# Web Basics - XML

Lesson 3: XML Schema Definition

# Lesson Objectives

- In this lesson, you will learn about:
  - Advantages of Schema over DTD
  - Method to write a schema definition for an XML file
  - Data types used in schemas
  - Simple and Complex type of elements
  - Restrictions on XSD elements
  - Indicator – Order, Occurrence, and Group

# Introduction to XML Schema

- The XML Schema Definition Language is an XML language for describing and constraining the content of XML documents
- XML Schema is a W3C recommendation
- XML Schema defines what it means for an XML document to be valid
- XML Schema are a radical departure from Document Type Definitions (DTDs), the existing schema mechanism inherited from SGML

# XML Schemas

- Drawbacks of DTD:
  - Use of non-XML syntax
  - No support for data typing
  - Non-extensibility

# Why Use XML Schemas?

- XML Schemas
  - support data types
  - use XML syntax
  - secure data communication
  - are extensible
  - Well-Formed is not enough

# XML Schema

- An XML Schema defines:
  - Elements that can appear in a document
  - Attributes that can appear in a document
  - The elements that are child elements
  - The order of child elements
  - The number of child elements
  - The criteria whether an element is empty or can include text
  - Data types for elements and attributes
  - Default and fixed values for elements and attributes

# Namespaces

- XML Namespaces provide a method to avoid element name conflicts
- Name Conflicts: In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications
- XML Namespaces provides a method to avoid element name conflicts

# Namespaces

```
<table>
 <tr>
<td>Apples</td>
<td>Bananas</td>
</tr>
</table>
```

```
<table>
<name>African Coffee
Table</name>
<width>80</width>
<length>120</length>
</table>
```

```
<tables>
      <table> …..</table>
      <table> ….</table>
</tables>
```

How do you differentiate between these table?

# Namespaces

- The namespace attribute is placed in the start tag of an element and has the following syntax:

  xmlns:namespace-

  prefix="namespace“
- The W3C namespace specification states that the namespace itself should be an Uniform Resource Identifier (URI)
- When a namespace is defined in the start tag of an element, all child elements with the same prefix are associated with the same namespace

# Solving the Name Conflict Using a Prefix

- Code Snippet

```
<root>
<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td><h:td>Bananas</h:td>
  </h:tr>
</h:table>
<f:table xmlns:f="http://www.w3schools.com/furniture">
```

# Solving the Name Conflict Using a Prefix

- Code Snippet continued

```
<f:name>African Coffee Table</f:name>
<f:width>80</f:width><f:length>120</f:length>
</f:table>
 </root>
```

# Illustration(Message.xsd)

- Let us see an example on writing a schema definition:

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

# Illustration(Message.xsd)

- Code Snippet continued

```
<xs:element name="subject" type="xs:string"/>
<xs:element name="text" type="xs:string"/>
<xs:attribute name="priority" type="xs:string"      use="required"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

# Using XSD in XML Document

- Example:

```
<note xmlns="http://www.w3.org/2001/XMLSchema "
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=“message.xsd">
```

# XML-Schema Definition

- Simple Element:
  - <xs:element name="title" type="xs:string"/>
- where “title” is the name of the element and “xs:string” is the data type of the element
- Specifying default or fixed values:
  - <xs:element name="title" type="xs:string" default="No Title"/>
  - <xs:element name="category" type="xs:string" fixed="Common"/>

# XML Schema Data Types

- XML Schema Data Types belongs to following categories:
  - XSD String: String data types are used for values that contains character strings.
  - XSD Date: Date and time data types are used for values that contain date and time.
  - XSD Numeric: Numeric data types are used for numeric values
  - XSD Misc: Other miscellaneous data types like boolean, base64Binary, hexBinary, float, double, etc.

# String Data Types

- String Data Type:
  - <xs:element name=" Author" type="xs:string"/>
  - <Author>John Smith</Author>
- NormalizedString Data Type:
  - <xs:element name="Author“ type="xs:normalizedString"/>
  - <Author>John Smith</Author>
- Token Data Type:
  - <xs:element name="Author" type="xs:token"/>
  - <Author>John Smith</Author>

# Date and Time Data Types

- Date Data Type:

  <xs:element name="publishdate" type="xs:date"/>

  < publishdate>2002-09-24</ publishdate>

- Time Data Type:

  <xs:element name="publishtime" type="xs:time"/>

  < publishtime>09:00:00</ publishtime>

- DateTime Data Type:

  <xs:element name=" publishdatetime" type="xs:dateTime"/>

  < publishdatetime>2002-05-30T09:00:00</ publishdatetime>

# Numeric Data Types

- Decimal Data Type:

```
<xs:element name="price" type="xs:decimal"/>
        <price>999.50</price> or
        <price>+999.5450</price> or
        <price>-999.5230</price>
```

- Integer Data Type:

```
<xs:element name="price" type="xs:integer"/>
        <price>999</price> Or
        <price>+999</price> Or
        <price>-999</price>
```

# Miscellaneous Data Types

- Boolean Data Type:

```
<xs:element name="disabled” type="xs:boolean"/>
        <disabled>true</disabled>
```

- Binary Data Types:

```
<xs:element name="blobsrc" type="xs:hexBinary"/>
```

- AnyURI Data Type:

```
<xs:element  name=“PicSrc” type="xs:anyURI"/>
        <PicSrc>"http://www.w3schools.com/images/smiley.gif" </ PicSrc >
```

# Attribute in XSD

- Defining an Attribute:

  <xs:attribute name=“AuthorID” type=“xs:string"/>

  where “AuthorID” is the name of the attribute and “xs:string” specifies the data type of the attribute.

- Creating Optional and Required Attributes:

  <xs:attribute name=“btype" type="xs:string" use="required"/>

  Attributes are optional by default

# Complex Type Element

- Illustration:

```
<xs:element name=“book">
<xs:complexType>
  <xs:sequence>
    <xs:element name=“title" type="xs:string"/>
    <xs:element name=“author" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

# Simple Type Element

- Illustration:

```
<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>
```

# XSD Restrictions in a Nutshell

- Let us see some of the restrictions on XSD Elements:

| Constraint | Description |
|---|---|
| Enumeration | Defines a list of acceptable values |
| FractionDigits | Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero. |
| Length | Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero. |
| MaxExclusive | Specifies the upper bounds for numeric values (the value must be less than this value) |
| MaxInclusive | Specifies the upper bounds for numeric values (the value must be less than or equal to this value) |
| MaxLength | Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero. |

# Restriction on Values

- Example

```
<xs:element name=“Quantity">
    <xs:simpleType>
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="0"/>
            <xs:maxInclusive value=“500"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

# Restriction on Set Values

- Example

```
<xs:element name=“Category">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value=“Dot Net/>
<xs:enumeration value=“BI"/>
<xs:enumeration value=“RDBMS"/>
<xs:enumeration value=“J2EE"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
```

# Restrictions on Series of Values

- To limit the content of an XML element to define a series of numbers or letters that can be used, we can use the pattern constraint.

```
<xs:element name="letter">
<xs:simpleType>
  <xs:restriction base="xs:string">
  <xs:pattern value="[a-z]"/>
</xs:restriction>
</xs:simpleType>
 </xs:element>
```

- The only acceptable value is ONE of the LOWERCASE letters from a to z
- The “Category” element is a simple type with a restriction.
- The acceptable values are Dot Net, BI, RDBMS, and J2EE

# Restrictions on Series of Values

- **Some more examples of Pattern**

| | |
|---|---|
| [a-zA-Z][a-zA-Z][a-zA-Z] | THREE of the LOWERCASE OR UPPERCASE letters from a to z |
| [0-9]{10} | Any 10 digit number |
| [A-Z][0-9]{3} | 1 uppercase letter followed by 3 digits |
| [0-9][0-9][0-9][a-zA-Z]* | 3digits followed by any number of uppercase or lowercase letters |
| EMP[#_!] | ‘EMP’ followed by 1 # or ! Or _ |

- The “Category” element is a simple type with a restriction.
- The acceptable values are Dot Net, BI, RDBMS, and J2EE

# Types of Indicators

- We have seven types of indicators:
  - Order indicators:
    - All
    - Choice
    - Sequence
- Occurrence indicators:
  - maxOccurs
  - minOccurs
- Group indicators:
  - Group name
  - attributeGroup name

# All Indicator

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

```
<xs:element name=“book">
 <xs:complexType>
 <xs:all>
<xs:element name=“title" type="xs:string"/>
<xs:element name=“author" type="xs:string"/>
</xs:all>
</xs:complexType>
</xs:element>
```

# 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="employee" type="employee"/>
<xs:element name="member" type="member"/>
</xs:choice>
</xs:complexType>
</xs:element>
```

# Sequence Indicator

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

```
<xs:element name=“book">
<xs:complexType>
  <xs:sequence>
    <xs:element name=“title" type="xs:string"/>
    <xs:element name=“author" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

# maxOccurs Indicator

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

```
<xs:element name=“book">
<xs:complexType>
  <xs:sequence>
    <xs:element name=“title" type="xs:string"/>
    <xs:element name=“author" type="xs:string"/>
    <xs:element name=“vendor" type="xs:string“ maxOccurs=“2"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

# minOccurs Indicator

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

```
<xs:element name="book">
<xs:complexType>
  <xs:sequence>
    <xs:element name="title" type="xs:string"/>
    <xs:element name="author" type="xs:string"/>
    <xs:element name="vendor" type="xs:string"      maxOccurs="2"
minOccurs="0" />
  </xs:sequence>
</xs:complexType>
</xs:element>
```

# Group Indicators

- Group Indicators:
  - Group indicators are used to define related sets of elements
- Element Groups:
  - Element groups are defined with the group declaration, as shown below:

```
<xs:group name="groupname">  ... </xs:group>
```

# Group Indicators (Contd)

```
<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>
```

```
<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>
```

# Demo on XML-Schema Definition

- Demo on:
  - Run Validator
  - Code to be validated
  - Schema file
  - Shiporder.xsd (schema File)
  - Shiporder.xml
    (xml Document)

# Summary

- In this lesson, you have learnt:
  - A schema describes the arrangement of markup and character data within a valid XML document
  - The purpose of XML Schema and XML DTD is same
  - Currently, only Microsoft IE5 supports XML Schema
  - XML Schema vocabulary defines different elements

# Review Question

- Question 1: List any four valid datatypes in XML Schema: ___, ___, ___, and ___.

- Question 2:The elements defined in a schema come from this namespace:
  - Option 1 : sourceNamespace
  - Option 2: targetNamespace
  - Option 3: cannot be specified

- Question 3: Choice is an Occurrence indicator.
  - True/False