Markup Languages

Lecture 4.
XML Schema

Introduction to XML Schema

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

- XML Schema became a W3C Recommendation 02. May 2001.

XML Schemas are the Successors of DTDs

- XML Schemas are extensible to future additions
- XML Schemas are richer and more powerful than DTDs
- XML Schemas are written in XML
- XML Schemas support data types
- XML Schemas support namespaces

XSD How To?

XML:

```
<?xml version="1.0"?>
<?xml-stylesheet ... ?>
<catalog>
<cd>
  <title>Empire Burlesgue
    </title>
  <artist>Bob Dylan
    </artist>
</cd>
<cd>
  <title>Live A Paris</title>
  <artist>Celin Dion
    </artist>
</cd>
</catalog>
```

DTD:

```
<!ELEMENT catalog (cd+)>
<!ELEMENT cd (title, artist)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT artist (#PCDATA)>
```

XML Schema:

```
<xs:element name="catalog">
<xs:complexType>
 <xs:sequence>
  <xs:element name="cd“ maxoccurs="unbounded">
  <xs:complexType>
   <xs:sequence>
    <xs:element name=“title"  type=“xs:string“/>
    <xs:element name=“artist"  type=“xs:string“/>
   <xs:sequence>
  </xs:complexType>
  </xs:element>
 </xs:sequence>
</xs:complexType>
</xs:element>
```

Compare References to DTD and XML Schema

A Reference to a DTD:

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

<!DOCTYPE catalog SYSTEM
"cd.xsd">

<catalog>
<cd>
  <title>Empire Burlesgue</title>
  <artist>Bob Dylan</artist>
</catalog>
```

A Reference to an XML Schema:

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

<catalog xmlns:xsi="http://
www.w3.org/2001/XMLSchema-
instance"
xsi:schemaLocation="cd.xsd">
<cd>
  <title>Empire Burlesgue</title>
  <artist>Bob Dylan</artist>
</catalog>
```

XML Schema

XML-file: cd.xml

```
<?xml version="1.0">
<catalog
   xmlns:xsi="http://www.w3.org/
   2001/XMLSchema-instance"
   xsi:schemaLocation="cd.xsd">
 <cd>
  <title>empire burlesque</title>
  <artist>bob dylan</artist>
 </cd>
 ...
</catalog>
```

XSD-file: cd.xsd

```
<?xml version="1.0"?>
<xs:schema
   xmlns:xs="http://www.w3.org/
        2001/XMLSchema">
...
</xs:schema>
```

XSD Elements

- **Simple Element**
  - A simple element is an XML element that can contain only text. It cannot contain any other elements or attributes.

XSD Simple Elements

- The text can be of many different types. It can be one of the types included in the XML Schema definition (boolean, string, date, etc.), or it can be a custom type that you can define yourself.
- The syntax for defining a simple element is:
  - `<xs:element name="name_element" type="name_type"/>`
- 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

Simple Elements Example

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

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

Default and Fixed Values for Simple Elements

- Simple elements may have a default value OR a fixed value specified.
    - **A default value** is automatically assigned to the element when no other value is specified.
    - Example:
      ```
      <xs:element name="color" type="xs:string"
      default="red"/>
      ```
    - **A fixed value** is also automatically assigned to the element, and you cannot specify another value.
    - Example:
      ```
      <xs:element name="color" type="xs:string"
      fixed="red"/>
      ```

Task

- Write XSD rules for next bold elements:

```
<person>
  <surname>Ivanova</surname>
  <name>Elena</name>
  <email>elene@gmail.com</email>
  <gender>female</gender>
  <date_birth>04.06.2013</date_birth>
  <marital_status>married</marital_status>
  <cell>89123456789</cell>
</person>
<objective>
   <post>programmer</post>
  <salary>100000</salary>
</objective>
```

