Agenda

- XML Overview
- XML Tree Structure
- XML Syntax Rules
- XML Elements
- XML Attributes
- Validating XML Documents

XML Overview (1 of 2)

- XML:
  - Is a markup language for documents containing structured information.
  - Is used to describe, store, and transport data.
  - Is pure information wrapped in user-defined tags.
  - Was created so that richly structured documents could be used over the internet.

XML Overview (2 of 2)

- Design Goals for XML:
  - XML shall be straightforwardly usable over the Internet.
  - XML shall support a wide variety of applications.
  - XML shall be compatible with SGML.
  - It shall be easy to write programs that process XML documents.
  - The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
  - XML documents should be human-legible and reasonably clear.
  - The XML design should be prepared quickly.
  - The design of XML shall be formal and concise.
  - XML documents shall be easy to create.
  - Terseness in XML markup is of minimal importance.

XML Tree Structure (1 of 3)

- Elements in XML documents form a logical tree structure.
- XML tree structure illustrates the hierarchy and locality of the elements in a XML document.
- XML tree structure can help in showing which elements are the descendants and ancestors of each element.

i Refer to the Classroom.xml sample code.

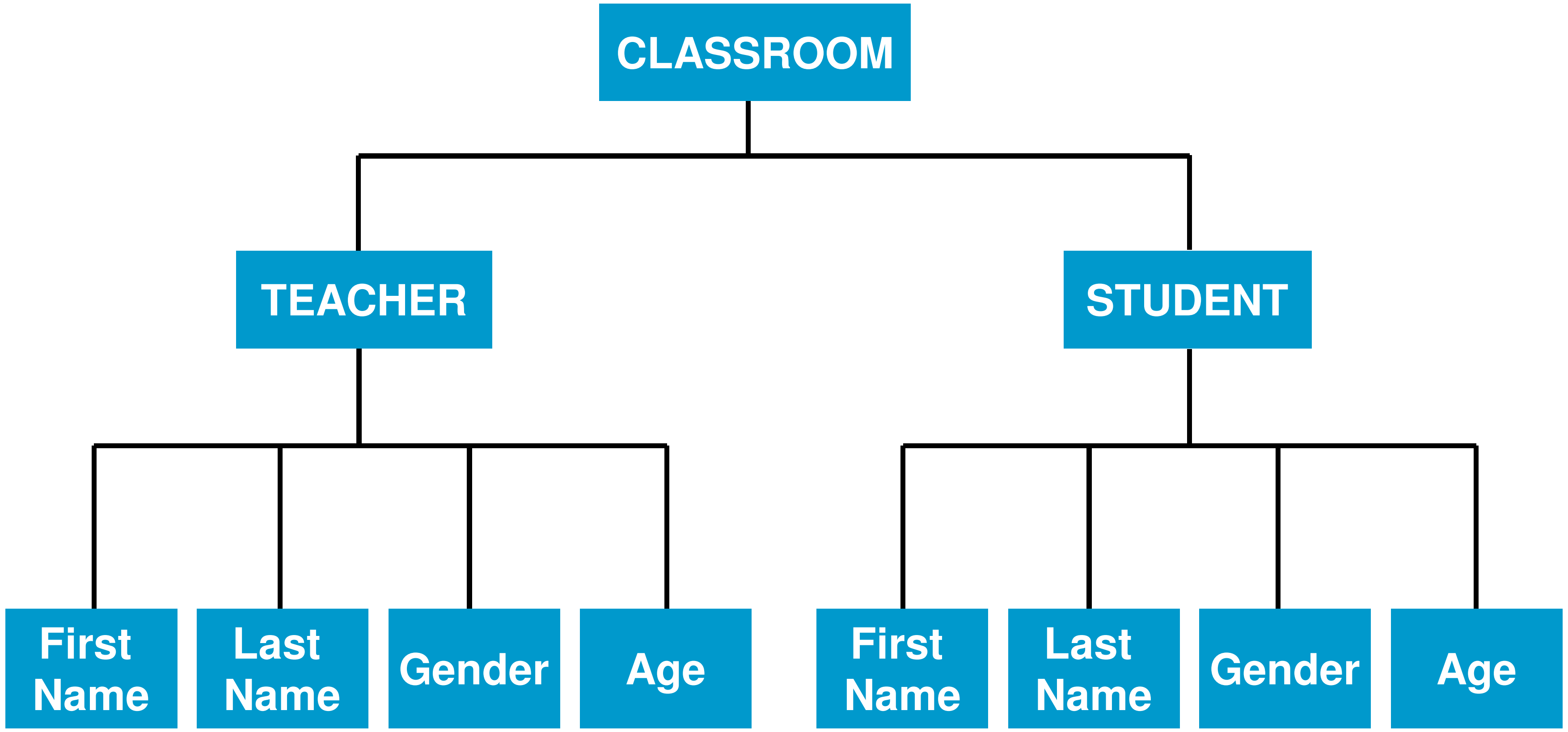
XML Tree Structure (2 of 3)

