4 : Fundamentals of XML

IT2406 - Web Application Development 1

Level I - Semester 2

XML Overview

What is XML?

- Standard “markup” language for information
  - SGML with 80% functionality but 20% complexity
  - Designed by W3C member companies
- Extensible
  - Can be used for both documents and messages
  - Unlike HTML, new “tags” can be defined
- International
  - Based on Unicode character set

HTML But Better…

- HTML
  - Defines “visual” document layout
    - Paragraphs, images, etc…
  - Browsers allow liberal use (and abuse)
- XML
  - Defines semantic structure for data
    - Music collection, financial transaction, etc…
  - Strict definition for document syntax

The Basic Rules

- XML is case sensitive
- All start tags must have end tags
- Elements must be properly nested
- XML declaration is the first statement
- Every document must contain a root element
- Attribute values must have quotation marks
- Certain characters are reserved for parsing

Common Errors for Element Naming

- Do not use white space when creating names for elements
- Element names cannot begin with a digit, although names can contain digits
- Only certain punctuation allowed – periods, colons, and hyphens

Walking through an Example

- Modify the computer.xml document
  - Add a new element named “software” with an attribute named “language”
  - The attribute’s value should be the name of a programming language
  - Create another XML element called “IFStatment”
  - Use the IFStatment element to tag the following data: if (a < b && b >= 0)
  - Close the “software” tag
- After you have added these new items into the XML document, parse it again to ensure that it is still well formed. Use the feedback to correct any errors.

Part 2: Legal Building Blocks of XML

- A Document Type Definition (**DTD**) allows the developer to create a set of rules to specify legal content and place restrictions on an XML file

- If the XML document does not follow the rules contained within the DTD, a parser generates an error

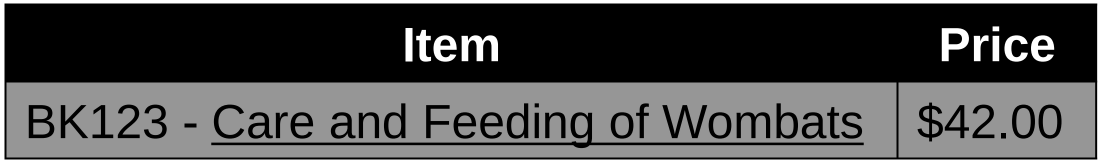
- An XML document that conforms to the rules within a DTD is said to be **valid**

Why Use a DTD?

- A single DTD ensures a common format for each XML document that references it
- An application can use a standard DTD to verify that data that it receives from the outside world is valid
- A description of legal, valid data further contributes to the interoperability and efficiency of using XML

An Example in HTML

```
<table border='1'>
  <tr style='background:black;color:white'>
    <th>Item
    <th>Price
  </tr>
  <tr valign='top' style='background:silver'>
    <td>BK123 - <u>Care and Feeding of Wombats</u>
    <td>$42.00
  </tr>
</table>
```



Item	Price
BK123 - Care and Feeding of Wombats	$42.00

The Same Thing in XML

```
<order>
  <item code='BK123'>
    <name>Care and Feeding of Wombats</name>
    <price currency='USD'>42.00</price>
  </item>
</order>
```

```
- <order>
-  <item code="BK123">
     <name>Care and Feeding of Wombats</name>
     <price currency="USD">42.00</price>
   </item>
  </order>
```

The Business Connection

- Protocol independence
  - Eases intra-business communication
  - Allows information interchange with partners
- Platform independence
  - Bridges legacy systems to new applications
- Open standard
  - Freedom from data control (e.g. EDI)
  - Everyone “speaks” the same language

XML Syntax: Documents

Basic Document Structure

- Element tags
  - Elements have associated attributes
- Text content
- Miscellaneous
  - Encoding, document type declarations
  - Entity references
  - Comments, processing instructions, etc…

Example XML Document (1 of 6)

- XML declaration

```
<?xml version=‘1.0’ encoding=‘Shift_JIS’?>
<!DOCTYPE order SYSTEM ‘grammar.dtd’>
<?xml-stylesheet type=‘text/xsl’ href=‘style.xsl’?>
<order>
  <item code=‘BK123’>
    <name>Care and Feeding of Wombats</name>
    <price currency=‘USD’>42.00</price>
  </item>
</order>
```

Example XML Document (2 of 6)

- Document type declaration

```
<?xml version=‘1.0’ encoding=‘Shift_JIS’?>
<!DOCTYPE order SYSTEM ‘grammar.dtd’>
<?xml-stylesheet type=‘text/xsl’ href=‘style.xsl’?>
<order>
  <item code=‘BK123’>
    <name>Care and Feeding of Wombats</name>
    <price currency=‘USD’>42.00</price>
  </item>
</order>
```

