

XML Schema

Assoc. Prof. Dr. Kanda Runapongsa Saikaew
(krunapon@kku.ac.th)

Dept. of Computer Engineering
Faculty of Engineering
Khon Kaen University



Overview

- The “schema” element
- Referencing a schema in an XML document
- Simple and complex types
- Element and attribute declarations
- Occurrence constraints
- Content models
- Annotations



Sample XML (nation.xml)

```
<?xml version="1.0"?>
<nation id="th">
    <name>Thailand</name>
    <location>Southeast Asia</location>
</nation>
```



XML Schema Example: nation_ns.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://campus.en.kku.ac.th"
    xmlns="http://campus.en.kku.ac.th"
    elementFormDefault="qualified">
    <xsd:element name="nation"
        type="nationType"/>
    <xsd:complexType name="nationType">
        <xsd:sequence>
            <xsd:element name="name"
                type="xsd:string"/>
            <xsd:element name="location"
                type="xsd:string"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:ID"/>
    </xsd:complexType>
</xsd:schema>
```



What does A Schema Consist of?

- A “schema” element
- A variety of sub-elements
 - “element” elements

Example:

```
<xsd:element name="nation"  
type="nationType"/>
```

- “complexType” elements

Example:

```
<xsd:complexType  
name="nationType">...</xsd:complexType>
```

- “simpleType” elements



The “schema” Element

```
<?xml version="1.0"?>  
<xsd:schema  
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
    targetNamespace="http://campus.en.kku.ac.th"  
    xmlns="http://campus.en.kku.ac.th"  
    elementFormDefault="qualified">
```



Attributes of “schema” (1/4)

- `xmlns:xsd="http://www.w3.org/2001/XMLSchema"`
 - Indicates that the elements and data types used in the schema (`schema`, `element`, ...) come from the <http://www.w3.org/2001/XMLSchema> namespace
 - The elements and data types that come from the <http://www.w3.org/2001/XMLSchema> namespace should be prefixed with `xsd:`



Attributes of “schema” (2/4)

- targetNamespace="http://campus.en.kku.ac.th"
 - Indicates that the elements defined by this schema (nation, name, location, etc.) are in the “http://campus.en.kku.ac.th” namespace



Attributes of “schema” (3/4)

- `xmlns="http://campus.en.kku.ac.th"`
 - Indicates that the default namespace is `http://campus.en.kku.ac.th`
- `elementFormDefault="qualified"`
 - Indicates that any elements used by the XML instance document which were declared in this schema must be namespace qualified

Attributes of “schema” (4/4)

- **attributeFormDefault="unqualified"**
 - Indicates that any attributes used by the XML instance document may not be namespace qualified
- The default values of "attributeFormDefault" and "elementFormDefault" are "unqualified"



Referencing a Schema in nation_ns.xml

```
<?xml version="1.0"?>
<nation
  xmlns:xsi=
  "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
  "http://campus.en.kku.ac.th nation_ns.xsd"
  xmlns="http://campus.en.kku.ac.th" id="th">
  <name>Thailand</name>
  <location>Southeast Asia</location>
</nation>
```



Referencing a Schema with Namespace

- XML Schema Instance namespace is
 - `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`
- We can use the `schemaLocation` attribute which has two values
 - The first value is the namespace to use
 - The second value is the location of the XML schema to use
- `xsi:schemaLocation="http://campus.en.kku.ac.th nation_ns.xsd"`



Schema without Namespace (nation_no_ns.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="nation"
    type="nationType"/>
  <xsd:complexType name="nationType">
    <xsd:sequence>
      <xsd:element name="name"
        type="xsd:string"/>
      <xsd:element name="location"
        type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID"/>
  </xsd:complexType>
</xsd:schema>
```



Referencing a Schema in nation_no_ns.xml

```
<?xml version="1.0"?>
  <nation
    xmlns:xsi=
      "http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation=
        "nation_no_ns.xsd" id="th">
    <name>Thailand</name>
    <location>Southeast Asia</location>
  </nation>
```



