INT305 Lab 4 XML Schema (v2.0)

XML Schema

- is commonly known as XML Schema Definition (XSD).
- It is used to describe and validate the structure and the content of XML data.
- It defines the elements, attributes, and data types.
- Schema element supports Namespaces

The structure and namespace of an XSD file

Syntax:

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

- An XSD file is XML file. It must start with an XML declaration.
- Root of XSD is <schema> . . . </schema>
- XSD makes use of namespaces because it makes a distinction between code that belongs to XSD and code that refers to the defined elements and attributes.
- xmlns:xs indicates that the elements and data types used in the schema come from the "http://www.w3.org/2001/XMLSchema" namespace.
- Complex XSD files refer to more than one "Schema" namespace.

Association of XSD with an XML file

- An XML document described by a XSD is called an instance document.

Schema-Instance Link w/o Namespace	
XML-Schema	<?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"> <xs:element name="root" type="xs:anyType" /> </xs:schema>
XML-Instance	<?xml version="1.0" encoding="UTF-8"?> <root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="xml-schema.xsd"> Some String content </root>

Schema-instance link without namespace

- The xsi:noNamespaceSchemaLocation attribute defines the URL of XSD file.

Schema-Instance Link with Namespace	
XML-Schema	<?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://porkaew/xml"> <xs:element name="root" type="xs:anyType" /> </xs:schema>
XML-Instance	<?xml version="1.0" encoding="UTF-8"?> <d:root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:d="http://porkaew/xml" xsi:schemaLocation="http://porkaew/xml xml-schema.xsd"> Some String content </d:root>

Shema-instance link with namespace

- Both the XML and the XSD file must contain a namespace declaration for your domain (e.g., http://porkaew/xml)
- targetNamespace indicates that the elements defined by this schema come from the "http://porkaew/xml" namespace.
- The XML file must contain
  - a namespace declaration for XML Schema-instance
  - a xsi:schemaLocation attribute that tells where to file the XSDs. This attribute can have as many "namespace-URL" pairs **("namespace xsd-file")** as you like.

XML Schema Definition (XSD) elements

- An element is the building block of an XML document and is defined within the XSD.
- The three types of XSD shema elements can be defined as:
  - SimpleType contains only text and cannot have other elements or attributes.
  - ComplexType can contain text, elements and attributes. It can be a parent to all the elements and attributes within it.
  - GlobalType is a single type can be defined in the XML document, which can then be used by all other references.
- Elements declared as the direct child of an XML schema are considered global elements and can be used throughout the schema.
- However, element declared within a complexType cannot be used elsewhere in the schema because they are considered local elements.

Element Declaration

Syntax:

<xs:element name= "x" type= "y" default="z" fixed="y" minOccurs="m" maxOccurs="n" />

- name This is the name that will appear in the XML document.
- type This provide the description of what can be contained within the element. There are a number of predefined types such as xs:string, xs:integer, xs:boolean or xs:date
- default If no value specified in the XML document, the application should use the default specified in the XSD.

- fixed — The value in the XML document can only have the value specified in the XSD.
- minOccurs — can be assigned any non-negative integer value (0, 1 (default), 2, 3,…)
- maxOccurs — can be assigned any non-negative integer value (0, 1 (default), 2, 3,…) or "unbounded" (no maximum)

The default values for minOccurs and maxOccurs is 1. Thus, if both minOccurs and maxOccurs attributes are absent, the element must appear once and once only.

Example: A simple element without children and attributes

<xs:element name="customer_dob" type="xs:date" />	<customer_dob> 2000-01-12T12:13:14Z </customer_dob>
<xs:element name="orderID" type="xs:int" />	<orderID>5756</orderID>
<xs:element name="title" type="xs:string" />	<title>XML Schema</title>

Task 1: write element declaration

<firstname>Henry</firstname>	<xs:element name="firstname" type="xs:string" />
<quantity> 100 </quantity> Note: The default is 1	<xs:element name="quantity" type="xs:int" default=1 />
<color> red </color> Note: "red" is the fixed value	<xs:element name="color" type="xs:string" fixed="red" />

Attribute Declaration

Syntax:

<xs:attribute name= "x" type= "y" use="optional (default) | prohibited | required" >

- To declare attributes, you must define the element it belongs to as complexType, since simple elements cannot have attributes.
- The "use" property in the XSD definition specifies whether the attribute is optional or mandatory (by default, it is optional).
- An attribute is typically specified within the XSD definition for an element, this ties the attribute to the element.
- Attributes also can be specified global and then referenced.