Example XML Document (3 of 6)

- Processing instructions

```
<?xml version=‘1.0’ encoding=‘Shift_JIS’?>
<!DOCTYPE order SYSTEM ‘grammar.dtd’>
<?xml-stylesheet type=‘text/xsl’ href=‘style.xsl’?>
<order>
  <item code=‘BK123’>
    <name>Care and Feeding of Wombats</name>
    <price currency=‘USD’>42.00</price>
  </item>
</order>
```

Example XML Document (4 of 6)

- Element tags

```
<?xml version='1.0' encoding='Shift_JIS'?>
<!DOCTYPE order SYSTEM 'grammar.dtd'>
<?xml-stylesheet type='text/xsl' href='style.xsl'?>
<order>
  <item code='BK123'>
    <name>Care and Feeding of Wombats</name>
    <price currency='USD'>42.00</price>
  </item>
</order>
```

Example XML Document (5 of 6)

- Attributes of element tags

```
<?xml version='1.0' encoding='Shift_JIS'?>
<!DOCTYPE order SYSTEM 'grammar.dtd'>
<?xml-stylesheet type='text/xsl' href='style.xsl'?>
<order>
  <item code='BK123'>
    <name>Care and Feeding of Wombats</name>
    <price currency='USD'>42.00</price>
  </item>
</order>
```

Example XML Document (6 of 6)

- Text content

```
<?xml version='1.0' encoding='Shift_JIS'?>
<!DOCTYPE order SYSTEM 'grammar.dtd'>
<?xml-stylesheet type='text/xsl' href='style.xsl'?>
<order>
  <item code='BK123'>
    <name>Care and Feeding of Wombats</name>
    <price currency='USD'>42.00</price>
  </item>
</order>
```

Differences with HTML

- Elements must be balanced, properly nested

• e.g. <br />	OK
• e.g. <b>bold <i> and italic </i> text</b>	OK
• e.g. <b>bold <i> and italic </b> text</i>	BAD!
• e.g. <ul> <li> list item </ul>	BAD!

- Attributes must be specified, quoted

• e.g. <img src='images/banner.gif'/>	OK
• e.g. <img src=images/banner.gif />	BAD!
• e.g. <ul compact> <li> list item </li> </ul>	BAD!

Other Important Points

- Documents *must* be well-formed
  - Document contains single root element
  - Elements are balanced and properly nested
  - Attributes are specified and quoted
  - Text content contains legal XML characters
- Documents *may* be valid
  - Document structure and content follows rules specified by grammar (e.g. DTD, XML Schema)

XML Syntax: DTDs

Validation of XML Documents

- XML documents *must* be well-formed
- XML documents *may* be valid
  - Validation verifies that the structure and content of the document follows rules specified by grammar
- Types of grammars
  - Document Type Definition (DTD)
  - XML Schema (XSD)
  - Relax NG (RNG)

What is a DTD?

- Document Type Definition
  - Defined in the XML 1.0 specification
  - Allows user to create new document grammars
    - A subset borrowed from SGML
    - Uses non-XML syntax!
  - Document-centric
    - Focus on document structure
    - Lack of “normal” datatypes (e.g. int, float)

Document Structure

- Element declaration
    - Element name
    - Content model
- Attribute list declaration
    - Element name
    - Attribute name
    - Value type
    - Default value

Element Declaration

- Content models
  - ANY
  - EMPTY
  - Children
    - Nestable groups of sequences and/or choices
    - Occurrences for individual elements and groups
  - Mixed content
    - Intermixed elements and parsed character data

Children Content Model

- Sequences
  - Order required e.g. (foo,bar,baz)
- Choices
  - Any one from list e.g. (foo|bar|baz)
- Nested sequences and choices
  - e.g. (foo,bar,(baz|mumble))
  - e.g. (foo|(bar,baz))

Children Occurrences

- Specify occurrence count for…
  - Individual elements
  - Groups of sequences and choices
- Occurrences

Exactly one	e.g.	foo	(foo,bar)
Zero or one	e.g.	foo?	(foo,bar)?
Zero or more	e.g.	foo*	(foo|bar)*
One or more	e.g.	foo+	(foo|bar)+

Attribute List Declaration

- Value types
  - CDATA
  - ENTITY, ENTITIES
  - ID, IDREF, IDREFS
  - NMTOKEN, NMTOKENS
  - NOTATION
  - Enumeration of valuese.g. (true|false)
- Default value
  - #IMPLIED, #REQUIRED, #FIXED
  - Default value if not specified in document

Example DTD (1 of 6)