Referencing a Schema without Namespace

- Suppose that we want to refer to a Schema that it is just a file, we use “noNamespaceSchemaLocation” attribute

```
<nation
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
    xsi:noNamespaceSchemaLocation=
```

```
    "nation_no_ns.xsd">
```



Schema Data Types

- **Simple type**
 - Do not have sub-elements
 - Do not have attributes
 - Predefined type or derived from predefined type
 - Example: <name>Thailand</name>
- **Complex type**
 - Have either sub-elements or attributes
 - Example:
<nation ..><name>...</name></nation>



Definition and Declaration

- Definition
 - Create **new types** (both simple and complex types)
- Declaration
 - Enable elements and attributes with specific names and types (both simple and complex) to appear in document instances



anyType

- Base type from which all simple and complex types are derived
- Does not contain its contents in any way
- Default type when no type is specified
 - `<xsd:element name="anything" type="xsd:anyType"/>` is same as
`<xsd:element name="anything"/>`
- Use more constrained types whenever possible



Common Built-in Simple Data Types

- string, normalizedString

Example: <element name="Title" type="string"/>

- int, long, short, unsignedInt

Example: <element name="numItems" type="int"/>

- time, dateTime, date, duration

• Example: <element name="from" type="time"/>



Defining New Simple Types (1/2)

- New simple types are defined by deriving them from existing simple types (built-in's and derived)
- In particular, we can derive a new simple type by restricting an existing simple type
- Example:

```
<xsd:simpleType name="passwordType">
    <xsd:restriction base="xsd:string">
        <xsd:length value="8"/>
    </xsd:restriction>
</xsd:simpleType>
```



Defining New Simple Types (2/2)

- We use the “simpleType” element to define and name the new simple type

- Example:

```
<xsd:simpleType name="passwordType">
```

- We use the “restriction” element to indicate the existing (base) type, and to identify the “facets” that constrain the range of values

- Example:

```
<xsd:restriction base="xsd:string">  
    <xsd:length value="8"/>  
</xsd:restriction>
```



Restrictions/Facets

- Restrictions are used to control acceptable values for XML elements or attributes
- Many kinds of restrictions/facets
 - Restrictions on values
 - Restrictions on patterns
 - Restrictions on a set of values



Restrictions on Values

- We can restrict the range of the base type by employing two facets called “minInclusive” and “maxInclusive”
- In XML Schema:

```
<xsd:element name="myInteger" type="myIntegerType"/>
  <xsd:simpleType name="myIntegerType">
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="99999"/>
    </xsd:restriction>
  </xsd:simpleType>
```

- In XML:

```
<myInteger>12345</myInteger>
```



Restriction on Patterns (1/2)

- To limit the content to a series of numbers or letters, use the pattern constraint
- In XML Schema:

```
<xsd:simpleType name="SKU">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{3}-[A-Z]{2}"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:element name="item" type="SKU"/>
```

- In XML:

```
<item>872-AA</item>
```



Restrictions on Patterns (2/2)

- To define a “password” type that has 8 characters which each of them is either a character or a digit
- In XML Schema:

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

- In XML:

```
<password>1qaz2wsx</password>
```



Restriction 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
- In XML Schema:

```
<xsd:simpleType name="provinceType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="Chiang Rai"/>
    <xsd:enumeration value="Chiang Mai"/>
    <xsd:enumeration value="Khon Kaen"/>
  </xsd:restriction>
</xsd:simpleType>
```

- In XML:

```
<province>Chiang Rai</province>
```



Restrictions on Lengths

- We can also specify the minimum and maximum lengths of values using “minLength” and “maxLength” attributes
- In XML Schema:

```
<xsd:simpleType name="passwordType">
  <xsd:restriction base="xsd:string">
    <xsd:length value="8"/>
    ...
</xsd:simpleType>
<xsd:element name="password" type="passwordType"/>
```

- In XML: <password>1q2w3e4r</password>

Declaring a Simple Element

- The syntax for declaring a simple element

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

- Where xxx is the element name and yyy is the data type of the element
- Example:

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



Declaring an Attribute

- The syntax for declaring a simple element

```
<xsd:attribute name="xxx" type="yyy"/>
```

- Where xxx is the attribute name and yyy is the data type of the attribute
- Example:

```
<xsd:attribute name="id"  
type="xsd:ID"/>
```



Attributes

- All attributes are declared as simple types
- Only complex elements can have attributes
- An element with attributes always has a complex type definition
- Example:

```
<xsd:complexType name="nationType">  
    ...  
    <xsd:attribute name="id" type="xsd:ID"/>  
</xsd:complexType>
```



Default & Fixed Values

- A default value is automatically assigned to
 - The element when no value is specified
 - The attribute when no attribute is specified
- A fixed value is automatically assigned to elements
 - We cannot specify another element value



Default Values of Elements & Attributes

- Default values of both attributes and elements are declared using the “default” attribute
- Default attribute values apply when attributes are missing
- Default element values apply when elements are empty



Default Element Value Example

- The simple empty element which its default value specified in XML schema

- In XML Schema:

```
<xsd:element name="price"  
type="xsd:decimal" default="100.0"/>
```

- In XML:

- <price/>

```
<!-- has the same effect as  
<price>100.00</price> -->
```



Default Attribute Value Example

- The attribute which its default value specified in XML schema

- In XML Schema:

```
<xsd:element name="nation">
```

```
...<
```

```
<xsd:attribute name="country"
type="xsd:NMTOKEN" default="Thailand"/>
</xsd:element>
```

- In XML:

```
<nation>...</nation>
```

has the same effect as

```
<nation country="Thailand">...</nation>
```



Fixed Attribute Value Example

- The attribute which its value is fixed

- In XML Schema:

```
<xsd:element name="nation">
```

...

```
    <xsd:attribute name="country"  
      type="xsd:NMTOKEN" fixed="Thailand"/>  
</xsd:element>
```

- In XML:

```
<nation>...</nation>
```

has the same effect as

```
<nation country="Thailand">...</nation>
```

cannot write this

```
<nation country="USA">...</nation>
```



Element Declaration Reference (1/2)

- The element declarations have described so far have each associated a name with an existing type definition
- Sometimes it is preferable to use an existing element rather than declare a new element
- Example:

```
<staff>
```

```
    <manager><name>r</name></manager>
```

```
    <employee><name>e</name></employee>
```

```
</staff>
```



Element Declaration Reference (2/2)

- The declaration references an existing element, comment, that was declared elsewhere in the schema
- The value of “ref” attribute must reference a global element which is the element directly under `<schema>` root element

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

```
<element name="manager">  
    <xsd:complexType>
```

...

```
    <xsd:element ref="name"/>
```

...

```
<element>
```



Element Declaration Reference Example

```
<xsd:schema>
...
<xsd:element name="comment" type="xsd:string"/>
<xsd:complexType name="PurchaseOrderType">
<xsd:sequence>
...
<xsd:element ref="comment"/>
</xsd:sequence>
<xsd:attribute name="orderDate" type="xsd:date"/>
</xsd:complexType>
</xsd:schema>
```



Complex Elements

- A complex element is an XML element that contains other elements and/or attributes
- Several kinds of complex elements
 - Empty elements
 - Elements that contain only sub-elements
 - Elements that contain both sub-elements and text



Complex Type Definitions

- New complex types are defined using the “complexType” element
 - Example: <xs:complexType name=“nationType”>
- The definitions contain
 - Element declarations
 - Example: <xs:element name=“name” type=“xs:string”/>
 - Element references
 - Example: <xs:element ref=“location”/>
 - Attribute declarations
 - Example: <xs:attribute name=“id” type=“xs:ID”/>



Defining a Complex Type

```
<xsd:complexType name="nationType">
  <xsd:sequence>
    <xsd:element name="name"
      type="xsd:string"/>
    <xsd:element name="location"
      type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="id"
    type="xsd:ID"/>
</xsd:complexType>
```



Sharing the Complex Type

- Defining the “address” type

```
<xsd:complexType  
    name="ThaiAddress"></xsd:complexType>
```

