XML Schema

An XML Schema:

- defines elements that can appear in a document
- defines attributes that can appear within elements
- defines which elements are child elements
- defines the sequence in which the child elements can appear
- defines the number of child elements
- defines whether an element is empty or can include text
- defines default values for attributes

The purpose of a Schema is to define the legal building blocks of an XML document, just like a DTD.

XML Schema – Better than DTDs

XML Schemas

- are easier to learn than DTD
- are extensible to future additions
- are richer and more useful than DTDs
- are written in XML
- support data types

Example: Shipping Order

```xml
<?xml version="1.0"?>
<shipOrder>
    <shipTo>
        <name>Svendson</name>
        <street>Oslo St</street>
        <address>400 Main</address>
        <country>Norway</country>
    </shipTo>
    <items>
        <item>
            <title>Wheel</title>
            <quantity>1</quantity>
            <price>10.90</price>
        </item>
        <item>
            <title>Cam</title>
            <quantity>1</quantity>
            <price>9.90</price>
        </item>
    </items>
</shipOrder>
```

XML Schema for Shipping Order

```
<xsd:schema xmlns:xsd=http://www.w3.org/1999/XMLSchema>

<xsd:element name="shipOrder" type="order"/>

<xsd:complexType name="order">
  <xsd:element name="shipTo" type="shipAddress"/>
  <xsd:element name="items" type="cdItems"/>
</xsd:complexType>

<xsd:complexType name="shipAddress">
  <xsd:element name="name" type="xsd:string"/>
  <xsd:element name="street" type="xsd:string"/>
  <xsd:element name="address" type="xsd:string"/>
  <xsd:element name="country" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="cdItems">
  <xsd:element name="item" minOccurs="0" maxOccurs="unbounded" type="cdItem
</xsd:complexType>

<xsd:complexType name="cdItem">
  <xsd:element name="title" type="xsd:string"/>
  <xsd:element name="quantity" type="xsd:positiveInteger"/>
  <xsd:element name="price" type="xsd:decimal"/>
</xsd:complexType>

</xsd:schema>
```

Purchase Order – A more detailed example

- *Instance document*: An XML document that conforms to an XML Schema
- Elements that contain sub-elements or carry attributes are said to have *complex types*
- Elements that contain numbers (and strings, and dates, etc.) but do not contain any sub-elements are said to have *simple types*
- *Attributes* always have simple types

```xml
<?xml version="1.0"?>
<purchaseOrder orderDate="1999-10-20">

  <shipTo country="US">
    <name>Alice Smith</name>
    <street>123 Maple Street</street>
    <city>Mill Valley</city>
    <state>CA</state>
    <zip>90952</zip>
  </shipTo>

  <billTo country="US">
    <name>Robert Smith</name>
    <street>8 Oak Avenue</street>
    <city>Old Town</city>
    <state>PA</state>
    <zip>95819</zip>
  </billTo>

  <comment>Hurry, my lawn is going wild!</comment>

  <items>
    <item partNum="872-AA">
      <productName>Lawnmower</productName>
      <quantity>1</quantity>
      <USPrice>148.95</USPrice>
      <comment>Confirm this is electric</comment>
    </item>

    <item partNum="926-AA">
      <productName>Baby Monitor</productName>
      <quantity>1</quantity>
      <USPrice>39.98</USPrice>
      <shipDate>1999-05-21</shipDate>
    </item>
  </items>

</purchaseOrder>
```

Defining the USAddress Type

```
<xsd:complexType name="USAddress" >
  <xsd:sequence>
    <xsd:element name="name"   type="xsd:string"/>
    <xsd:element name="street" type="xsd:string"/>
    <xsd:element name="city"   type="xsd:string"/>
    <xsd:element name="state"  type="xsd:string"/>
    <xsd:element name="zip"    type="xsd:decimal"/>
  </xsd:sequence>
  <xsd:attribute name="country" type="xsd:NMTOKEN" fixed="US"/>
</xsd:complexType>
```

The `PurchaseOrderType` definition contains element declarations involving complex types. `<xsd:element name="comment" type="xsd:string"/>`

The comment element is globally defined under the schema element.

```
<xsd:complexType name="PurchaseOrderType">
  <xsd:sequence>
    <xsd:element name="shipTo" type="USAddress"/>
    <xsd:element name="billTo" type="USAddress"/>
    <xsd:element ref="comment" minOccurs="0"/>
    <xsd:element name="items"  type="Items"/>
  </xsd:sequence>

  <xsd:attribute name="orderDate" type="xsd:date"/>

</xsd:complexType>
```