- Sample code:

```
   <?xml version="1.0" ?>
   <classroom>
   <teacher>
        <first_name>Victoria</first_name>
        <last_name>Brooke</last_name>
        <gender>Female</gender>
        <age>30</age>
   </teacher>
   <student>
        <first_name>Michael</first_name>
        <last_name>Rogers</last_name>
        <gender>Male</gender>
        <age>18</age>
   </student>
</classroom>
```

XML Tree Structure (3 of 3)

- XML Tree of previous code:



XML Syntax Rules (1 of 5)

- All XML documents should begin with an XML declaration. The XML declaration is a processing instruction that identifies the document as being XML.

Example :

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
```

- There are no predefined tags in XML, users have to make their own tags.
- XML documents must have exactly one root element, also known as the document element.
- Basic syntax for XML elements:

Syntax :

```
<element_name>element value</element_name>
```

Examples :

```
<first_name>Jason</first_name>
<book_title>My Favorite Book</book_title>
```

XML Syntax Rules (2 of 5)

- All XML elements must have a corresponding closing tag.

Invalid :
```
<some_tag>some value….
```

Valid :
```
<some_tag>some value placed here</some_tag>
```

- XML tags are case sensitive.

Invalid :
```
<Item_Name>7 Tonner Rice</item_name>
```

Valid :
```
<item_name>7 Tonner Rice</item_name>
```

XML Syntax Rules (3 of 5)

- XML elements must be properly nested

Invalid :

```
<parent_element>
       <child_element>Some value here</parent_element>
       <child_element2>Some value here  </child_element>
</child_element2>
```

Valid :

```
<parent_element>
       <child_element>Some value here</child_element>
       <child_element2>Another value here</child_element2>
</parent_element>
```

XML Syntax Rules (4 of 5)

- XML attribute values must be placed within quotes

Invalid :
```
<some_tag attribute1=attributeValue>….</some_tag>
```

Valid :
```
<some_tag attribute1=“attributeValue”>….</some_tag>
```

- Make use of entity references for special characters

&lt;	<	Less than symbol
&gt;	>	Greater than symbol
&amp;	&	Ampersand
&apos;	‘	Apostrophe
&quot;	“	Quotation mark

XML Syntax Rules (5 of 5)

- Example using entity references:

```
<?xml version="1.0" ?>
<quote>
   <author>Sam Ewing </author>
   <paragraph> &quot; Success has a simple formula: do your
       best, and people may like it.  &quot; </paragraph>
    </quote>
```

- The value of <paragraph> when accessed will be:

  “Success has a simple formula: do your best, and people may like

Refer to the quote.xml sample code.

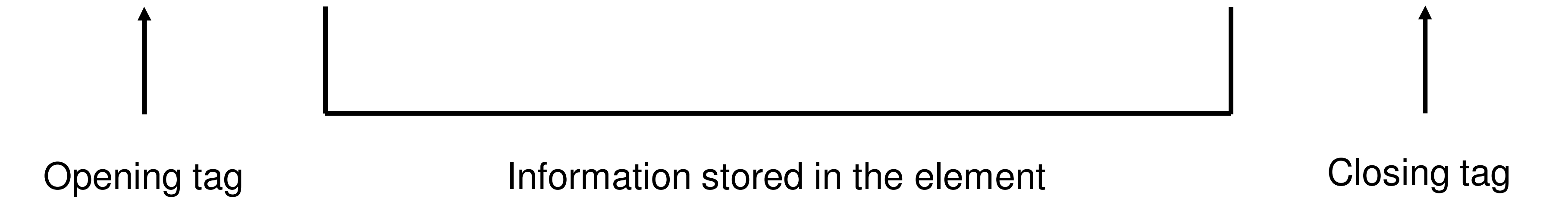
XML Elements

- Elements are used to classify data in an XML document to make the data understandable.
- Elements can have any name desired and are usually descriptive of the data they hold.
- Elements can contain other elements, usually to include more details.

XML Elements (Cont.)

- Elements can contain attributes that also allow additional information.
- Elements are defined by its opening and closing tags.

