XML SCHEMA

Well-Formed is Not Enough

- Must begin with the XML declaration
- Must have one unique root element
- Start-tags must have matching end-tags
- Elements are case sensitive
- All elements must be closed
- All elements must be properly nested
- All attribute values must be quoted

Introduction

- An XML schema describes the structure of an XML document.
- The XML Schema language is also referred to as XML Schema Definition (XSD).

Purpose of an XML Schema

- Elements and attributes that can appear in a document
- Number of (and order of) child elements
- Data types for elements and attributes
- Default and fixed values for elements and attributes

XML Schemas are the Successors of DTDs

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

```
<!ELEMENT note (to, from,
heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

Sample XML Schema

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"> <xs:element name="note">
 <xs:complexType>
   <xs:sequence>
     <xs:element name="to" type="xs:string"/>
     <xs:element name="from" type="xs:string"/>
     <xs:element name="heading" type="xs:string"/>
     <xs:element name="body" type="xs:string"/>
   </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

Sample XML

```xml
<?xml version="1.0"?>

<note xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="note.xsd">
<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</note>
```

XML Schemas use XML Syntax

- You don't have to learn a new language
- You can use your XML editor to edit your Schema files
- You can use your XML parser to parse your Schema files
- You can manipulate your Schema with the XML DOM
- You can transform your Schema with XSLT

Root Element

The <schema> element is the root element of every XML Schema

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="https://www.w3schools.com"
xmlns="https://www.w3schools.com"
elementFormDefault="qualified">
...
...
</xs:schema>
```

`xmlns:xs` indicates that the elements and data types used in the schema come from the "http://www.w3.org/2001/XMLSchema" namespace.

`targetNamespace` indicates where the elements defined by this schema come from.

`elementFormDefault` indicates that any elements used by the XML instance document which were declared in this schema must be namespace qualified.

XML Simple Element

- XML Schemas define the elements of your XML files.
- A simple element is an XML element that contains only text. It cannot contain any other elements or attributes.

```
<lastname>Smith</lastname>
```

- The syntax for defining a simple element is:

```
<xs:element name="xxx" type="yyy"/>
```

XML Simple Element

- XML Schema has a lot of built-in data types. The most common types are:
  - xs:string
  - xs:decimal
  - xs:integer
  - xs:boolean
  - xs:date
  - xs:time

Example

XML

```
<lastname>Refsnes</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>
```

SCHEMA

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

XSD Complex Elements

A complex element is an XML element that contains other elements and/or attributes.

There are four kinds of complex elements:

1. empty elements

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

2. elements that contain only text and attribute

```
<food type="dessert">Ice cream</food>
```

3. elements that contain other elements and may or may not have attributes

```
<employee>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</employee>
```

4. elements that contain both other elements and text

```
<description>
  It happened on <date lang="norwegian">03.03.99</date>
</description>
```

Complex Empty Elements

An empty complex element cannot have contents, only attributes.

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

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

Complex Text-Only Elements

This type contains only simple content (text and attributes), therefore we add a simpleContent element around the content. When using simple content, you must define an extension OR a restriction within the simpleContent element.

```
<shoesize country="france">35</shoesize>
```

Use the extension/restriction element to expand or to limit the base simple type for the element.

```
<xs:element name="shoesize">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:integer">
        <xs:attribute name="country" type="xs:string" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Complex Elements Only

An "elements-only" complex type contains an element that contains only other elements.

```
<person id="1">
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</person>

<xs:element name="person”>
 <xs:complexType>
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
  <xs:attribute name=“id" type="xs:string" />
 </xs:complexType>
</xs:element>
```

Complex Types with Mixed Content

An XML element, "letter", that contains both text and other elements:

```
<letter>
  Dear Mr.<name>John Smith</name>.
  Your order <orderid>1032</orderid>
  will be shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

The following schema declares the "letter" element:

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

XSD Indicators

- There are seven indicators:
  - Order indicators:
    - All
    - Choice
    - Sequence
  - Occurrence indicators:
    - maxOccurs
    - minOccurs
  - Group indicators:
    - Group name
    - attributeGroup name

Order Indicators

Order indicators are used to define the order of the elements

All Indicator

The <all> indicator specifies that the child elements can appear in any order, and that each child element must occur only once:

```
<xs:element name="person">
  <xs:complexType>
    <xs:all>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

Order Indicators

Choice Indicator

The <choice> indicator specifies that either one child element or another can occur:

```
<xs:element name="person">
  <xs:complexType>
    <xs:choice>
      <xs:element name=“teaching" type="employee"/>
      <xs:element name=“non-teaching" type="member"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Order Indicators

Sequence Indicator

The <sequence> indicator specifies that the child elements must appear in a specific order:

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Occurrence Indicators

Occurrence indicators are used to define how often an element can occur.

maxOccurs Indicator

The <maxOccurs> indicator specifies the maximum number of times an element can occur:

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string" maxOccurs="10"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The example above indicates that the "child_name" element can occur a minimum of one time (the default value for minOccurs is 1) and a maximum of ten times in the "person" element. To allow an element to appear an unlimited number of times, use the maxOccurs="unbounded" statement

Occurrence Indicators

minOccurs Indicator

The <minOccurs> indicator specifies the minimum number of times an element can occur:

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"/>
      <xs:element name="child_name" type="xs:string" maxOccurs="10" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

The example above indicates that the "child_name" element can occur a minimum of zero times and a maximum of ten times in the "person" element.

Group Indicators

Group indicators are used to define related sets of elements.

Element Groups

You must define an all, choice, or sequence element inside the group declaration. The following example defines a group named "persongroup", that defines a group of elements that must occur in an exact sequence:

```
<xs:group name="persongroup">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="birthday" type="xs:date"/>
  </xs:sequence>
</xs:group>
```

Group Indicators

After you have defined a group, you can reference it in another definition, like this:

```
<xs:element name="person" type="personinfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:group ref="persongroup"/>
    <xs:element name="country" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Group Indicators

Attribute Groups

The following example defines an attribute group named "personattrgroup":

```
<xs:attributeGroup name="personattrgroup">
  <xs:attribute name="name" type="xs:string"/>
  <xs:attribute name="birthday" type="xs:date"/>
</xs:attributeGroup>
```

After you have defined an attribute group, you can reference it in another definition, like this:

```
<xs:element name="person">
  <xs:complexType>
    <xs:attributeGroup ref="personattrgroup"/>
  </xs:complexType>
</xs:element>
```

END OF LECTURE