`Items` type defined below (note that we have an unnamed type):

```
<xsd:complexType name="Items">
<xsd:sequence>
  <xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="productName" type="xsd:string"/>
        <xsd:element name="quantity">
          <xsd:simpleType>
            <xsd:restriction base="xsd:positiveInteger">
              <xsd:maxExclusive value="100"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="USPrice"  type="xsd:decimal"/>
        <xsd:element ref="comment"   minOccurs="0"/>
        <xsd:element name="shipDate" type="xsd:date" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="partNum" type="SKU" use="required"/>
    </xsd:complexType>
  </xsd:element>
</xsd:sequence>
</xsd:complexType>
```

Same figure repeated below (if you want to cut and paste)

```
<xsd:complexType name="Items">
  <xsd:sequence>
    <xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="productName" type="xsd:string"/>
          <xsd:element name="quantity">
            <xsd:simpleType>
              <xsd:restriction base="xsd:positiveInteger">
                <xsd:maxExclusive value="100"/>
              </xsd:restriction>
            </xsd:simpleType>
          </xsd:element>
          <xsd:element name="USPrice"  type="xsd:decimal"/>
          <xsd:element ref="comment"   minOccurs="0"/>
          <xsd:element name="shipDate" type="xsd:date" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="partNum" type="SKU" use="required"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

`SKU` type

```
<!-- Stock Keeping Unit, a code for identifying products -->
<xsd:simpleType name="SKU">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{3}-[A-Z]{2}"/>
  </xsd:restriction>
</xsd:simpleType>
```

- The above type restricts the SKU code to start with 3 digits followed by a “-” followed by 2 upper-case letters.
- The earlier example of restricting a simple type was “quantity” with a sub-type of 1 to 99.
- Restriction of a simple type starts with a “base” simple type and using a facet such as “pattern”, elements are restricted to a subset.

Complete XML Schema Specification for Purchase Order document:

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

  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Purchase order schema for Example.com.    Copyright 2000
      Example.com. All rights reserved.
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name="purchaseOrder" type="PurchaseOrderType"/>

  <xsd:element name="comment" type="xsd:string"/>

  Complex Type PurchaseOrderType goes here
  Complex Type USAddress goes here
  Complex Type Items goes here
  Simple Type SKU goes here

</xsd:schema>
```

Deriving New Simple Types

A large collection of built-in types are available in XML Schema

```
xsd:string
xsd:integer
xsd:positiveInteger,
xsd:decimal
xsd:boolean
xsd:date
xsd:NMTOKENS
etc.
```

https://www.w3.org/TR/xmlschema-1/

https://www.w3.org/TR/xmlschema-2/

We have seen two examples: SKU and Quantity.

The following example defines myInteger (value between 10000 and 99999) using `minInclusive` and `maxInclusive` facets:

```
<xsd:simpleType name="myInteger">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="10000"/>
    <xsd:maxInclusive value="99999"/>
  </xsd:restriction>
</xsd:simpleType>
```

Enumeration facet:

```
<xsd:simpleType name="USState">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="AK"/>
    <xsd:enumeration value="AL"/>
    <xsd:enumeration value="AR"/>
    <!-- and so on ... -->
  </xsd:restriction>
</xsd:simpleType>
```

List Types

XML Schema has 3 built-in list types: NMTOKENS, IDREFS, ENTITIES

Creating new list types from simple types:

```
<xsd:simpleType name="listOfMyIntType">
  <xsd:list itemType="myInteger"/>
</xsd:simpleType>
```

The following XML fragment conforms to the above SimpleType:

```
<listOfMyInt>20003 15037 95977 95945</listOfMyInt>
```

Several facets can be applied to list types:

```
length
minLength
maxLength
enumeration
```

For example, to define a list of exactly six US states (SixUSStates),

- First define a new list type called `USStateList` from `USState`.

- Then derive `SixUSStates` by restricting `USStateList` to only six items

```
<xsd:simpleType name="USStateList">
  <xsd:list itemType="USState"/>
</xsd:simpleType>

<xsd:simpleType name="SixUSStates">
  <xsd:restriction base="USStateList">
    <xsd:length value="6"/>
  </xsd:restriction>
</xsd:simpleType>
```

The following data will conform to the above type:

```
<sixStates>PA NY CA NY LA AK</sixStates>
```

