 예

스키마 문서	<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>
XML 문서	<book> <title>시인과 도둑</title> <author>이문열</author> <price>9000</price> </book>

Data Types in XML Schema

2. 정의되는 위치에 따른 분류
   - Global type definition
     - <schema> 엘리먼트의 자식으로 선언
     - name 속성을 가짐
     - 다른 엘리먼트 선언 또는 데이터 타입 정의에서 참조됨
   - Local type definition
     - 특정 엘리먼트 또는 속성 선언 내에서 정의됨 (local scope)
     - name 속성을 갖지 않음 (anonymous)
     - 다른 엘리먼트나 속성 선언에서 참조될 수 없고, 해당 엘리먼트나 속성 선언에만 적용됨

Data Types in XML Schema

- Global simple type 정의 및 사용 예

스키마 문서

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <!-- global simple type 정의 -->
  <xsd:simpleType name="stTest">
     …
  </xsd:simpleType>

  <!-- global simple type 사용 -->
  <xsd:element name="test1" type="stTest"/>
  <xsd:attribute name="test2" type="stTest"/>
  …
</xsd:schema>
```


Data Types in XML Schema

- Local simple type 정의 및 사용 예

스키마 문서

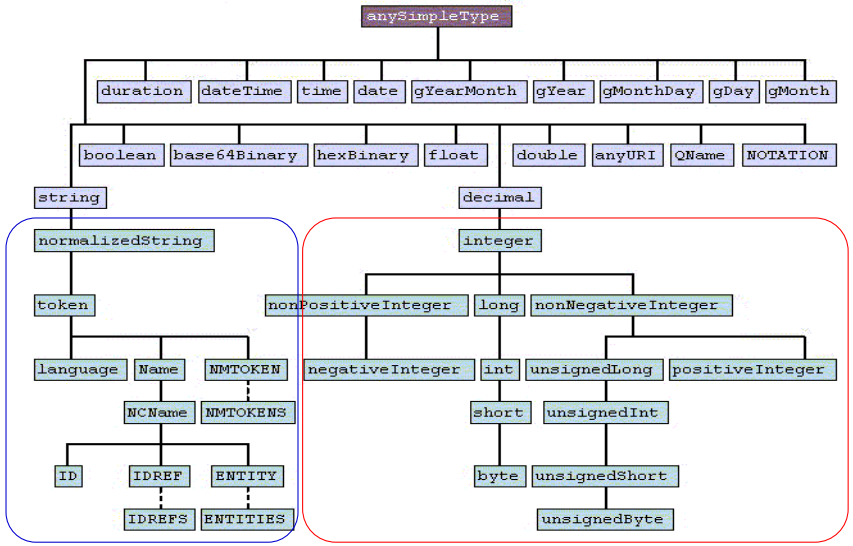
```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="test1">
    <!-- local simple type 정의 및 현 엘리먼트의 타입으로 사용 -->
    <xsd:simpleType>
       …
    </xsd:simpleType>
  </xsd:element>

  <xsd:attribute name="test2">
    <!-- local simple type 정의 및 현 속성의 타입으로 사용 -->
    <xsd:simpleType>
       …
    </xsd:simpleType>
  </xsd:attribute>
</xsd:schema>
```


Built-in Simple Type

- Built-in simple type 계층도



Built-in Simple Type

- 기본 데이터 타입(primitive data type)
  - 19개의 데이터 타입
    - 문자열, 이진 데이터, 숫자, 날짜/시간 데이터 타입
  - 문자열 데이터 타입

string	유효한 문자열
anyURI	표준 인터넷 URI를 나타내는 데이터 타입
NOTATION	외부의 비 XML content에 대한 notation 이름
QName	Namespace prefix를 포함한 Qualified Name 문자열

  - 이진 데이터 타입

boolean	true / false 값을 가짐
hexBinary	16진수로 부호화된 이진 데이터 타입
base64Binary	base64로 부호화된 이진 데이터 타입

Built-in Simple Type

  - 숫자 데이터타입

decimal	십진수	double	8바이트 실수
float	4바이트 실수		

  - 날짜/시간 데이터 타입

duration	기간 표현
dateTime	날자와 시간 표현, 날자는 그레고리력 사용
date	날자 표현 (yyyy-mm-dd 형태)
time	시간 표현
gYearMonth	그레고리력의 년과 월 표현, yyyy-mm 형태로 표현
gYear	그레고리력의 년 표현
gMonthDay	그레고리력의 월과 일 표현
gMonth	그레고리력의 월 표현
gDay	그레고리력의 일 표현

Built-in Simple Type

- 파생 데이터 타입 (derived data type)
  - string에서 파생된 데이터 타입

normalizedString	각각의 공백 문자들이 하나의 스페이스 문자로 대치되어 있는 문자열
token	모든 공백문자가 단 하나의 스페이스들로 변환되어 있는 문자열 (앞뒤에 오는 공백문자들은 모두 제거됨)
language	자연 언어 식별 문자열
Name	모든 적합한 XML1.0 이름
NCName	":"을 포함하지 않는 XML1.0 이름
ID	연관된 엘리먼트를 식별하는 고유값
IDREF	ID 속성 값을 통한 다른 엘리먼트에 대한 참조
IDREFS	IDREF 값들로 구성된 목록
NMTOKEN	적합한 XML 이름 문자들로 구성되어 있는 문자열 (XML 이름의 첫문자 제한은 적용되지 않음)
NMTOKENS	NMTOKEN 값들로 구성된 목록
ENTITY	적합한 NCName이 되는 문자열 (unparsed entity 이름)
ENTITIES	ENTITY 값들로 구성된 목록

Built-in Simple Type

  - decimal에서 파생된 데이터 타입

integer	소수점이 허용되지 않는 정수
negativeInteger	음의 정수
positiveInteger	양의 정수
nonNagativeInteger	0과 양의 정수
nonPositiveInteger	0과 음의 정수
byte	1바이트 정수
short	2바이트 정수
int	4바이트 정수
long	8바이트 정수
unsignedByte	부호 없는 1바이트 정수
unsignedShort	부호 없는 2바이트 정수
unsignedInt	부호 없는 4바이트 정수
unsignedLong	부호 없는 8바이트 정수

Simple Type Definition

- <simpleType> 엘리먼트를 사용해서 정의
  - 형태에 따라 atomic type, list type, union type으로 구분됨
- 다른 simple type을 파생하여 새로운 simple type을 정의
  - Derivation by restriction, list, or union

엘리먼트	의미
<restriction/>	Built-in simple type 또는 이미 정의된 사용자 정의 simple type을 제한하여 새로운 simple type을 정의
<list/>	공백 문자열로 분리된 토큰들의 리스트를 값으로 갖는 타입 정의
<union/>	여러 개의 simple type들을 결합하여 여러 종류의 데이터 값을 갖는 타입 정의

- 주의: 내부에 다른 엘리먼트나 속성들을 포함할 수 없음

Simple Type Definition

1. Derivation by restriction (제한)
   - 새로운 simple type은 기존의 다른 simple type이 가질 수 있는 값의 범위를 제한해서 정의할 수 있음

문법