- Text declaration

```
<?xml version=‘1.0’ encoding=‘ISO-8859-1’?>
<!ELEMENT order (item)+>
<!ELEMENT item (name,price)>
<!ATTLIST item code NMTOKEN #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ATTLIST price currency NMTOKEN ‘USD’>
```

Example DTD (2 of 6)

- Element declarations

```
<?xml version=‘1.0’ encoding=‘ISO-8859-1’?>
<!ELEMENT order (item)+>
<!ELEMENT item (name,price)>
<!ATTLIST item code NMTOKEN #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ATTLIST price currency NMTOKEN ‘USD’>
```

Example DTD (3 of 6)

- Element content models

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!ELEMENT order (item)+>
<!ELEMENT item (name,price)>
<!ATTLIST item code NMTOKEN #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ATTLIST price currency NMTOKEN 'USD'>
```

Example DTD (4 of 6)

- Attribute list declarations

```
<?xml version=‘1.0’ encoding=‘ISO-8859-1’?>
<!ELEMENT order (item)+>
<!ELEMENT item (name,price)>
<!ATTLIST item code NMTOKEN #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ATTLIST price currency NMTOKEN ‘USD’>
```

Example DTD (5 of 6)

- Attribute value type

```
<?xml version=‘1.0’ encoding=‘ISO-8859-1’?>
<!ELEMENT order (item)+>
<!ELEMENT item (name,price)>
<!ATTLIST item code NMTOKEN #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ATTLIST price currency NMTOKEN ‘USD’>
```

Example DTD (6 of 6)

- Attribute default value

```
<?xml version=‘1.0’ encoding=‘ISO-8859-1’?>
<!ELEMENT order (item)+>
<!ELEMENT item (name,price)>
<!ATTLIST item code NMTOKEN #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ATTLIST price currency NMTOKEN ‘USD’>
```

Macro Substitution Using Entities

- What are entities?
    - Document pieces, or “storage units”
    - Simplify writing of documents and DTD grammars
    - Modularize documents and DTD grammars
- Types
    - General entities for use in document
        - Example of use: &entity;
    - Parameter entities for use in DTD
        - Example of use: %entity;

General Entities

- Declaration
  - <!ENTITY name ‘Andy Clark’>
  - <!ENTITY content SYSTEM ‘pet-peeves.ent’>
- Reference in document
  - <name>&name;</name>
  - <pet-peeves>&content;</pet-peeves>

Parameter Entities

- Declaration
  - <!ENTITY % boolean ‘(true|false)’>
  - <!ENTITY % html SYSTEM ‘html.dtd’>
- Reference in DTD
  - <!ATTLIST person cool %boolean; #IMPLIED>
  - %html;

Specifying DTD in Document

- Doctype declaration
  - *Must* appear before the root element
  - *May* contain declarations *internal* to document
  - *May* reference declarations *external* to document
- Internal subset
  - Commonly used to declare general entities
  - Overrides declarations in external subset

Doctype Example (1 of 4)

- Only internal subset

```
<?xml version='1.0' encoding='UTF-16'?>
<!DOCTYPE root [
  <!ELEMENT root (stem)>
  <!ELEMENT stem EMPTY>
]>
<root>
  <stem/>
</root>
```

Doctype Example (2 of 4)

- Only external subset
  - Using system identifier
  - Using public identifier

```
<?xml version=‘1.0’ encoding=‘UTF-16’?>
<!DOCTYPE root SYSTEM ‘tree.dtd’>
<root> <stem/> </root>
```

```
<?xml version=‘1.0’ encoding=‘UTF-16’?>
<!DOCTYPE root PUBLIC ‘-//Tree 1.0//EN’  ‘tree.dtd’>
<root> <stem/> </root>
```

Doctype Example (3 of 4)

- Internal *and* external subset

```
<?xml version=‘1.0’ encoding=‘UTF-16’?>
<!DOCTYPE root SYSTEM ‘tree.dtd’ [
  <!ELEMENT root (stem)>
  <!ELEMENT stem EMPTY>
]>
<root>
  <stem/>
</root>
```

Doctype Example (4 of 4)

- Syntactically legal but never used