Sharing the “address” type

```
<xsd:complexType name="purchaseOrderType">  
    <xsd:sequence>  
        <xsd:element name="shipTo"  
        type="ThaiAddress"/>  
        <xsd:element name="billTo"  
        type="ThaiAddress"/>  
    </xsd:sequence>  
</xsd:complexType>
```



Explicit Type vs. Anonymous Type

- Explicit type
 - One in which **a name** is given to the type
 - Element that uses the type is generally defined in a different section of the file
 - Object-oriented in that same explicit type is used as the type for several different elements
- Implicit type (nameless, anonymous)
 - Use when the type is not needed by multiple elements



Example of Explicit Type

- When defining an explicit type, “complexType” or “simpleType” must “name” attribute

```
<xsd:complexType  
      name="ThaiAddress"></xsd:complexType>
```

- When declaring an element, we can use the defined type

```
<xsd:element name="shipTo"  
type="ThaiAddress"/>
```



Anonymous Type Definitions

- A type can be more succinctly defined as an anonymous type
 - Saves the overhead of having to be named and explicitly referenced
- In general, you can identify anonymous types by
 - The lack of a “type=” in an element (or attribute) declaration and
 - The presence of an un-named (simple or complex) type definition



Example of Anonymous Type

```
<xsd:element name="item" minOccurs="0"
  maxOccurs="unbounded">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="productName"
        type="xsd:string"/>
      <xsd:element
        name="quantity"
        type="xsd:int"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```



Occurrence Constraints

- We can control the number of occurrences of elements by using “minOccurs” and “maxOccurs” attributes
- The default values of these attributes are 1
- The values of these attributes are nonnegative integers
- The value of “maxOccurs” attribute can also be “unbounded”



Constraints for an Optional Element

- DTD: An optional element, use symbol ?

```
<!ELEMENT purchaseOrderType (billTo, shipTo,  
comment?, items)>
```

- XML Schema: use minOccurs="0"

```
<xsd:complexType  
name="purchaseOrderType2">  
  <xsd:sequence>  
    <xsd:element ref="comment"  
      minOccurs="0"/>  
  </xsd:sequence>  
</xsd:complexType>
```



Constraints for Multiple Occurrence (1/2)

- DTD: A multiple occurrence element, use *
- XML Schema: use minOccurs="0" and maxOccurs="unbounded"

```
<xsd:complexType name="items">
  <xsd:sequence>
    <xsd:element name="item"
      minOccurs="0"
      maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```



Constraints for Multiple Occurrence (2/2)

- DTD: A multiple occurrence element , use +
`<!ELEMENT Items (item+)>`
- XML Schema → maxOccurs=“unbounded”
`<xsd:element name=“item”
maxOccurs=“unbounded”/>`
- DTD: An element with min appearance = 2 max appearance = 5
→ cannot do it
- XML Schema
`<xsd:element name=“item” minOccurs=“2”
maxOccurs=“5”/>`



Occurrences of Attributes

- Attributes may appear once or not at all
- Possible values of “use” attribute
 - required
 - optional
 - fixed
 - default
- The default value of “use” attribute is “optional”
 - Example: `<xs:attribute name="id" type="xs:ID"/>` has the same effect as
`<xs:attribute name="id" type="xs:ID" use="optional"/>`



Element Content

- How content of an element gets constructed
- Three different ways
 - Complex types from simple types
 - Mixed content
 - Elements mixed with character content
 - Empty content



Complex Types from Simple Types

- In XML:

```
<USPrice>345.67</USPrice>
```

- In XML Schema:

```
<xsd:element name="USPrice"  
type="xsd:decimal"/>
```

- In XML:

```
<internationalPrice currency="BAHT">  
345.23</internationalPrice>
```

- Should we define the type for internationalPrice as **complexType** or **simpleType**?