```
<restriction base="기반이 되는 simple type 이름">
    facet elements
</restriction>
```

   - Facets
     - 데이터 타입이 가질 수 있는 값의 범위(value space)를 제한하는 방법

문법

```
<facetName value="값" />
```


Simple Type Definition

- Facet 의 종류

facet name	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 최소 개수 지정
maxLength	문자열의 최대 길이 또는 리스트 항목의 최대 개수 지정
enumeration	Enumeration 타입이 가질 수 있는 값 지정
pattern	문자 데이터의 포맷을 나타내는 정규식 지정

Simple Type Definition

- 사용 예: 정수를 기반으로 특정 범위를 갖는 새로운 simple type 정의

스키마 문서

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

<!-- 사용자 정의 simple type 적용 -->
<xsd:element name="price" type="stPrice"/>
```

XML 문서

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

<!-- 잘못된 작성 방법 -->
<price>350000</price>
<price>-10000</price>
```

Simple Type Definition

– 사용 예: enumeration 값을 갖는 simple type 정의

스키마 문서

```
<!-- 사용자 정의 simple type 정의 -->
<xsd:simpleType name="stUnit">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="원"/>
    <xsd:enumeration value="달러"/>
  </xsd:restriction>
</xsd:simpleType>
<!-- 사용자 정의 simple type 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="엔">25000</price>
```


Simple Type Definition

– 사용 예: 고정된 패턴 값을 갖는 simple type 정의

스키마 문서

```
<!-- 사용자 정의 simple type 정의 -->
<xsd:simpleType name="stSsn">
  <xsd:restriction base="xsd:string">
    <xsd:length value="14"/>
    <xsd:pattern value="\d{6}-\d{7}"/>
  </xsd:restriction>
</xsd:simpleType>
<!-- 사용자 정의 simple type 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>
```


Simple Type Definition

– 정규식(regular expression)을 이용한 문자열 pattern 지정 방법

정규식	설명	가능한 값
Book.	. : 한 개의 문자	Booka, Bookb, …, Book1, Book2
Book\d	\d : 0~9 사이의 숫자	Book0, Book1, …, Book9
Book\d*	* : zero or more	Book, Book0, …, Book100, …
Book\d+	+ : one or more	Book0, Book1, …, Book9
a?x	? : zero or one	x , ax
(a|b)x	| : 선택	ax, bx
(a|b)+x	() : 블럭	ax, bx, aax, abx, bbbx, …
[ab]\d	[ab] : a 또는 b가 와야 됨	a1, b5
Book[^0]\d*	^0 : 0은 제외	Book1, …, Book100, …
\d{3}-\d{4}	{3} : 반드시 3개가 와야 됨	110-1111, 123-1234
\d{2,3}-\d{1,4}	{1,4} : 1개~4개 올 수 있음	10-111, 123-1234
\d{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

Simple Type Definition

– 사용자 정의 simple type을 다시 제한하여 새로운 simple type 을 정의

스키마 문서

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

XML 문서

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

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

€200000하면 오류! 제한만 가능! 확장은 불가!

Simple Type Definition

2. Derivation by list
   - Atomic type 값들의 리스트로 이루어지는 타입
     - 값들은 공백으로 구분됨
   - *itemType* 속성을 통해 목록에서 사용할 수 있는 데이터 타입을 지정
   - built-in list types: NMTOKENS, IDREFS, ENTITIES
   - 예

문법	**<list itemType=**"simple type 이름"**/>**
스키마 문서	<!-- 사용자 정의 simple type 정의 --> <xsd:simpleType **name="stAuthor"**> **<xsd:list itemType**="xsd:string"/> </xsd:simpleType> <!-- 사용자 정의 simple type 적용 --> <xsd:element name="author" **type="stAuthor"**/>
XML 문서	<author>신민철 채규태 이규미</author>

Simple Type Definition

3. Derivation by union
   - 주어진 여러 개의 atomic type이나 list type들 중에서 선택 가능
   - 사용할 수 있는 데이터 타입들은 *memberTypes* 속성을 이용하여 표현
   - 예 1

```
<xsd:simpleType name="listOfIntType">
    <xsd:list itemType="xsd:int"/>
</xsd:simpleType>
<xsd:simpleType name="zipUnion">
    <xsd:union memberTypes="USState listOfIntType" />
</xsd:simpleType>
<xsd:element name="zips" type="zipUnion"/>
```

두 타입의 값을 모두 사용 가능

```
<zips>CA</zips>
<zips>AK</zips>
<zips>51480 5814 8138</zips>
```

USState 타입의 값

listOfIntType의 값

Simple Type Definition

   - 예 2

스키마 문서

```
<!-- 사용자 정의 simple type 정의 -->
<xsd:simpleType name="stAuthor1">
  <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="stAuthor1  stAuthor2"/>
</xsd:simpleType>
<!-- 사용자 정의 simple type 적용 -->
<xsd:element name="author" type="stAuthor3"/>
```

XML 문서

```
<author>신민철</author>
<author>선우혜경</author>
```


Complex Type Definition

- complex data type
  - 속성이나 자식 엘리먼트를 갖는 **엘리먼트** 선언에 필요한 타입
  - <complexType /> 을 사용해서 정의

- complex type을 정의하는 방법
  1. Restriction of the un-restricted type(ur-type)
     - content 구조 및 속성을 직접 정의
  2. Extension or restriction of a simple content complex type
     - <simpleContent /> 사용
     - 2-1. Extension
     - 2-2. Restriction
  3. Extension or restriction of a complex content complex type
     - <complexContent /> 사용
     - 3-1. Extension
     - 3-2. Restriction

Complex Type Definition

- <complexType /> 형식

```
<complexType
        name = NCName
        abstract = boolean : false
        mixed = boolean : false
        block = (#all | List of (extension | restriction))
        final = (#all | List of (extension | restriction))
        id = ID
        {any attributes with non-schema namespace . . .} >

        Content: (annotation?,
                (simpleContent | complexContent |
                ((sequence | choice | all | group )?,
                ((attribute | attributeGroup)*, anyAttribute?))))