```
<?xml version=‘1.0’ encoding=‘UTF-16’?>
<!DOCTYPE root >
<root>
  <stem/>
</root>
```

Beyond DTDs…

- DTD limitations
  - Simple document structures
  - Lack of “real” datatypes
- Advanced schema languages
  - XML Schema
  - Relax NG
  - …

XML Namespaces

The Problem

- Documents use different vocabularies
  - Example 1: CD music collection
  - Example 2: online order transaction
- Merging multiple documents together
  - Name collisions can occur
    - Example 1: albums have a <name>
    - Example 2: customers have a <name>
  - How do you differentiate between the two?

The Solution: Namespaces!

- What is a namespace?
  - A syntactic way to differentiate similar names in an XML document
- Binding namespaces
  - Uses Uniform Resource Identifier (URI)
    - e.g. “http://example.com/NS”
  - Can bind to a named or “default” prefix

Namespace Binding Syntax

- Use “xmlns” attribute
  - Named prefix
    - e.g. <a:foo xmlns:a=‘http://example.com/NS’/>
  - Default prefix
    - e.g. <foo xmlns=‘http://example.com/NS’/>
- Element and attribute names are “qualified”
  - URI, local part (or “local name”) pair
    - e.g. { “http://example.com/NS” , “foo” }

Example Document (1 of 3)

- Namespace binding

```
<?xml version=‘1.0’ encoding=‘UTF-8’?>
<order>
  <item code=‘BK123’>
    <name>Care and Feeding of Wombats</name>
    <desc xmlns:html=‘http://www.w3.org/1999/xhtml’>
       The <html:b>best</html:b> book ever written!
    </desc>
  </item>
</order>
```

Example Document (2 of 3)

- Namespace scope

```
<?xml version='1.0' encoding='UTF-8'?>
<order>
  <item code='BK123'>
    <name>Care and Feeding of Wombats</name>
    <desc xmlns:html='http://www.w3.org/1999/xhtml'>
       The <html:b>best</html:b> book ever written!
    </desc>
  </item>
</order>
```

Example Document (3 of 3)

- Bound elements

```
<?xml version='1.0' encoding='UTF-8'?>
<order>
  <item code='BK123'>
    <name>Care and Feeding of Wombats</name>
    <desc xmlns:html='http://www.w3.org/1999/xhtml'>
      The <html:b>best</html:b> book ever written!
    </desc>
  </item>
</order>
```

Important Points

- Namespace “scope” is the element and descendents from point of binding
- Attributes are **not** in element’s namespace
  - Unless implicitly prefixed
- Can **not** unbind named prefixes
  - However, you *can* unbind default prefix

Using Namespaces with DTDs

- The problem
  - DTD syntax does not support namespaces
- The solution
  - Use a namespace-aware schema language
  - Use parameter entity “trick” to add simple namespace support to existing DTDs

Parameter Entity Trick: Step 1

- Define parameter entities
  - Prefix, suffix, and xmlns parameter entity

```
<!ENTITY % prefix ''>
<!ENTITY % suffix ''>
```

  - Xmlns parameter entity

```
<!ENTITY % xmlns ‘xmlns%suffix;’>
```

Parameter Entity Trick: Step 2

- Define element name parameter entities
  - One for every element name in grammar

```
<!ENTITY % order ‘%prefix;order’ >
<!ENTITY % item ‘%prefix;item’>
<!ENTITY % name ‘%prefix;name’>
<!ENTITY % price ‘%prefix;price’>
```

Parameter Entity Trick: Step 3

- Modify all element declarations to reference element names by parameter entity

```
<!ELEMENT %order; (%item;)+>
<!ELEMENT %item; (%name;,%price;)>
<!ELEMENT %name; (#PCDATA)>
<!ELEMENT %price; (#PCDATA)>
```

Parameter Entity Trick: Step 4

- Declare namespace binding attribute for all possible root elements

```
<!ATTLIST %order; %xmlns; CDATA ‘http://example.com/NS’>
```

Add Namespace Information to Existing, Un-prefixed Documents

- Existing documents gain namespace info

```
<?xml version=‘1.0’ encoding=‘EBCDIC’?>
<!DOCTYPE order SYSTEM ‘grammar.dtd’>
<order>
  <item code=‘BK123’>
    <name>Care and Feeding of Wombats</name>
    <price currency=‘USD’>42.00</price>
  </item>
</order>
```

Use New Prefix with Same DTD

- Redefine prefix, suffix in DTD internal subset

```
<?xml version='1.0' encoding='EBCDIC'?>
<!DOCTYPE a:order SYSTEM 'grammar.dtd' [
  <!ENTITY % prefix 'a:'>
  <!ENTITY % suffix ':a'>
]>
<a:order xmlns:a='http://example.com/NS'>
  <a:item code='BK123'>
  <!-- ... -->
```

Useful Links

- XML 1.0 Specification
  - http://www.w3.org/TR/REC-xml
- Annotated XML 1.0 Specification
  - http://www.xml.com/axml/testaxml.htm
- Informational web sites
  - http://www.xml.com/
  - http://www.xmlhack.com/
- Namespaces in XML Specification
  - http://www.w3.org/TR/REC-xml-names