Derive Complex Types from Simple Types

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



Mixed Content1

- Mixed content contains both sub-elements and character data

- In XML:

```
<salutation>
  Dear Dr.<name>Prawase Wasi</name>
</salutation>
```

- In XML Schema:

```
<xsd:element name="salutation">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element name="name"
type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```



Mixed Content2

- In XML:

```
<quote>This is an interesting quote
  <text>To live is to choose ...</text>
  <by>Kofi Anan</by>
</quote>
```

- In XML Schema:

```
<xsd:element name="quote">
  <xsd:complexType mixed="true">
    <xsd:choice minOccurs="0"
maxOccurs="unbounded">
      <xsd:element name="text" type="xsd:string"/>
      <xsd:element name="by" type="xsd:string"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```



Empty Content1

- In XML:

```
<internationalPrice currency="BAHT" value="345.23"/>
```

- In XML Schema:

```
<xsd:element name="internationalPrice">
  <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>
```



Empty Content2

- In XML:

```
<internationalPrice currency="BAHT"  
value="345.23"/>
```

- In XML Schema:

```
<xsd:element name="internationalPrice">  
    <xsd:complexType>  
        <xsd:attribute name="currency"  
type="xsd:string"/>  
        <xsd:attribute name="value"  
type="xsd:decimal"/>  
    </xsd:complexType>  
</xsd:element>
```



simpleContent vs. complexContent

- simpleContent indicates that the content model of the new type contains only character data and no element declaration
- complexContent indicates that the content model of the new type contains
 - Sub-elements
 - Sub-elements and character data (mixed content)
 - Nothing (empty content)
- A complex type defined without complexContent interpreted as shorthand for complex content that restricts anyType



“simpleContent” Element Example

- In XML:

```
<amount currency="BAHT">1000</amount>
```

- XML Schema:

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

“complexContent” Element Example (1/2)

```
<xsd:complexType name="bookType">
    <xsd:sequence>
        <xsd:element name="foreword"
type="xsd:string"/>
        <xsd:element name="tableContent"
type="xsd:string"/>
        <xsd:element name="bookContent"
minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```



“complexContent” Element Example (2/2)

```
<xsd:complexType name="bookExtendType">
  <xsd:complexContent>
    <xsd:extension base="bookType">
      <xsd:sequence>
        <xsd:element
          name="bookBib"
          type="xsd:string"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```



Extension vs. Restriction

- Schema gives you these derivation options:
 - extension:
 - Add elements to the end of the children
 - Add additional attributes
 - restriction:
 - Put additional constraints on the value
 - Restricting the possible values or occurrences of sub-elements



Restriction for simpleType Example

- Put additional constraints on the value

```
<xsd:simpleType name="pageType">  
    <xsd:restriction base="xsd:integer">  
        <xsd:minInclusive value="10"/>  
        <xsd:maxInclusive value="5000"/>  
    </xsd:restriction>  
</xsd:simpleType>
```



Restriction of complexType Example

- Restricting the possible values or occurrences of sub-elements from minOccurs="0" to minOccurs = "2" and maxOccurs="unbounded" to maxOccurs="5"

```
<xsd:complexType name="bookRestrictType">
  <xsd:complexContent>
    <xsd:restriction base="bookType">
      <xsd:sequence>
        <xsd:element name="foreword" type="xsd:string"/>
        <xsd:element name="tableContent"
type="xsd:string"/>
        <xsd:element name="bookContent" type="xsd:string"
          minOccurs= "2"
          maxOccurs= "5"/>
      </xsd:sequence>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```



Building Content Models

- XML Schema enables groups of elements to be defined and named
 - Elements can be used to build up the content models of complex types
- The group elements “sequence”, “all”, and “choice” control how elements in the group should appear



“sequence” Element

- “sequence” specifies that the elements in the group must appear in the same order (sequence) as they are declared
- XML Schema

```
<xsd:element name="purchaseOrder">  
  <xsd:complexType>  
    <xsd:sequence>  
      <xsd:element name="shipTo" type="xsd:string"/>  
      <xsd:element name="billTo" type="xsd:string"/>  
    </xsd:sequence>  
  </xsd:complexType>  
</xsd:element>
```



“sequence” Element XML Example

- Valid XML fragment