</complexType>
```


Complex Type Definition

- Global complex type 정의

스키마 문서

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<!-- Global complex type 정의 -->
<xsd:complexType name="ctTest">      (name 필요!)
   …
</xsd:complexType>

<!-- Global complex type 적용 -->
<xsd:element name="엘리먼트명" type="ctTest" />

<xsd:attribute name="속성명" type="ctTest" />
                        (오류: 속성은 complex type 사용 불가!)
```


Complex Type Definition

- Local complex type 정의

스키마 문서

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <xsd:element name=" 엘리먼트명">
      <!-- Local complex type 정의 -->
      <xsd:complexType>       (주의: name 속성 사용 불가!)
         ~~~
      </xsd:complexType>
   </xsd:element>

   <xsd:attribute name=" 속성명 ">
      <xsd:complexType>      (오류: 속성은 simple type만 가능!)
         ~~~
      </xsd:complexType>
   </xsd:attribute>
</xsd:schema>
```


Complex Type Definition

1. *Restriction of the ur-type definition*

- 순차적인(순서가 있는) 자식 엘리먼트들을 갖는 complex type 정의
  - <sequence />를 사용하여 정의
    - 주의: 자식 엘리먼트가 하나만 있어도 <sequence>를 사용해야 함

스키마 문서

```
<xsd:element name="booklist">
  <!-- Local complex type 정의 -->
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="book" type="ctBook" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<!-- Global complex type 정의 -->
<xsd:complexType name="ctBook">
  <xsd:sequence>
    <xsd:element name="title" type="xsd:string"/>
    <xsd:element name="author" type="xsd:string“ maxOccurs="unbounded”/>
  </xsd:sequence>
</xsd:complexType>
```

Complex Type Definition

1. *Restriction of the ur-type definition*

- 자식 엘리먼트들을 선택적으로 가질 수 있는 complex type 정의
  - <choice />를 사용하여 정의
  - 예

스키마 문서

```
<!-- Global complex type 정의 -->
<xsd:complexType name="ctBooklist">
    <xsd:sequence>
        <xsd:element name="book" type="ctBook“ maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ctBook">
    <xsd:sequence>
        <xsd:element name="title" type="xsd:string"/>
        <xsd:choice  maxOccurs="unbounded“ >
            <xsd:element name="author" type="xsd:string"/>
            <xsd:element name="writer" type="xsd:string"/>
        </xsd:choice>
    </xsd:sequence>
</xsd:complexType>
```


Complex Type Definition

1. *Restriction of the ur-type definition*

- 예 (계속)

XML 문서

```
<?xml version="1.0" encoding="UTF-8"?>
<booklist  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="c6_1202.xsd">
    <book>
        <title>XML Programming</title>
        <author>신민철</author>
    </book>
    <book>
        <title>DB Programming</title>
        <writer>채규태</writer>
    </book>
    <book>
        <title>JDBC Programming</title>
        <author>신민철</author>
        <writer>채규태</writer>
    </book>
</booklist>
```


Complex Type Definition

1. *Restriction of the ur-type definition*

- 자식 엘리먼트와 속성을 동시에 갖는 complex type 정의
  - <attribute /> 선언은 <sequence />나 <choice /> 선언 다음에 와야 함

스키마 문서

```
<xsd:complexType name=”ctBook”>
 <!--자식 엘리먼트 선언-->
 <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>

<xsd:element name=” book”  type=”ctBook” maxOccurs=” unbounded” />
```

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


Complex Type Definition

1. *Restriction of the ur-type definition*

- 속성만 갖는 complex type 정의
  - empty 엘리먼트를 선언할 때에도 complex type 사용

스키마 문서

```
<!-- Global complex type 정의 -->
<xsd:complexType name=”ctImg”>
  <xsd:attribute name=”src” type=”xsd:anyURI” />
</xsd:complexType>

<!-- Global complex type 적용 -->
<xsd:element name=” img”  type=”ctImg” minOccurs=“ 0” />
```

XML 문서

```
<img src=” /images/background.gif” />
```

Complex Type Definition

1. *Restriction of the ur-type definition*

- 데이터 및 속성을 동시에 갖는 complex type 정의
  - 자식 엘리먼트 없이 속성과 simple type content만 갖는 엘리먼트 선언에 사용

스키마 문서

```
<!-- Global complex type 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>

<!-- Global complex type 적용 -->
<xsd:element name=” price”  type=”ctPrice” />
```

XML 문서

```
<price unit=”원”>25000</price>
```


Complex Type Definition

1. *Restriction of the ur-type definition*

- 참고: <all />
  - 명시된 자식 엘리먼트들이 **순서없이 한번씩만** 나타날 수 있음
    - minOccurs=“0”으로 지정할 경우 자식 엘리먼트 생략 가능 (default: 1)
  - 예

```
<xsd:complexType name="PurchaseOrderType">
   <xsd:all  minOccurs=“0”>
       <xsd:element name="shipTo" type="USAddress"/>
       <xsd:element name="billTo" type="USAddress"/>
       <xsd:element name="items" type="Items"/>
   </xsd:all>
   <xsd:attribute name="orderDate" type="xsd:date"/>
</xsd:complexType>
```

⇩

```
<purchaseOrder orderDate=“2013-04-13”>
   <items> …  </items>
   <shipTo> … </shipTo>
</purchaseOrder >
```


Complex Type Definition

1. *Restriction of the ur-type definition*

- 참고: 혼합형(mixed) content를 갖는 엘리먼트 타입 정의
  - 자식 엘리먼트와 문자열 데이터가 혼합되어 사용될 수 있음

```
<xsd:element name="letter">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element name="name" type="xs:string"/>
      <xsd:element name="orderid" type="xs:positiveInteger"/>
      <xsd:element name="shipdate" type="xs:date"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

⇩

```
<letter>
Dear Mr.<name>John Smith</name>.
Your order <orderid>1032</orderid>
will be shipped on <shipdate>2013-04-13</shipdate>.
</letter>
```


Complex Type Definition

- *Extension or Restriction of a complex type*

- Complex type의 확장 및 제한
  - 기존 complex type을 확장 또는 제한하여 새로운 complex type을 정의할 수 있음 (상속 개념)
    - 확장(extension) : 기반이 되는 complex type에 속성이나 엘리먼트를 추가
    - 제한(restriction) : 기반이 되는 complex type이
      - ✓ simple content인 경우 → 데이터 및 속성 값의 범위를 제한
      - ✓ complex content인 경우 → 자식 엘리먼트나 속성을 제한
  - XML 문서에서 타입의 다형성(polymorphism) 이용 가능

Complex Type Definition

2.1 *Extension of a simple content complex type*

- Simple content를 갖는 complex type의 확장
  - 데이터와 속성을 갖는 simple content complex type의 확장
    - **속성을 추가**하여 새로운 complex type을 정의하는 것
  - <simpleContent />, <extension /> 사용