```
<myElement> This sentence is found inside the element </myElement>
- myElement-
```



XML Attributes

- XML Attributes provide additional information to the element they belong to.
- Attributes are information that are often not part of the data but are used in manipulating the data the element holds.
- Attributes are commonly used for identification purposes, in such cases, there are more than one element of the same type.

XML Attributes (Cont.)

- XML Attribute Sample:

```
<?xml version="1.0" ?>
  <class_list  year=“3”  section=“B”>
      <student  id=“B-0001”>
             <first_name>Anna</first_name>
             <last_name>Sanders</last_name>
             <gender>Female</gender>
      </student>
      <student  id=“B-0002”>
             <first_name>John</first_name>
             <last_name>dela Cruz</last_name>
             <gender>Male</gender>
      </student>
  </class_list>
```

i Refer to the classlist.xml sample code.

Validating XML Documents (1 of 17)

- Validating XML documents is done through a DTD (Document Type Definition) or an XSD (XML Schema Definition).
- There are different types of XML documents in terms of validity:

Broken XML Documents	Refer to XML Documents where syntax rules are violated.
Well-Formed XML Documents	Refer to XML documents that fully comply to the syntax rules.
Valid XML Documents	Refer to XML documents that are well-formed and comply to a DTD/XSD.

Validating XML Documents (2 of 17)

- A DTD defines the structure of an XML document with a list of legal elements and attributes.
- XML Schema Definition (XSD) is the XML-based alternative of DTDs, having the same purpose of DTDs but more powerful and extensible.
- DTD is the older standard. It is most probable that XSD will replace DTD for validating XML Documents.

i Refer to the person_DTD.xml sample code.

Validating XML Documents (3 of 17)

- XHML Schema (XSD):
  - Is extensible to accept future additions.
  - Is more powerful than its predecessor.
  - Is written similarly to XML as XSD makes use of XML syntax, hence most of the rules of XML apply to XSD.
  - Has data types and namespaces, unlike DTDs.
  - Includes the following data types:
    - String
    - Date
    - Numeric
    - Many others

Refer to the person_XSD.xml and person.xsd sample codes.

Validating XML Documents (4 of 17)

- The <schema> element is the root element for XSD.

Syntax :

```
<xs:schema>
…
…
…
</xs:schema>
```

- The <schema> element can have attributes, including the default namespace to be used.

Validating XML Documents (5 of 17)

- Attributes of elements are defined within the elements where the attributes belong.

Syntax :

```
<xs:attribute name=“name_of_attribute" type=“data_type"/>
```

Example :

```
<xs:element name=“dog”>
     <xs:complexType>
          <xs:simpleContent>
               <xs:extension base=“xs:string”>
                    <xs:attribute name=“breed" type=“xs:string"/>
                </xs:extension>
           </xs:simpleContent>
       </xs:complexType>
</xs:element>
```

i Refer to the Samples.xml and XSD_Samples.xsd sample codes.

Validating XML Documents (6 of 17)

- Elements in schemas define the structure and properties of elements in XML documents.
- Elements in schemas are divided into two types:
  - Simple Elements
  - Complex Elements

Validating XML Documents (7 of 17)

- Simple elements refer to elements containing “only text” and cannot contain other elements or attributes.

In XML :

```
<first_name>Michael Angelo</first_name>
```

Syntax :