```
<purchaseOrder>
  <shipTo>
    Khon Kaen
  </shipTo>
  <billTo>
    Chiang Rai
  </billTo>
</purchaseOrder>
```

- Invalid XML fragment

```
<purchaseOrder>
  <billTo>
    Chiang Rai
  </billTo>
  <shipTo>
    Khon Kaen
  </shipTo>
</purchaseOrder>
```



“choice” Element

- “choice” allows only one of its children to appear in an instance

```
<xsd:complexType name="currencyType">
  <xsd:choice>
    <xsd:element name="thaiBaht"
      type="xsd:decimal"/>
    <xsd:element name="usDollar"
      type="xsd:decimal"/>
  </xsd:choice>
</xsd:complexType>
```



“choice” Element XML Example

- Valid XML fragment

```
<currency>  
  <thaiBaht>30.82  
  </thaiBaht>  
</currency>
```

- Invalid XML fragment

```
<currency>  
  <thaiBaht>30.82  
  </thaiBaht>  
  <usDollar>1  
  </usDollar>  
</currency>
```



“all” Element

- “all” indicates that all the elements in the group may appear once and they may appear in any order

```
<xsd:complexType name="nameType">  
  <xsd:all>  
    <xsd:element name="fname"  
      type="xsd:string"/>  
    <xsd:element name="lname"  
      type="xsd:string"/>  
  </xsd:all>  
</xsd:complexType>
```



“all” Element XML Example

- Invalid XML Fragment
- Valid XML Fragment

```
<name>  
  <fname>Panya  
  </fname>  
</name>
```

```
<name>  
  <fname>Panya  
  </fname>  
  <lname>Nirankul  
  </lname>  
</name>
```



“group” Element (1/2)

- Define related sets of elements
- Naming the group so that we can reference it in another complex type definition

```
<xsd:group name="shipAndBill">  
    <xsd:sequence>  
        <xsd:element name="shipTo"  
            type="xsd:string"/>  
        <xsd:element name="billTo"  
            type="xsd:string"/>  
    </xsd:sequence>  
</xsd:group>
```



“group” Element (2/2)

```
<xsd:element name="purchaseOrder">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:group ref="shipAndBill"/>
      <xsd:element name="items"
        type="xsd:decimal"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```



XML Doc which Refers XSD with group

```
<purchaseOrder  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
    xsi:schemaLocation="http://campus.en.kku.ac.th group.xsd"  
    xmlns="http://campus.en.kku.ac.th">  
    <shipTo>Khon Kaen</shipTo>  
    <billTo>Chiang Rai</billTo>  
    <items>2</items>  
</purchaseOrder>
```



“attributeGroup” Element

- Define related sets of attributes
- Naming the group so that we can reference it in another complex type definition

```
<xsd:attributeGroup name="bookDescription">  
    <xsd:attribute name="bookID"  
        type="xsd:ID"/>  
    <xsd:attribute name="numberPages"  
        type="xsd:integer"/>  
</xsd:attributeGroup>
```



“attributeGroup” Element

```
<xsd:element name="book">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="bookTitle"
        type="xsd:string"/>
      <xsd:element name="bookContent"
        type="xsd:string"/>
    </xsd:sequence>
    <xsd:attributeGroup
      ref="bookDescription"/>
  </xsd:complexType>
</xsd:element>
```



XML Doc which Refers XSD with attrGroup

```
<?xml version="1.0" encoding="UTF-8"?>
<book
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://campus.en.kku.ac.th
        attrGroup.xsd"
    xmlns="http://campus.en.kku.ac.th" bookID="xmlws"
    numberPages="500">
    <bookTitle>XML and Web Services</bookTitle>
    <bookContent>Having fun and learning</bookContent>
</book>
```



Nil Values

- Sometimes it is desirable to represent a “null” value being sent to or from a relational database with an element that is present

- XML Schema:

```
<xsd:element name="shipDate"  
type="xsd:date"  
nillable="true"/>
```

- XML:

```
<shipDate xsi:nil="true"></shipDate>
```



Annotations (1/2)