문법	

	<xsd:complexType name="complex type 이름"**>**
	  <xsd:simpleContent>
	  <xsd:extension base="확장할 complex type 이름">
	  속성 선언들
	  </xsd:extension>
	  </xsd:simpleContent>
	</xsd:complexType>

반드시 simple content를 갖는 complex type을 지정해야 함

Complex Type Definition

2.1 *Extension of a simple content complex type*

- Simple content를 갖는 complex type의 확장
  - 예

스키마 문서	<!-- Global complex type 정의 --> **<xsd:complexType name="ctPrice2">** <!-- simple content를 갖는 complex type 정의 --> **<xsd:simpleContent>** <!-- 슬라이드 p.64의 ctPrice type을 확장 --> **<xsd:extension base="ctPrice">** **<xsd:attribute name="card" type="xsd:boolean"/>** **</xsd:extension>** **</xsd:simpleContent>** **</xsd:complexType>** <!-- Global complex type 적용 --> <xsd:element name="price" **type="ctPrice2"**/>
XML 문서	<price unit="원" **card="true"**>25000</price>

Complex Type Definition

2.2 *Restriction of a simple content complex type*

- Simple content를 갖는 complex type의 제한
  - 기반이 되는 simple content complex type을 제한하여 새로운 complex type을 정의
    - 기반 complex type의 **데이터 값의 범위**를 제한
  - <simpleContent />, <restriction /> 사용

문법	

	<xsd:complexType name="complex type 이름">
	  <xsd:simpleContent>
	  <xsd:restriction base="제한할 complex type 이름">
	  facet 엘리먼트들
	  </xsd:restriction>
	  </xsd:simpleContent>
	</xsd:complexType>

반드시 simple content를 갖는 complex type을 지정해야 함

Complex Type Definition

2.2 *Restriction of a simple content complex type*

- Simple content를 갖는 complex type의 제한
  - facet 엘리먼트: 값에 대한 제한 방법을 표현

facet 이름	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	enumeration 타입이 가질 수 있는 값 지정
pattern	문자 데이터의 포맷을 나타내는 정규식 지정

Complex Type Definition

2.2 *Restriction of a simple content complex type*

- Simple content를 갖는 complex type의 제한
  - 예

스키마 문서

```
<!-- Global complex type 정의 -->
<xsd:complexType name="ctPrice3">
  <!-- simple content를 갖는 complex type 정의 -->
  <xsd:simpleContent>
    <!-- 슬라이드 p.64의 ctPrice type을 제한 -->
    <xsd:restriction base="ctPrice">
      <xsd:minInclusive value="1000"/>
      <xsd:maxInclusive value="100000"/>
    </xsd:restriction>
  </xsd:simpleContent>
</xsd:complexType>

<!-- Global complex type 적용 -->
<xsd:element name="price" type="ctPrice3"/>
```

XML 문서

```
<price unit="원">50000</price>
```


Complex Type Definition

3.1 *Extension of a complex content complex type*

- 자식 엘리먼트나 속성을 갖는 complex type의 확장
  - 자식 엘리먼트나 속성을 갖고 있는 complex content complex type에 **새로운 자식 엘리먼트나 속성을 추가**하여 새로운 complex type을 정의
    - Content가 없는 complex type도 확장 가능 (empty element)

문법

```
<xsd:complexType name="complex type 이름">
  <xsd:complexContent>
    <xsd:extension base="확장할 complex type 이름">
      <xsd:sequence>
        추가할 엘리먼트 선언들
      </xsd:sequence>

      추가할 속성 선언들
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```


Complex Type Definition

3.1 *Extension of a complex content complex type*

- 예: c6-1305.xsd

스키마 문서

```
<xsd:complexType name="ctBook">
    <xsd:sequence>
        <xsd:element name="title" type="xsd:string"/>
        <xsd:element name="author" type="xsd:string"
                     minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
</xsd:complexType>
<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>
```


Complex Type Definition

3.2 *Restriction of a complex content complex type*

- 자식 엘리먼트나 속성을 갖는 complex type의 제한
  - 기반 complex type의 자식 엘리먼트나 속성에 대해 제한
    - 기반 complex type의 **모든 자식 엘리먼트들을 재선언**해야 함
      - ✓ 자식 엘리먼트의 타입 변경(제한), 빈도 수 제한, default 값 선언 등 가능
      - ✓ 새로운 자식 엘리먼트를 추가할 수는 없음
      - ✓ **주의: 자식 엘리먼트가 local type으로 선언된 경우 제한 불가**
    - 속성은 **제한하고자 하는 속성만 재선언** 가능
      - ✓ 속성 값의 제한은 작은 범위로의 제한만 가능

문법

```
<xsd:complexType name="complex type 이름">
  <xsd:complexContent>
    <xsd:restriction base="제한할 기반 complex type 이름">
      - 기반 type이 가지고 있는 모든 엘리먼트들을 재선언
      - 기반 type에서 제한하고자 하는 속성들만 재선언
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```

Complex Type Definition

3.2 *Restriction of a complex content complex type*

– 예

스키마 문서

```
<xsd:complexType name="ctBook3">
  <xsd:complexContent>
    <xsd:restriction base="ctBook2">          ← p.74의 ctBook2를 기반으로 정의
      <xsd:sequence>
        <xsd:element name="title" type="xsd:string" default="..." />
        <xsd:element name="author" type="xsd:string"
                     maxOccurs="0" minOccurs="0" />
        <xsd:element name="publisher" type="xsd:string"/> <!-- 재선언 -->
        <xsd:element name="price">      <!-- price 타입 재정의(제한) -->
          <xsd:simpleType>
            <xsd:restriction base="xsd:int">
              <xsd:minInclusive value="0"/>
              <xsd:maxInclusive value="100000"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
      </xsd:sequence>
```

revisionDate
max=0
min=0

이미 제한해서 ctBook4에서 제한 불가!