XSD Attributes

- Simple elements cannot have attributes. If an element has attributes, it is considered to be of a complex type. But the attribute itself is always declared as a simple type.
- <xs:attribute name="name_attr" type="name_type"/>

- Example:

  <lastname **lang="EN"**>Smith</lastname>

  <xs:attribute name="lang" type="xs:string"/>

Default and Fixed Values for Attributes

- Attributes may have a default value OR a fixed value specified.
  - A default value is automatically assigned to the attribute when no other value is specified.
  - Example:
    ```
    <xs:attribute name="lang" type="xs:string"
    default="EN"/>
    ```
  - A fixed value is also automatically assigned to the attribute, and you cannot specify another value.
  - Example:
    ```
    <xs:attribute name="lang" type="xs:string"
    fixed="EN"/>
    ```

Optional and Required Attributes

- Attributes are optional by default. To specify that the attribute is required, use the "use" attribute:
  - <xs:attribute name="lang" type="xs:string" **use="required"**/>

Restrictions on Content

- When an XML element or attribute has a data type defined, it puts restrictions on the element's or attribute's content.
- Example: If an XML element is of type "xs:date" and contains a string like "Hello World", the element will **not validate**.

XSD Restrictions/Facets

- With XML Schemas, you can also add your own restrictions to your XML elements and attributes. These restrictions are called **facets**.

Restrictions on Values

- The following example defines an element called "age" with a restriction. The value of age cannot be lower than 0 or greater than 120:

-
  ```
  <xs:element name="age">
    <xs:simpleType>
      <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="120"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  ```

Restrictions on a Set of Values

- To limit the content of an XML element to a set of acceptable values, we would use the enumeration constraint.

-
  ```
  <xs:element name="car">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="Audi"/>
        <xs:enumeration value="Golf"/>
        <xs:enumeration value="BMW"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  ```

Restrictions on a Series of Values

- To limit the content of an XML element to define a series of numbers or letters that can be used, we would use the **pattern constraint**.
- The example below defines an element called "letter" with a restriction. The only acceptable value is ONE of the LOWERCASE letters from a to z:
-
```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrictions on a Series of Values

- The next example defines an element called "initials" with a restriction. The only acceptable value is THREE of the UPPERCASE letters from a to z:
-
```
<xs:element name="initials">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z][A-Z][A-Z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrictions on a Series of Values

- The next example also defines an element called "initials" with a restriction. The only acceptable value is THREE of the LOWERCASE OR UPPERCASE letters from a to z:

-
  ```
  <xs:element name="initials">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="[A-Z][A-Z][A-Z]"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  ```

Restrictions on a Series of Values

- The next example defines an element called "choice" with a restriction. The only acceptable value is ONE of the following letters: x, y, OR z:
-
```
<xs:element name="choice">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[xyz]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrictions on a Series of Values

- The next example defines an element called "prodid" with a restriction. The only acceptable value is FIVE digits in a sequence, and each digit must be in a range from 0 to 9:

-
  ```
  <xs:element name="prodid">
    <xs:simpleType>
      <xs:restriction base="xs:integer">
        <xs:pattern value="[0-9][0-9][0-9][0-9][0-9]"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  ```

Other Restrictions on a Series of Values

- The example below defines an element called "letter" with a restriction. The acceptable value is zero or more occurrences of lowercase letters from a to z:
-
```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="([a-z])*"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Other Restrictions on a Series of Values

- The next example also defines an element called "letter" with a restriction. The acceptable value is one or more pairs of letters, each pair consisting of a lower case letter followed by an upper case letter. For example, "sToP" will be validated by this pattern, but not "Stop" or "STOP" or "stop":
-
```
<xs:element name="letter">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="([a-z][A-Z])+"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Other Restrictions on a Series of Values

- The next example defines an element called "gender" with a restriction. The only acceptable value is male OR female:

-
  ```
  <xs:element name="gender">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="male|female"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  ```