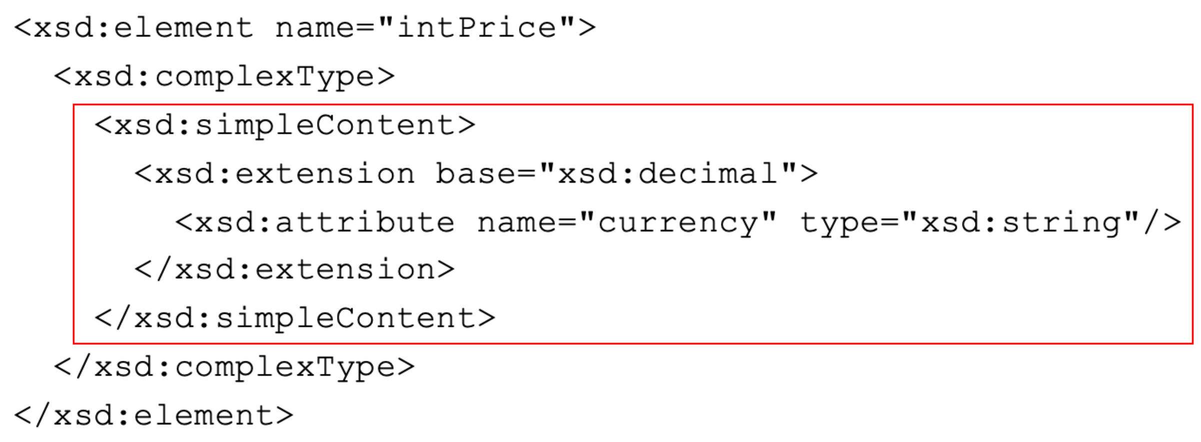
Deriving Complex Types from Simple Types

So far we have seen how to introduce “attributes” in elements of Complex Types.

How to declare an element that has simple content and an attribute as well such as:

```
<intPrice currency="EUR">423.46</intPrice>
```

This is done as follows:



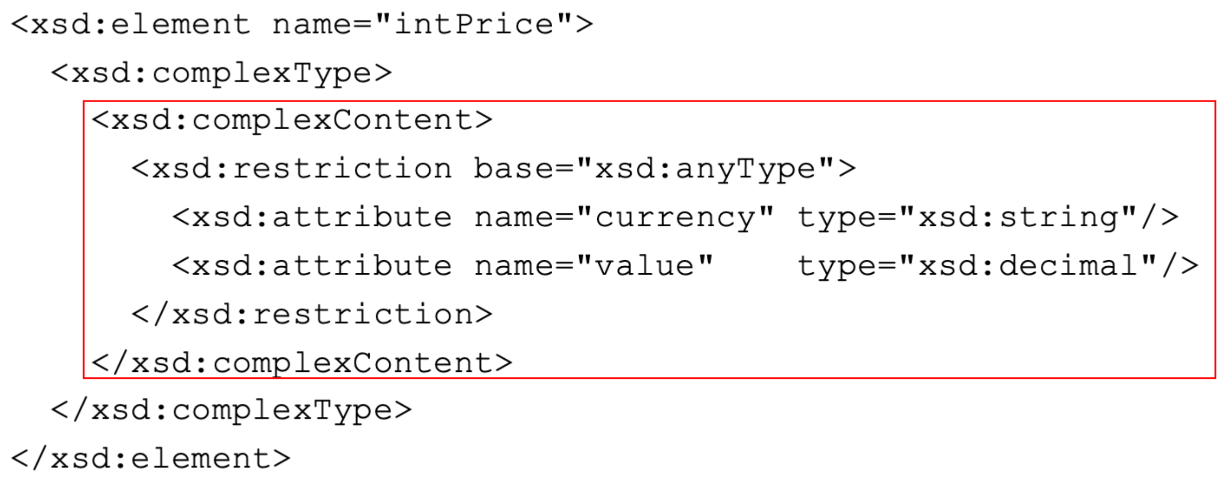
```
<xsd:element name="intPrice">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:decimal">
        <xsd:attribute name="currency" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

Same figure repeated below (if you want to cut and paste)

```
<xsd:element name="intPrice">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:decimal">
        <xsd:attribute name="currency" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

How to declare an empty element with one or more attributes such as:

```
<intPrice currency="EUR" value="423.46"/>
```



```
<xsd:element name="intPrice">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:restriction base="xsd:anyType">
        <xsd:attribute name="currency" type="xsd:string"/>
        <xsd:attribute name="value"    type="xsd:decimal"/>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

Same figure repeated below (if you want to cut and paste)

```
<xsd:element name="intPrice">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:restriction base="xsd:anyType">
        <xsd:attribute name="currency" type="xsd:string"/>
        <xsd:attribute name="value"    type="xsd:decimal"/>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

XML Schema - Summary

- A flexible and powerful schema language
- Syntax is XML itself
- Variety of data types and ability to extend type system
- Variety of data “facets” and “patterns” to impose domain constraints
- Can define advanced constraints such as “primary key” and “referential integrity”