이용기간
max = 14

Complex Type Definition

3.2 *Restriction of a complex content complex type*

– 예 (계속)

스키마 문서

```
        <xsd:attribute name="id" type="xsd:ID"/>   <!-- 재선언 생략 가능 -->
        <xsd:attribute name="kind">      <!-- kind 속성 타입 재정의(제한) -->
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="컴퓨터"/>
              <xsd:enumeration value="자연과학"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:attribute>
      </xsd:restriction>
    </xsd:complexContent>
</xsd:complexType>

<!-- Global complex type 적용 -->
<xsd:element name="book" type="ctBook3" maxOccurs="unbounded"/>
```

Complex Type Definition

3.2 *Restriction of a complex content complex type*

– 예 (계속)

XML 문서

```
<?xml version="1.0" encoding="UTF-8"?>
<booklist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="c6_1305.xsd">

  <book id="b1" kind="컴퓨터" >  <!-- 제한된 범위의 값만 사용 가능 -->
    <title>XML And Java</title>
    <publisher>프리렉</publisher>
                                  <!-- author 엘리먼트는 사용 불가 -->
    <price>25000</price>          <!-- 제한된 범위의 값만 사용 가능 -->
  </book>
  …
</booklist>
```


Complex Type Definition

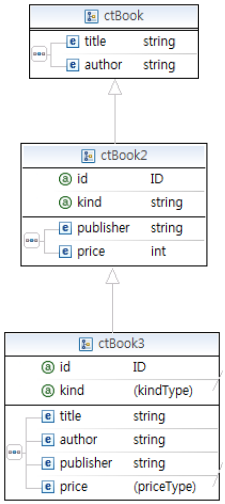
- Complex type의 다형성(polymorphism)
  - 엘리먼트 선언 시: *xsi:type* 속성을 이용하여 다형성을 갖는 complex type을 지정
  - XML 문서 작성 시: 해당 complex type 외에 그것을 확장한 파생 complex type의 엘리먼트도 사용 가능

문법	<엘리먼트 이름 xsi:type="complex type 이름">
스키마 문서	<!-- Global complex type 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> <price>10000</price> </book>

ctBook 뿐만 아니라 *ctBook*으로부터 파생된 타입도 사용 가능

Complex Type Definition

– 예: p.74, p.76~77의 ctBook, ctBook2, ctBook3 타입 정의 이용



스키마 문서

```
<xsd:complexType name="ctBooklist">
  <xsd:sequence>
    <xsd:element name="book" type="ctBook"
                 maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="booklist" type="ctBooklist"/>
```


Complex Type Definition

– 예 (계속)

XML 문서

```
<?xml version="1.0" encoding="UTF-8"?>
<booklist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="c6_1305.xsd">
  <book>                                  ← ctBook 타입
    <title>XML And Java</title>
    <author>신민철</author>
  </book>
  <book id="b1" kind="수필" xsi:type="ctBook2">
    <title>사랑과 전쟁</title>
    <author>이사랑</author>
    <publisher>사랑문화사</publisher>
    <price>250000</price>
  </book>
  <book id="b2" kind="자연과학" xsi:type="ctBook3">
    <title>우주의 역사</title>
    <publisher>과학출판사</publisher>
    <price>25000</price>
  </book>
</booklist>
```

Model Group, Attribute Group

- 필요성
  - 스키마 문서에서 여러 가지 complex type 정의 시 중복해서 사용되는 엘리먼트나 속성 선언들을 그룹으로 정의 (이름 부여)
  - ref 속성을 사용하여 여러 곳에서 참조하여 사용할 수 있음
  - 스키마의 재사용성 향상, 가독성 향상, 변경 용이

- 종류
  - 모델 그룹(model group) : 중복되는 엘리먼트 선언들을 모아서 만든 그룹
  - 속성 그룹(attribute group) : 중복되는 속성 선언들을 모아서 만든 그룹

Model Group, Attribute Group

- 모델 그룹
  - 형식

```
<group  name = "모델 그룹 이름">
    Content: (annotation?, (all | choice | sequence)?)
</group>

<group ref = "참조할 모델 그룹 이름" />
```

  - named <group> element
    - 이름을 가진 model group (a group of elements) 정의
    - 전역적으로 정의됨 (<schema>의 자식 엘리먼트)
  - unnamed <group> element
    - 모델 그룹을 참조하여 재사용하기 위함
    - ref 속성을 이용하여 참조할 그룹 지정
    - complex type definition 내에서 사용

Model Group, Attribute Group

– 사용 예

```
<!-- 모델 그룹 정의 -->
<xsd:group name="shipAndBill">
   <xsd:sequence>
        <xsd:element name="shipTo" type="USAddress"/>
        <xsd:element name="billTo" type="USAddress"/>
   </xsd:sequence>
</xsd:group>

<!-- 모델 그룹 참조 -->
<xsd:complexType name="PurchaseOrderType">
   <xsd:sequence>
        <xsd:choice>
             <xsd:group ref="shipAndBill"/>
             <xsd:element name="singleUSAddress" type="USAddress"/>
        </xsd:choice>
        <xsd:element name="items" type="Items"/>
   </xsd:sequence>
   <xsd:attribute name="orderDate" type="xsd:date"/>
</xsd:complexType>
```

참조

Model Group, Attribute Group

❑ 속성 그룹

– 형식

```
<attributeGroup  name = "속성 그룹 이름">
    Content: (annotation?, attribute*)
</attributeGroup>

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

– 사용 방법은 모델 그룹과 유사

- 전역적으로 정의
- complex type 정의 시 ref 속성을 이용하여 참조됨

Model Group, Attribute Group

– 사용 예

```
<!-- 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>
```

참조 참조

Constraints

❑ 제약 조건

– 제약조건의 종류: unique, key, keyref (key reference)

– 각 제약조건은 <selector /> 및 <field /> 엘리먼트를 이용하여 정의

- xpath 속성: XPath 식을 사용하여 대상을 지정
- <selector />
  - ✓ 제약조건이 적용되는 엘리먼트들의 집합을 설정
  - ✓ XPath 식은 제약조건이 포함된 엘리먼트에 대해 상대적인 경로로 명세됨
    - 일반적으로 루트 엘리먼트 내에서 정의
- <field />
  - ✓ 제약조건을 만족해야 하는 속성 또는 자식 엘리먼트를 지정
  - ✓ XPath 식은 <selector />에 의해 선택된 엘리먼트 집합에 대해 상대적으로 명세됨

Unique Constraint

- **<unique /> element**

문법	<unique name="unique 제약조건 이름"> <selector xpath="대상 엘리먼트의 경로"/> <field xpath="제약을 적용할 속성이나 자식 엘리먼트의 경로"/> </unique>

- <selector>에 의해 선택된 엘리먼트들의 각 instance에 대해, <field> 에 명시된 자식 엘리먼트 또는 속성의 값이 유일해야 함을 의미
  - 모든 instance에 대해 <field>에 명시된 자식 엘리먼트나 속성이 반드시 존재할 필요는 없음
- <element> 선언 내에서, 마지막 부분에 정의되어야 함

Unique Constraint

- 예: 스키마 문서(c6_1505.xsd)

