Enterprise Systems and Architecture

CMPU4025

XML

XML

- Stands for E**X**tensible **M**arkup **L**anguage
- Similar to HTML
- XML was designed to **carry data**, not to display data like HTML
- XML tags are **not predefined**. You must define your own tags

XML Example

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

XML – Naming Rules

- Names can contain letters, numbers, and other characters
- Names cannot start with a number or punctuation character
- Names cannot start with the letters xml (or XML, or Xml, etc.)
- Names cannot contain spaces

XML Documents

- XML documents consist of three parts
  - The prolog
  - The document body
  - The epilog
- The prolog is optional and provides information about the document itself

Structure of an XML doc

- The prolog consists of four parts in the following order:
  - XML declaration
  - Miscellaneous statements or comments
  - Document type declaration / Schema
  - Miscellaneous statements or comments
- This order has to be followed or the parser will generate an error message.
- *None* of these four parts are required, but it is good form to include them.

XML Declaration

- The XML declaration is always the first line of code in an XML document. It tells the processor what follows is written using XML. It can also provide any information about how the parser should interpret the code.
- The complete syntax is:

  <?xml version="*version number*" encoding="*encoding type*" standalone="*yes | no*" ?>
- A sample declaration might look like this:

  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>

XML Comments

- Comments or miscellaneous statements go after the declaration. Comments may appear anywhere after the declaration.
- The syntax for comments is:

  <!- - *comment text* - ->
- This is the same syntax for HTML comments

Elements and Attributes

- Elements are the basic building blocks of XML files.
- XML supports two types of **elements**:
  - Closed
    - *<element_name>Content</element_name>*
    - Example: <Artist>Miles Davis</Artist>
  - Empty (also called Open)
    - <element_name/>
    - Example: <Artist/>

Elements and Attributes

- Element names are case sensitive
- Elements can be nested, as follows:

```
<CD>Kind of Blue
   <TRACK>So What (9:22)</TRACK>
   <TRACK>Blue in Green (5:37)</TRACK>
</CD>
```

Elements and Attributes

- Nested elements are called ***child elements***.
- Elements must be nested correctly. Child elements must be enclosed within their parent elements.

Elements and Attributes

- All elements must be nested within a single **document** or **root element**. There can be only **one** root element.
- An **open** or **empty** element is an element that contains no content. They can be used to mark sections of the document for the XML parser.

Elements and Attributes

- An **attribute** is a feature or characteristic of an element. Attributes are text strings and must be placed in single or double quotes. The syntax is:

<element_name attribute="value"> … </element_name>

Example:

<note **id="12"**>This is a note…></note>

XML Example for book data…

```
<bookstore>
 <book category="CHILDREN">
      <title>Harry Potter</title>
      <author>J K. Rowling</author>
      <year>2005</year>
      <price>29.99</price>
 </book>
 <book category="WEB">
      <title>Learning XML</title>
      <author>Erik T. Ray</author>
      <year>2003</year>
      <price>39.95</price>
 </book>
</bookstore>
```

Well Formed XML

- **Well-Formed** XML is:
  - XML text that satisfies the syntactic rules as laid out in the XML specification
  - **Non well-formed** XML will cause errors in applications and parsers that expect strict XML syntax

Elements and Attributes

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!-- This document contains data on Jazz Warehouse special offers -->