- XML Schema provides three elements for annotating schemas for the benefit of both human readers and applications
- These three elements are “annotation”, “documentation”, and “appinfo”



Annotations (2/2)

- “annotation” element has two element children:
 - “appinfo” - typically for additional application information (e.g., relational tables, constraint rules, Java method mappings)
 - “documentation” - intended to contain the human-readable documentation for the construct

Annotations Example1

```
<xsd:annotation>
    <xsd:documentation xml:lang="en">
        Purchase order schema for abc.com
        Copyright 2004 abc.com
    </xsd:documentation>
</xsd:annotation>
```



Annotations Example2

```
<xsd:annotation>
    <xsd:documentation xml:lang="en">
        Purchase order schema for abc.com
        Copyright 2004 abc.com
    </xsd:documentation>
</xsd:annotation>
<xsd:annotation>
    <xsd:appinfo>
        <java:method name="setBingo"/>
    </xsd:appinfo>
    <xsd:documentation
xmlns="http://www.w3.org/1999/xhtml">
        <p>This element represents a bingo game.</p>
    </xsd:documentation>
</xsd:annotation>
```



“import” Element

- As schemas become larger, it is often desirable to divide their content among several schema documents for purposes such as ease of maintenance, reuse, and readability
- You can use an import element to bring in definitions and declarations from an imported schema into the current schema.
- The imported schema can come from a different namespace than the current schema does.
- You can add multiple import elements to an XML schema, however, prefixes and namespaces have to unique amongst the imported schemas.



“import” Element Syntax

- The import element is used to add multiple schemas with different target namespace to a document.
- Syntax
 - `<import id=ID namespace=anyURI
schemaLocation=anyURI any attributes>
(annotation?)
</import>`



Example: XML Data Model of a Company

- Imagine a project which involves creating a model of a company using XML Schemas
- A company is comprised of people and products
- The company information is described in Company.xsd
- The person information is described in Person.xsd
- The product information is described in Product.xsd



Product.xsd

```
<?xml version="1.0"?>
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.product.org"
    xmlns="http://www.product.org"
    elementFormDefault="unqualified"> <xsd:complexType
        name="ProductType">    <xsd:sequence>
            <xsd:element name="Type"
                type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```



Person.xsd

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.person.org"
  xmlns="http://www.person.org"
  elementFormDefault="unqualified">
  <xsd:complexType name="PersonType">
    <xsd:sequence>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="SSN" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```



Company.xsd (1/2)

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.company.org"
    xmlns="http://www.company.org"
    elementFormDefault="unqualified"
    xmlns:per="http://www.person.org"
    xmlns:pro="http://www.product.org">
    <xsd:import namespace="http://www.person.org"
        schemaLocation="Person.xsd"/>
    <xsd:import namespace="http://www.product.org"
        schemaLocation="Product.xsd"/>
    <xsd:element name="Company">
```



Company.xsd (2/2)

```
<xsd:complexType>
    <xsd:sequence>
        <xsd:element name="Person"
type="per:PersonType"
            maxOccurs="unbounded"/>
        <xsd:element name="Product"
type="pro:ProductType"
            maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```



Homogenous: Product.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.company.org"
  xmlns="http://www.company.org"
  elementFormDefault="qualified">
  <xsd:complexType name="ProductType">
    <xsd:sequence>
      <xsd:element name="Type" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```



Summary

- W3C Schema Elements
 - element, attribute
 - simpleType, complexType
 - simpleContent, complexContent
 - minOccurs, maxOccurs
 - enumeration, list, union
 - sequence, choice, all, group, attributeGroup
 - import
 - annotation, documentation, appinfo



References

- XML Schema Part 0: Primer Second Edition
W3C Recommendation 28 October 2004
<http://www.w3.org/TR/xmlschema-0/>
- XML Schema Part 1: Structures
<http://www.w3.org/TR/xmlschema-1/>
- XML Schema Part 2: Datatypes
<http://www.w3.org/TR/xmlschema-2/>
- XML Schema: Best Practices
<http://www.xfront.com/ZeroOneOrManyNamespaces.html>