```
<xsd:element name="booklist" >
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="book" type="ctBook" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:unique name="keyBook">          // 또는 <xsd:key … > 정의
        <xsd:selector  xpath="./book" />  // booklist 엘리먼트에 대한 상대 경로
        <xsd:field  xpath= "./@id" />     // booklist/book 엘리먼트에 대한 상대 경로
    </xsd:unique>
</xsd:element>

<xsd:complexType name="ctBook">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name= "id"  type="xsd:string" use="required"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
```

Unique Constraint

- 예: XML 문서

```
<?xml version="1.0" encoding="UTF-8"?>
<booklist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="c6_1505.xsd">

    <book id="b1">XML 기초서</book>
    <book id="b2">사랑과 야망</book>
    <book id="b3">Java Programming</book>
    …
</booklist>
```

- unique 제약조건은 <book>들의 집합에서 id 속성 값들이 모두 달라야 함을 의미
- key 제약조건은 id 속성이 <book>들에 대한 식별자 역할을 해야 함을 의미 (반드시 존재)

Key Constraint

- **<key /> element**

문법	<key name="key 제약조건 이름"> <selector xpath="대상 엘리먼트의 경로"/> <field xpath="제약을 적용할 (즉 key로 사용될) 속성이나 엘리먼트의 경로"/> </key>

- 대상 엘리먼트들에 대한 식별자 정의
- 다음 세 가지 조건을 동시에 만족해야 함
  - *unique* constraint
  - *existence* constraint (i.e., must exist)
  - *non-null* constraint

Key Constraint

```
<xs:element name="vehicle">
  <xs:complexType>
  . . .
    <xs:attribute name="plateNumber" type="xs:integer"/>
    <xs:attribute name="state" type="twoLetterCode"/>
  </xs:complexType>
</xs:element>

<xs:element name="state">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="code" type="twoLetterCode"/>
      <xs:element ref="vehicle" maxOccurs="unbounded"/>
      <xs:element ref="person“ maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:key name="reg">
        <xs:selector xpath="./vehicle"/>
        <xs:field xpath="@plateNumber"/>
  </xs:key>
</xs:element>
```

각 vehicle들은 *plateNumber* 속성을 key로 하여 구별됨을 의미함 (각 state 마다 적용)

Keyref Constraint

- **<keyref />** 제약 조건

문법	<keyref name=”제약조건 이름” refer=”참조할 key 제약조건 이름”> <selector xpath=”대상 엘리먼트의 경로”/> <field xpath=”제약을 적용할 속성이나 엘리먼트의 경로”/> </keyref>

  - 참조 무결성(referential integrity) 제약을 표현
    - <field /> 엘리먼트에 지정된 속성 또는 엘리먼트의 값은 반드시 다른 엘리먼트의 키 값(<key /> 제약조건으로 지정)을 참조해야 함
      - ✓ 즉, 해당 키 값으로 사용된 값들 만을 사용 가능

Keyref Constraint

- 예: 스키마 문서(c6_1504.xsd)

```
<xsd:element name="booklist“ >
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="kinds">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="kind" type="ctKind" maxOccurs="unbounded"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="book" type="ctBook" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:key  name="keyKind">
        <xsd:selector  xpath="./kinds/kind"/>
        <xsd:field  xpath= "./@id"/>
    </xsd:key>
    <xsd:keyref  name="keyrefBook" refer=“keyKind”>
        <xsd:selector  xpath="./book"/>
        <xsd:field  xpath= "./@bookKind"/>
    </xsd:keyref>
</xsd:element>
```

id 속성을 포함한다고 가정

bookKind 속성을 포함한다고 가정

참조

Keyref Constraint

- 예: 스키마 문서(c6_1504.xsd) (계속)

```
<xsd:complexType name="ctKind">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="id" type="xsd:string" use="required"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="ctBook">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="id" type="xsd:string" use="required"/>
            <xsd:attribute name="bookKind"  type="xsd:string" use="required"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
```

Keyref Constraint

- 예: XML 문서

```
<?xml version="1.0" encoding="UTF-8"?>
<booklist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="c6_1506.xsd">

    <kinds>
        <kind id="k1">컴퓨터</kind>
        <kind id="k2">소설</kind>
        <kind id="k3">잡지</kind>
    </kinds>

    <book id="b1" bookKind="k1">XML 기초서</book>
    <book id="b2" bookKind="k2">사랑과 야망</book>
    <book id="b3" bookKind="k1">Java Programming</book>
    …
</booklist>
```

  - <book>의 bookKind 속성 값(k1, k2)들은 <kind>의 id 속성 값들을 참조

idref 사용시 book id도 사용 가능 → 잘못되었지만 잡아내지 못함!

Notation 선언

- Notation
  - 이미지, 동영상, 음악 등 이진 데이터 파일의 포맷을 식별하기 위해 사용되는 특별한 엘리먼트
  - XML parser가 해석할 수 없는 이진 데이터가 어떤 종류의 포맷을 따르고, 이 데이터를 처리할 수 있는 helper application은 무엇인지를 응용 프로그램에게 알려줌

문법	<notation name="notation 이름" public="공개 식별자 (MIME 타입을 지정)" system="helper application"/>

  - 선언된 notation을 속성 값 또는 데이터 값으로 사용할 때는 NOTATION 타입을 제한하는 사용자 정의 simple type을 만들어서 사용

Notation 선언

- 사용 예 (c6_1803)

coverImage

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"/>

<!-- Global simple type 정의 -->
<xsd:simpleType name="stType">
    <xsd:restriction base="xsd:NOTATION">  <!-- NOTATION 타입을 제한 -->
        <xsd:enumeration value="bmp"/>
        <xsd:enumeration value="gif"/>
        <xsd:enumeration value="jpeg"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- Global simple type 적용 -->
<xsd:complexType name="ctImage">
    <xsd:attribute name="src" type="xsd:anyURI"/>
    <xsd:attribute name="type" type="stType"/>
</xsd:complexType>
```

Notation 선언

- 사용 예

XML 문서

```
<?xml version="1.0" encoding="UTF-8"?>
<booklist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="c6_1701.xsd">
    <book>
        <title>XML And Java</title>
        <image src="xmlandjava.gif" type="gif" />
    </book>
    <book>
        <title>사랑과 전쟁</title>
        <image src="loveandwar.bmp" type="bmp" />
    </book>
</booklist>
```