```
<xs:element name=“my_name" type=“the_datatype"/>
```

Example :

```
<xs:element name=“first_name” type=“xs:string”/>
```

i Refer to the Samples.xml and XSD_Samples.xsd sample codes.

Validating XML Documents (8 of 17)

- Simple elements can have default or fixed values.

Syntax:

```
<xs:element name=“my_name" type=“the_datatype“ default=“default_value”/>
<xs:element name=“my_name" type=“the_datatype“ fixed=“fixed_value”/>
```

Examples:

```
<xs:element name=“current_year” type=“xs:integer” default=“2008”/>
<xs:element name=“legal_age” type=“xs:integer” fixed=“18”/>
```

Validating XML Documents (9 of 17)

- Complex elements refer to elements that can have attributes, such as:
  - Elements that contain only text (with attributes).
  - Elements that are empty.
  - Elements containing other elements.
  - Elements that contain both text and other elements.

Validating XML Documents (10 of 17)

- Elements that contain “only text” but have attributes are considered complex elements.

XML:

```
<dog dog_tag_number=“100012”>Spike</dog>
```

XSD:

```
<xs:element name=“dog">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
         <xs:attribute name=“dog_tag_number" type="xs:integer" />
      </xs:extension>
     </xs:simpleContent>
   </xs:complexType>
</xs:element>
```

i Refer to the Samples.xml and XSD_Samples.xsd sample codes.

Validating XML Documents (11 of 17)

- Elements that are empty refer to elements that hold no text but may contain attributes.

XML:

```
<computer serial_ID=“103A-0212-00A7-101B” />
```

XSD:

```
<xs:element name=“computer">
    <xs:complexType>
        <xs:attribute name=“serial_ID" type="xs:string"/>
    </xs:complexType>
</xs:element>
```

i Refer to the Samples.xml and XSD_Samples.xsd sample codes.

Validating XML Documents (12 of 17)

- Elements that serve only to contain other elements are considered complex elements.

XML:

```
<car>
  <color>red</color>
  <wheels>4</wheels>
</car>
```

XSD :

```
<xs:element name=“car">
   <xs:complexType>
         <xs:sequence>
             <xs:element name=“color" type="xs:string"/>
             <xs:element name=“wheels" type="xs:integer"/>
         </xs:sequence>
   </xs:complexType>
</xs:element>
```

i Refer to the Samples.xml and XSD_Samples.xsd sample codes.

Validating XML Documents (13 of 17)

- Elements that contain both text and other elements are considered complex elements

XML:

```
<letter> Dear Mrs.
  <name>Erika Daniels</name>. Your child,
  <child_name>Michael</child_name>, has done something at
            school. Please come to the prinicipal’s office
            anytime tomorrow,
  <schedule>2008-08-21</schedule>.
</letter>
```

XSD :

```
<xs:element name="letter">
     <xs:complexType mixed="true">
            <xs:sequence>
                <xs:element name="name" type="xs:string"/>
                <xs:element name=“child_name" type="xs:string"/>
                <xs:element name=“schedule" type="xs:date"/>
            </xs:sequence>
   </xs:complexType>
</xs:element>
```

i Refer to the Samples.xml and XSD_Samples.xsd sample codes.

Validating XML Documents (14 of 17)

- Indicators control how elements can be used:
  - Order Indicators:
    - All
    - Choice
    - Sequence
  - Occurrence Indicators:
    - maxOccurs
    - minOccurs
  - Group Indicators:
    - Group name
    - attributeGroup name

Validating XML Documents (15 of 17)

- The <any> element allows other elements not specified in the schema, which makes it extensible.
- Similarly, the <anyAttribute> element allows other attributes not specified in the schema within the designated element.

Validating XML Documents (16 of 17)

- Element names can be substituted by defining a ‘substitutionGroup’ in the XML schema.
- Substitution can be useful when developers and users speak different languages, where it may be feasible to change element names for easier understanding.

Validating XML Documents (17 of 17)

- Restrictions set the acceptable values for elements and attributes in XML documents.
- Some restrictions on values that can be applied are:
  - Set of values
  - Series of values
  - Whitespace character
  - Length Restrictions