Example:

<xs:element name="order"> <xs:complexType> <xs:attribute name="id" type="xs:int" /> </xs:complexType> </xs:element>	<order id="6" /> or <order />
<xs:element name="order"> <xs:complexType> <xs:attribute name="id" type="xs:int" use="required" /> </xs:complexType> </xs:element>	<order id="6" />

The default and fixed attributes can be specified within the XSD attribute specification (in the same way are for elements).

Type Declaration

1. **simpleType**
   - define your own types by modifying existing built-in data type and specifies the constraints and information about the value of attributes or text-only elements
2. **complexType**
   - is a container for other elements and/or attributes.

Declaring simpleType

Syntax: <xs:simpleType name= "newSimpleType" > . . . </xs:simpleType>

Example: a simple element with a custom simple type

Define a simple type directly by naming	Define a simple type and then create an element using type attribute.
<xs:element name="age"> <xs:simpleType> <xs:restriction base="xs:integer"> <xs:minInclusive value="0"/> <xs:maxInclusive value="100"/> </xs:restriction> </xs:simpleType> </xs:element>	<xs:simpleType name="age_type"> <xs:restriction base="xs:integer"> <xs:minInclusive value="0"/> <xs:maxInclusive value="100"/> </xs:restriction> </xs:simpleType> <xs:element name="age" type="age_type" />

- This example defines an element called “age” that is a simple type with a restriction. The value of age can NOT be lower than 0 or greater than 100.
- For more information about the XSD restriction, visit the following URL: https://www.w3schools.com/xml/schema_facets.asp

Declaring complexType
Syntax:

```
<complexType name="newType">
    { Context }
</complexType>
```

where { Context }

```
<compositor>
  { ElementList }
</compositor>
  { AttibuteList }
```

where <compositor> is <all>, <sequence> , <choice>

<sequence minOccurs="m" maxOccurs="n" > . . . </sequence>

Compositors

- There are three types of compositors <xs:sequence> , <xs:choice>, and <xs:all>,.
- These compositors allow you to determine how the child elements within them within the XML document.

Compositor	**Description**
Sequence	The child elements in the XML document MUST appear in the order they are defined in the XSD schema.
Choice	Only one the child elements described in the XSD schema can appear in the XML document.
All	The child elements described in the XSD schema can appear in the XML document in any order.

The <xs:sequence> and <xs:choice> compositors can be nested inside other compositors, and give their own minOccurs and maxOccurs properties. This allows for quite complex combinations to be formed.

Example:

Define a complex type directly by naming	Define a Complex Type and then create element using type attribute.
<xs:element name = "student"> <xs:complexType> <xs:sequence> <xs:element name = "firstname" type = "xs:string"/> <xs:element name = "lastname" type = "xs:string"/> <xs:element name = "nickname" type = "xs:string"/> <xs:element name = "marks" type = "xs:positiveInteger"/> </xs:sequence> <xs:attribute name = 'rollno' type = 'xs:positiveInteger'/> </xs:complexType> <xs:element>	<xs:complexType name = "StudentType"> <xs:sequence> <xs:element name = "firstname" type = "xs:string"/> <xs:element name = "lastname" type = "xs:string"/> <xs:element name = "nickname" type = "xs:string"/> <xs:element name = "marks" type = "xs:positiveInteger"/> </xs:sequence> <xs:attribute name = 'rollno' type = 'xs:positiveInteger'/> </xs:complexType> <xs:element name = 'student' type = 'StudentType' />

```
<student rollno="1102">
    <firstname>Johnson</firstname>
    <lastname>Silver</lastname>
    <nickname>John</nickname>
    <marks>5</marks>
</student>
```

Example: Complex Types: Empty Elements

<xs:element name="eEmpty"> <xs:complexType/> </xs:element>	<d:eEmpty />
<xs:element name="eEmptyWithAtt"> <xs:complexType> <xs:attribute name="att" type="xs:string" /> </xs:complexType> </xs:element>	<d:eEmptyWithAtt att="value" />
<xs:element name="eEmptyWithAtt" type="eEmptyWithAttType" /> <xs:complexType name=eEmptyWithAttType> <xs:attribute name="att" type="xs:string" /> </xs:complexType>	<d:eEmptyWithAtt att="value" /> Give the complexType element a name, and the "eEmptyWithAtt" element have a type attribute that refers to the name of the complexType