Namespace를 갖는 스키마

- XML 스키마 선언
  - XML 스키마의 namespace을 설정하고 사용하기 위해 선언
  - 형식

```
<xsd:schema
        xmlns:xsd="http://www.w3.org/2001/XMLSchema" ①
        targetNamespace="http://www.dongduk.ac.kr/BML" ②
        xmlns="http://www.dongduk.ac.kr/BML" ③
        elementFormDefault="qualified" ④
        attributeFormDefault="unqualified" ⑤>
    ...
</xsd:schema>
```

namespace 접두사

default namespace → 접두사 없을 때 ② 적용

①, ③ namespace 선언

Namespace를 갖는 스키마

- XML 스키마 선언 속성

속성	설명
① **xmlns:xsd**	XML Schema의 namespace 지정
② **targetNamespace**	현재 스키마에서 선언된 이름들이 속하는 namespace 지정
③ **xmlns**	default namespace를 지정
④ **elementFormDefault**	스키마를 사용하는 XML 문서에 대한 지시문 - 속성값이 *qualified*이면: XML 문서에서 사용하는 모든 엘리먼트에 대해 namespace prefix를 사용해야 함 - 속성값이 *unqualified*이면: global element에 대해서만 namespace prefix 사용 - default: *unqualified*
⑤ **attributeFormDefault**	elementFormDefault와 동일한 의미이나 속성 이름에 대해 적용됨 - default: *unqualified*

접두사 붙이면 qualified

생략 → unqualified

Namespace를 갖는 스키마

- Target namespace를 default namespace로 선언
  - 스키마 문서에서 정의하는 타입 참조 시 타입 이름에 namespace 접두사 생략 가능

```
<xsd:schema
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace=http://www.dongduk.ac.kr/BML
      xmlns="http://www.dongduk.ac.kr/BML">

  <xsd:complexType name="ctBooklist">
    …
  </xsd:complexType>

  <xsd:element name="booklist" type="ctBooklist"/>
  ...
</xsd:schema>
```

namespace 접두사 생략

Namespace를 갖는 스키마

- XML Schema namespace를 default namespace로 선언
  - XML Schema 표준과 관련된 엘리먼트에는 "xsd"라는 접두사를 사용하지 않아도 됨
  - 그 대신 target namespace에 대한 접두사 필요

```
<schema
      xmlns="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.dongduk.ac.kr/BML"
      xmlns:bml="http://www.dongduk.ac.kr/BML">

  <complexType name="ctBooklist">
    …
  </complexType>

  <element name="booklist" type="bml:ctBooklist" />
  ...
</schema>
```

"bml" 접두사 사용

생략하면 스키마 안에 정의되었다고 해석 → 오류 (없어서..)

Namespace를 갖는 스키마

- XML 문서에서 스키마 지정 방법
  - XML 문서가 target namespace를 갖고 있는 스키마의 인스턴스일 경우 xsi:schemaLocation 속성을 이용하여 지정
    - 그렇지 않으면 xsi:noNamespaceSchemaLocation 속성 이용

문법

```
<?xml version="1.0" encoding="UTF-8"?>
<루트_엘리먼트_이름
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="namespace_이름  스키마_문서의_URI"
>
   …
</루트_엘리먼트_이름>
```

예

```
<?xml version="1.0" encoding="UTF-8"?>
<bml:booklist
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.dongduk.ac.kr/BML   bml.xsd"
   xmlns:bml="http://www.dongduk.ac.kr/BML">
   …
</bml:booklist>
```


Namespace를 갖는 스키마

- 예: po1.xsd

```
<schema targetNamespace="http://www.example.com/PO1"
        xmlns:po="http://www.example.com/PO1"
        xmlns="http://www.w3.org/2001/XMLSchema"
>
<element name="purchaseOrder" type="po:PurchaseOrderType"/>
<element name="comment" type="string"/>

<complexType name="PurchaseOrderType">
  <sequence>
    <element name="shipTo" type="po:USAddress"/>
    <element name="billTo" type="po:USAddress"/>
    <element ref="po:comment" minOccurs="0"/>
  </sequence>
  <attribute name="orderDate" type="date"/>
</complexType>
<complexType name="USAddress">
  <sequence>
    <element name="name" type="string"/>
    <element name="street" type="string"/>
  </sequence> </complexType> </schema>
```

elementFormDefault="qualified"
attributeFormDefault="qualified"
(default: unqualified)

global elements:
purchaseOrder, **comment**

local elements:
shipTo, **billTo**, **name**, **street**

Namespace를 갖는 스키마

- 예: po1.xml

```
<?xml version="1.0"?>
<apo:purchaseOrder  xmlns:apo="http://www.example.com/PO1"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.example.com/PO1
                                        http://www.example.com/po1.xsd"
                    orderDate="1999-10-20">
  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
  </shipTo>
  <billTo country="US">
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
  </billTo>
  <apo:comment>Hurry, my lawn is going wild!</apo:comment>
</apo:purchaseOrder>.
```

local elements

✓ po1.xsd에서 elementFormDefault="**unqualified**" 이므로 **global element**에 대해서만 **namespace prefix**가 필요함

Namespace를 갖는 스키마

- 예: po2.xsd (*elementFormDefault = "qualified"* 사용)

```
<schema   targetNamespace="http://www.example.com/PO2"
          xmlns:po="http://www.example.com/PO2"
          xmlns="http://www.w3.org/2001/XMLSchema"
          elementFormDefault="qualified">

<element name="purchaseOrder" type="po:PurchaseOrderType"/>
<element name="comment" type="string"/>
<complexType name="PurchaseOrderType">
  <sequence>
    <element name="shipTo" type="po:USAddress"/>
    <element name="billTo" type="po:USAddress"/>
    <element ref="comment" minOccurs="0"/>
  </sequence>
  <attribute name="orderDate" type="date"/>
</complexType>
<complexType name="USAddress">
  <sequence>
    <element name="name" type="string"/>
    <element name="street" type="string"/>
  </sequence>  </complexType> </schema>
```

Namespace를 갖는 스키마

- 예: po2-1.xml

```
<?xml version="1.0"?>
<apo:purchaseOrder  xmlns:apo="http://www.example.com/PO2"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.example.com/PO2
                                        http://www.example.com/po2.xsd"
                    orderDate="1999-10-20">
  <apo:shipTo country="US">
    <apo:name>Alice Smith</apo:name>
    <apo:street>123 Maple Street</apo:street>
  </apo:shipTo>
  <apo:billTo country="US">
    <apo:name>Robert Smith</apo:name>
    <apo:street>8 Oak Avenue</apo:street>
  </apo:billTo>
  <apo:comment>Hurry, my lawn is going wild!</apo:comment>
</apo:purchaseOrder>.
```

local elements

✓ po2.xsd에서 elementFormDefault="**qualified**" 이므로 **모든 element들에 대해 qualification이 필요함**