<SPECIALS>
<TITLE>Monthly Specials at the Jazz Warehouse</TITLE>
   <CD>Kind of Blue
      <ARTIST>Miles Davis</ARTIST>
      <TRACK length="9:22">So What</TRACK>
      <TRACK length="9:46">Freddie Freeloader</TRACK>
      <TRACK length="5:37">Blue in Green</TRACK>
      <TRACK length="11:33">All Blues</TRACK>
      <TRACK length="9:26">Flamenco Sketches</TRACK>
   </CD>
   <CD>Cookin'
      <ARTIST>Miles Davis</ARTIST>
      <TRACK length="5:57">My Funny Valentine</TRACK>
      <TRACK length="9:53">Blues by Five</TRACK>
      <TRACK length="4:22">Airegin</TRACK>
      <TRACK length="13:03">Tune-Up</TRACK>
   </CD>
   <CD>Blue Train
      <ARTIST>John Coltrane</ARTIST>
      <TRACK length="10:39">Blue Train</TRACK>
      <TRACK length="9:06">Moment's Notice</TRACK>
      <TRACK length="7:11">Locomotion</TRACK>
      <TRACK length="7:55">I'm Old Fashioned</TRACK>
      <TRACK length="7:03">Lazy Bird</TRACK>
   </CD>
</SPECIALS>
```

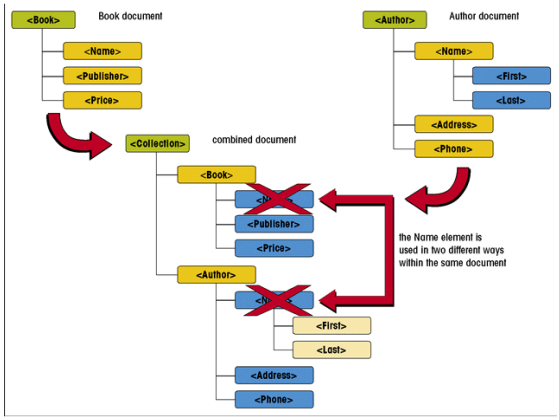
Namespaces

Working with Namespaces

- **Name collision** occurs when elements from two or more documents share the same name – i.e. if we combined the two XML documents (see next slide).
- Name collision is not a problem if you are not concerned with **validation**. The document content only needs to be **well-formed**.
- However, name collision will keep a document from being validated.

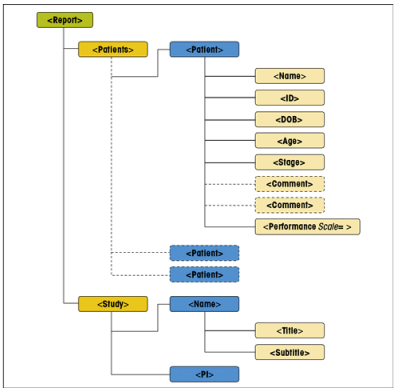
Name Collision

This figure shows two documents each with a Name element



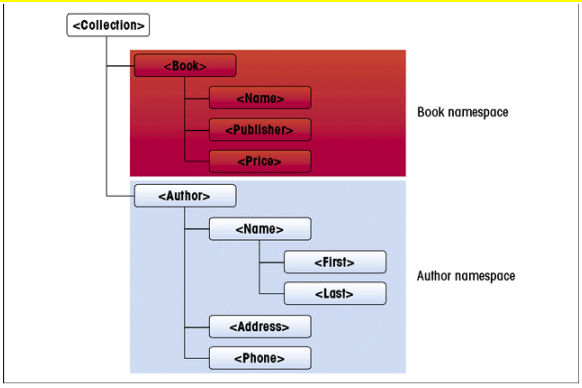
Example: Layout of UHOSP.XML Document

This figure shows the structure of the UHOSP.xml file and the collision between the Name element.



Using Namespaces to Avoid Name Collision

This figure shows how to use a namespace to avoid collision



Declaring a Namespace

- A **namespace** is a defined collection of element and attribute names.
- Names that belong to the same namespace must be unique. Elements can share the same name if they reside in different namespaces.
- Namespaces must be declared before they can be used.

Declaring a Namespace

- A namespace can be declared in the prolog or as an element attribute. The syntax to declare a namespace in the prolog is:

  <?xml:namespace ns="*URI*" prefix="*prefix*"?>

- Where *URI* is a Uniform Resource Identifier that assigns a unique name to the namespace, and *prefix* is a string of letters that associates each element or attribute in the document with the declared namespace.

Declaring a Namespace

- For example,

  <?xml:namespace ns=http://uhosp/patients/ns prefix="pat">

- Declares a namespace with the prefix "pat" and the URI http://uhosp/patients/ns.

- The URI is not a Web address. A URI identifies a physical or an abstract resource.

URI and URL

- A physical resource is a resource one can access and work with such as a file, a Web page, or an e-mail address. A URL is one type of URI.
- An abstract resource is one that doesn't have any physical existence, the URI is used as an identifier or an ID.
- The URI http://uhosp/patients/ns is simply a text identifier.

Applying a Namespace to an Element

- Once it has been declared and its URI specified, the namespace is applied to elements and attributes by inserting the namespace prefix before each element name that belongs to the namespace.

```
<prefix:element>
   content
</prefix:element>
```

- Here, *prefix* is the namespace prefix and element is the **local part** of the element name.

Applying a Namespace to an Element

- Prefixed names are called **qualified names** and an element name without a namespace prefix is called an **unqualified name**.
- Qualified names can be added to a document by adding the xmlns attribute to an element.

Declaring a Namespace as an Element Attribute

- The syntax is:

  xmlns:*prefix*="*URI*"

- Where *prefix* and *URI* are the prefix and URI for the namespace.

Declaring a Namespace as an Element Attribute

- For example, the code:

```
<pat:Patients  xmlns:pat="http://uhosp/patients/ns">
  <Patient>
    <Name>Cynthia Dibbs</Name>
    <ID>MR890-041-02</ID>
    <DOB>1945-05-22</DOB>
    <Age>58</Age>
    <Stage>II</Stage>
    <Performance Scale="Karnofsky">0.81</Performance>
  </Patient>
</pat:Patients>
```

Declaring a Namespace as an Element Attribute

- …applies the namespace http://uhosp/patients/ns namespace to the Patient element and all of its child elements.
- While the "pat" prefix was only added to the Patients element name, the XML parser considers the other elements parts of the Patients namespace and they inherit the namespace.

Declaring a Namespace as an Element Attribute

- They are unqualified elements, though, because they lack a namespace prefix.
- Declaring a namespace by adding it as an attribute of the document's root element places all elements in the namespace.
- All elements are children of the root element.

Declaring a Default Namespace

- You can specify a **default namespace** by omitting the prefix in the namespace declaration.
- The element containing the namespace attribute and all of its child elements are assumed to be part of the default namespace.

Using Namespaces with Attributes

- Attributes, like elements, can become qualified by adding the namespace prefix to the attribute name. For example,

  prefix:*attribute*=“*value*”

Using Namespaces with Attributes

- No element may contain two attributes with the same name.
- No element may contain two qualified attribute names with the same local part, pointing to identical namespaces, even if the prefixes are different.