Other Restrictions on a Series of Values

- The next example defines an element called "password" with a restriction. There must be exactly eight characters in a row and those characters must be lowercase or uppercase letters from a to z, or a number from 0 to 9:

-
  ```
  <xs:element name="password">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="[a-zA-Z0-9]{8}"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>>
  ```

Restrictions on Length

- To limit the length of a value in an element, we would use the length, maxLength, and minLength constraints.
- This example defines an element called "password" with a restriction. The value must be exactly eight characters:
-
```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrictions on Length

- This example defines another element called "password" with a restriction. The value must be minimum five characters and maximum eight characters:
-
```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="5"/>
      <xs:maxLength value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrictions on Whitespace Characters

- To specify how whitespace characters should be handled, we would use the whiteSpace constraint.
- This example defines an element called "address" with a restriction. The whiteSpace constraint is set to "preserve", which means that the XML processor WILL NOT remove any white space characters:
-
```
<xs:element name="address">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="preserve"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrictions on Whitespace Characters

- This example also defines an element called "address" with a restriction. The whiteSpace constraint is set to "replace", which means that the XML processor WILL REPLACE all white space characters (line feeds, tabs, spaces, and carriage returns) with spaces:
-
```
<xs:element name="address">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="replace"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrictions for Datatypes

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
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters that are acceptable
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled

Task

- Write XSD rules for next bold elements:

```
<person>
  <surname>Ivanova</surname>
  <name>Elena</name>
  <email>elene@gmail.com</email>
  <gender>female</gender>
  <date_birth>04.06.2013</date_birth>
  <marital_status>married</marital_status>
  <cell>8(912)345-67-89</cell>
</person>
<objective>
   <post>programmer</post>
  <salary>100000</salary>     5 000 <= salary < 500 000
</objective>
```

XSD Complex Elements

- A complex element is an XML element that contains other elements and/or attributes.
- Examples:
  - <product pid="1345"/>
  - <employee>
      <firstname>John</firstname>
      <lastname>Smith</lastname>
    </employee
  - <food type="dessert">Ice cream</food>
  - <description>
       It happened on
       <date lang="norwegian">03.03.99</date>
    </description>

How to Define a Complex Element

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

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

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

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

XSD Elements Only

-
  ```
  <person>
    <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:complexType>
  </xs:element>
  ```

XSD Order Indicators: Sequence

- 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>
```

XSD Order Indicators: all

- **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>
  ```

XSD Order Indicators: choice

- 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>
```

Task

```
<contacts>
  <tel>123-45-67</tel>
  <address>Lenina, 76</address>
  <email>123@mail.ru</email>
</contacts>
```

Require only one element

```
<info>
  <surname>Ivanova</surname>
  <name>Elena</name>
</info>
```

Require all elements

Occurrence Indicators

- **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 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.

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.

Occurrence Indicators

- To allow an element to appear an unlimited number of times, use the **maxOccurs="unbounded"** statement:
- <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" **maxOccurs="unbounded"** />
    </xs:sequence>
  </xs:complexType>
</xs:element>

XSD Empty Elements

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

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

XSD Text-Only Elements

- A complex text-only element can contain text and attributes.
- 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, like this:

-
  ```
  <xs:element name="somename">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="basetype">
          ....
          ....
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  ```

```
<xs:element name="somename">
  <xs:complexType>
    <xs:simpleContent>
      <xs:restriction base="basetype">
        ....
        ....
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Example Text-Only Elements

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

- XML schema:

```
<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>
```

XSD Mixed Content

- A mixed complex type element can contain attributes, elements, and text.

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

-
  ```
  <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>
  ```
- To enable character data to appear between the child-elements of "letter", the mixed attribute must be set to "true“.

Task

```
<contacts>
  <tel>123-45-67</tel>
  <address>Lenina, 76</address>
  <email>123@mail.ru</email>
  <email>123@mail.ru</email>
</contacts>

<photo url="image.jpg" />
```

$\}$ Require all elements