Namespace를 갖는 스키마

- 예: po2-2.xml (default namespace 사용)

```
<?xml version="1.0"?>
<purchaseOrder  xmlns=http://www.example.com/PO2
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http:// www.example.com/PO2
                                    http://www.example.com/po2.xsd "
                orderDate="1999-10-20">
  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
  </shipTo>
  <billTo country="US">
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
  </billTo>
  <comment>Hurry, my lawn is going wild! </comment>
</purchaseOrder>.
```

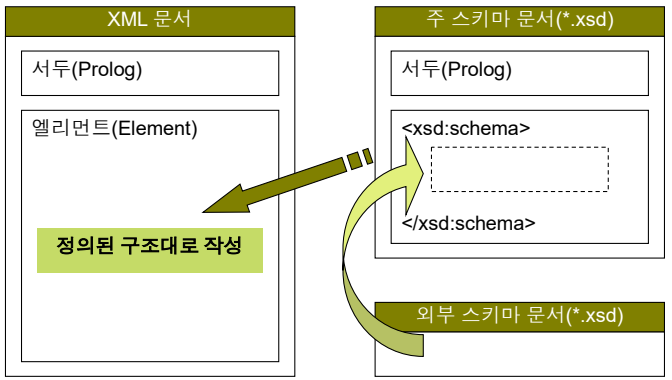
✓ 특정 namespace를 **default namespace**로 선언할 경우, 그 namespace에 속하는 **엘리먼트 이름에는 명시적인 qualification (prefix) 필요 없음** (자동적으로 qualification이 된 것임)

Namespace를 갖는 스키마

- 주의 사항
  - *elementFormDefault* = "*unqualified*" 인 경우 XML 문서에서 default namespace를 사용하는 것은 불가능함
    - default namespace를 지정할 경우 local element들에 대해서도 자동적으로 qualification이 되므로 *elementFormDefault* = "*unqualified*" 선언에 위배되기 때문
  - *elementFormDefault* = "*qualified*" 및 default namespace 사용 권장
  - 스키마 정의에서 global element 선언이 거의 없는 경우 *elementFormDefault* = "*unqualified*" 사용

스키마 문서의 결합

- 공통적으로 사용되는 정의 및 선언들을 별도의 스키마 문서에서 작성하고, 주 스키마 문서에서는 같은 내용을 중복 작성하지 않고 참조할 수 있음



스키마 문서의 결합

- **<include/>**
  - 여러 스키마 문서들을 조합하여 하나의 target namespace에 대한 스키마를 구성할 수 있음
  - **같은 namespace를 갖거나 또는 namespace가 없는** 스키마 문서만 포함 가능
  - ***schemaLocation*** 속성을 이용하여 스키마 문서의 위치를 나타냄

• "http://abc.com"을 namespace로 갖는 스키마 문서 (*http://abc.com/schema1.xsd*)

```
<schema targetNamespace="http://abc.com" … >
```

• 같은 namespace를 갖는 다른 스키마 문서(*http://abc.com/schema2.xsd*)

```
<schema targetNamespace="http://abc.com" … >
    <include schemaLocation="http://abc.com/schema1.xsd" />
```


스키마 문서의 결합

- <include/> 사용 예

포함될 스키마 문서: c6_1902.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd=
            "http://www.w3.org/2001/XMLSchema">

  <!-- Global simple type 정의 -->
  <xsd:simpleType name="stType">
    …
</xsd:schema>
```

주 스키마 문서: c6_1901.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd=
            "http://www.w3.org/2001/XMLSchema">

<xsd:include schemaLocation="c6_1902.xsd" />

  <!-- Global complex type 정의 -->
  <xsd:complexType name="ctImage">
    <xsd:attribute name="type" type="stType"/>
  </xsd:complexType> …
</xsd:schema>
```

참조

XML 문서: c6_1901.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<booklist xmlns:xsi="http://www.w3.org/2001/
                     XMLSchema-instance"
   xsi:noNamespaceSchemaLocation
            ="c6_1901.xsd">

  <book>
    <title>XML And Java</title>
    <image src="xmlandjava.gif" type="gif"/>
  </book>

  <book>
    <title>사랑과 전쟁</title>
    <image src="loveandwar.bmp"
   type="bmp"/>
  </book>

</booklist>
```

스키마 문서의 결합

- **<import/>**
  - **Namespace가 다른** 스키마 문서들을 결합하기 위해 사용
  - 다른 namespace에 정의된 스키마 요소들을 함께 사용할 수 있도록 함
  - <schema> 엘리먼트의 첫번째 자식 엘리먼트로 명시되어야 함
  - ***namespace*** 및 ***schemaLocation*** 속성 이용

문법	**<xsd:import namespace="포함할 스키마 문서의 target namespace" schemaLocation="포함할 스키마 문서의 경로"/>**

스키마 문서의 결합

- <import/> 사용 예

b.xsd

```
<schema xmlns =
    "http://www.w3.org/2001/XMLSchema"
targetNamespace="http://example.org/b/"
/>
<simpleType name="SKU"/>
. . .
</simpleType>
```

c.xsd

```
<schema xmlns =
  "http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://example.org/c/"
/>
<element name="c1" type="myCompType"/>
…
```

a.xsd

```
<xsd:schema xmlns:xsd =
        "http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://example.org/a"
  xmlns="http://example.org/a"
  xmlns:b="http://example.org/b/"
  xmlns:c="http://example.org/c/">

<xsd:import namespace="http://example.org/b/"
            schemaLocation="b.xsd" />
<xsd:import namespace="http://example.org/c/"
            schemaLocation="c.xsd" />

<xsd:element name="a1" type="a1Type"/>

<xsd:complexType name="a1Type">
  <xsd:sequence>
    <xsd:element name="a2" type="xsd:string"/>
    <xsd:element name="a3" type="b:SKU"/>
    <xsd:element ref="c:c1"/>
  </xsd:sequence>
</xsd:complexType>
```

스키마 문서의 결합

- **<redefine/>**
  - 같은 namespace를 갖거나 namespace가 없는 외부 스키마 문서를 include할 때, 그 스키마의 일부를 재정의해서 결합함
  - 외부 스키마 자체를 수정하는 것이 아니라, 포함하는 과정에 동적으로 재정의 한 후 사용하는 것임

문법	<xsd:redefine schemaLocation="포함할 스키마 문서의 경로"> 재정의 할 내용 </xsd:redefine>

스키마 문서의 결합

- <redefine/> 사용 예

포함될 스키마 문서: c6_1905.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="ctPerson">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="tel" type="xsd:string"/>
      <xsd:element name="address" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType> …
</xsd:schema>
```

참조

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:element name="address" type="xsd:token"
                       default="서울시 성북구 월곡동" />
        </xsd:sequence>
      </xsd:restriction></xsd:complexContent></xsd:complexType>
</xsd:redefine>
```

재정의