Example: Complex Types with Compositors

```
<xs:element name="ePerson">
 <xs:complexType>
  <xs:xsequence>
   <xs:element name="firstname" type="xs:string"/>
   <xs:element name="lastname" type="xs:string"/>
   <xs:choice>
    <xs:element name="phone" type="xs:string"/>
    <xs:element name="address" type="xs:string"/>
   </xs:choice>
  </xs:xsequence>
  <xs:attribute name="code" type="xs:ID"/>
  <xs:attribute name="role" type="xs:string"/>
 </xs:complexType>
</xs:element>
```

```
<d:ePerson code="L123" role="lecturer">
  <firstname>Kriengkrai</firstname>
  <lastname>Porkaew</lastname>
  <address>KMUTT</address>
</d:ePerson>
```

Example: Defining child element with a reference

```
<xs:element name="firstname" type="xs:string"/>
<xs:element name="lastname" type="xs:string"/>
<xs:element name="phone" type="xs:string"/>
<xs:element name="address" type="xs:string"/>

<xs:attribute name="code" type="xs:ID"/>
<xs:attribute name="role" type="xs:string"/>

<xs:element name="ePerson">
  <xs:complexType>
    <xs:sequence>
     <xs:element ref="firstname" />
     <xs:element ref="lastname" />
     <xs:choice>
        <xs:element ref="phone" />
        <xs:element ref="address" />
     </xs:choice>
    </xs:sequence>
    <xs:attribute ref="code" />
    <xs:attribute ref="role" />
  </xs:complexType>
</xs:element>
```

```
<d:ePerson code="L123" role="lecturer">
  <firstname>Kriengkrai</firstname>
  <lastname>Porkaew</lastname>
  <address>KMUTT</address>
</d:ePerson>
```

Task 2:

2.1 Write a complex element

```
<product prodid="1345" />
```

Write your answer below:

```
<xs:element name="product">
 <xs:complexType>
 <xs:attribute name="prodid" type="xs:int" />
 </xs:complexType>
</xs:element>
```

2.2 Write a XSD file and associate it with the below XML file (w/o namespace)

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Write your answer below:

xml-schema.xmd

```
<?xml version="1.0" ?>
<s:schema xmlns:s="http://www.w3.org/2001/XMLSchema">
<s:element name="note">
    <s:complexType>
        <s:sequence>
            <s:element name="to" type="s:string"/>
            <s:element name="from" type="s:string"/>
            <s:element name="heading" type="s:string"/>
            <s:element name="body" type="s:string"/>
        </s:sequence>
    </s:complexType>
</s:element>
</s:schema>
```

xml-instance.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<note xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
      i:noNamespaceSchemaLocation="./xml-schema.xsd"  >
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

2.3 Write a XSD file and associate it with the below XML file (w/o namespace)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- the attribute orderid must provided -->
<shiporder orderid="889923">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <!-- item range 0..unbounded -->
  <item>
    <title>Empire Burlesque</title>
    <!-- note is optional but maxOccurs is 1-->
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  <item>
    <title>Hide your heart</title>
    <quantity>1</quantity>
    <price>9.90</price>
  </item>
</shiporder>
```

Write your answer below:

xml-schema.xmd

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="shiporder">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="orderperson" type="xs:string"/>
            <xs:element name="shipto" type="xs:string"/>
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="name" type="xs:string"/>
                        <xs:element name="address" type="xs:string"/>
                        <xs:element name="city" type="xs:string"/>
                        <xs:element name="country" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="title" type="xs:string"/>
                        <xs:element name="note" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
                        <xs:element name="quantity" type="xs:positiveInteger"/>
                        <xs:element name="price" type="xs:decimal"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="orderid" type="xs:string" use="required" />
    </xs:complexType>
</xs:element>
</xs:schema>
```

xml-instance.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- the attribute orderid must provided -->
<shiporder xmlns:i="http://www.w3.org/2001/XMLSchema-instance"
          i:noNamespaceSchemaLocation="./xml-schema.xsd" orderid="889923">
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <!-- item range 0..unbounded -->
  <item>
    <title>Empire Burlesque</title>
    <!-- note is optional but maxOccurs is 1-->
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  <item>
    <title>Hide your heart</title>
    <quantity>1</quantity>
    <price>9.90</price>
  </item>
</shiporder>
```

Reference:

- http://edutechwiki.unige.ch/en/XML_Schema_tutorial_-_Basics
- https://www.codeguru.com/java/xsd-tutorial-xml-schemas-for-beginners/
- https://www.w3schools.com/xml/schema_intro.asp
- https://www.w3schools.com/xml/el_attribute.asp
- https://www.w3schools.com/xml/schema_elements_ref.asp
- https://www.techtarget.com/whatis/definition/XSD-XML-Schema-Definition