The eXtensible Markup Language (XML)

Outline

- Part 1: The basics of creating an XML document
- Part 2: Developing constraints for a well formed XML document
- Part 3: XML and supplementary technologies

XML Simplifies Things

- It simplifies data sharing
- It simplifies data transport
- It simplifies platform changes
- It simplifies data availability

Background for XML

- An Extensible Markup Language (XML) document describes the *structure of data*
- XML is a tool for data transportation and data storage in platform and language neutral way.
- XML and HTML have a similar syntax … both derived from SGML
- An XML document resides in its own file with an '.xml' extension
- XML plays an important role in the exchange of a wide variety of data on the web
- XML defines set of rules for encoding documents which is both human-readable and machine-readable
- All rules are defined in XML 1.0 specification developed by W3C an open standard

History of XML

• Standardized Generalized Markup Language (SGML) - allows information about the document's structure to be preserved

• SGML is used to specify mark up languages.

• The purpose of SGML is to create the vocabularies which could be used to mark up documents with structural tags.

• HTML - one of the most popular applications of SGML

• HTML - mark up language used for presentation i.e design a webpage

• HTML - All tags predefined

• Limitation of HTML – Data storage and interchange of data is not possible using HTML • XML bridges this gap – human readable, while being flexible enough to support platform and – architecture independent data interchange

Difference Between HTML and XML

- HTML tags have a fixed meaning and browsers know what it is.
- XML tags are different for different applications, and users know what they mean.
- HTML tags are used for display.
- XML tags are used to describe documents and data.

Example of an HTML Document

```
<html>
  <head><title>Example</title></head.
<body>
  <h1>This is an example of a page.</h1>
  <h2>Some information goes here.</h2>
</body>
</html>
```

Example of an XML Document

```
<?xml version="1.0"/>
<address>
  <name>Alice Lee</name>
  <email>alee@aol.com</email>
  <phone>212-346-1234</phone>
  <birthday>1985-03-22</birthday>
</address>
```

The Basic Rules

- XML is case sensitive
- All start tags must have end tags
- Tags must be properly nested.
  - **<name><email>…</name></email> is not allowed.**
  - **<name><email>…</email><name> is.**
- XML declaration is optional and if present then should be the first statement
- Every document must contain a root element
- Attribute values must have quotation marks." " or ' '

Eg. <mail name="Abc ">

- Certain characters are reserved for parsing

- Tags are case sensitive.
  - **<address> is not the same as <Address>**
- Tags that do not have end-tags must be terminated by a '/'.
  - <br /> is an html example.

<element></element> or <element />

- Empty elements can have attributes.

  Eg. <mail id="id1" />

- XML document comprises of
  - XML Declaration
  - Root element
  - Child elements

XML Declaration

• Declarations gives the information to the browser about the xml file.

• XML declaration prepares XML processor to work with the XML document.

• It is an optional line in a XML file but when used, it must be the first line of the XML file.

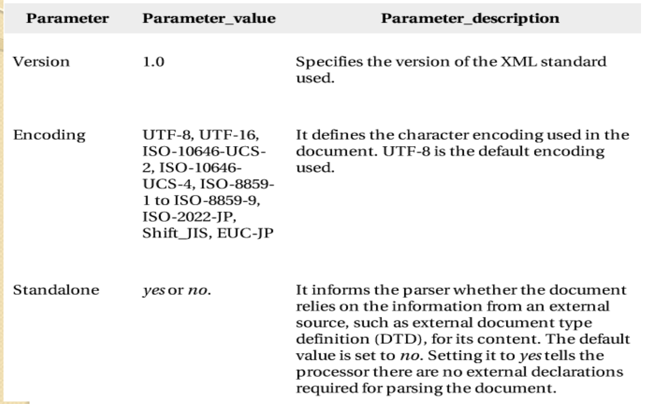
• The basic XML declaration statement only needs the language and the version.

• Example: <?xml version="1.0" encoding="UTF-8" ?>

<? indicates that the markup is a SGML declaration rather than a HTML tag

syntax shows XML declaration

```
<?xml   version="version_number"
encoding="encoding_declaration"
 standalone="standalone_status" ?>
```



Parameter	Parameter_value	Parameter_description
Version	1.0	Specifies the version of the XML standard used.
Encoding	UTF-8, UTF-16, ISO-10646-UCS-2, ISO-10646-UCS-4, ISO-8859-1 to ISO-8859-9, ISO-2022-JP, Shift_JIS, EUC-JP	It defines the character encoding used in the document. UTF-8 is the default encoding used.
Standalone	*yes* or *no*.	It informs the parser whether the document relies on the information from an external source, such as external document type definition (DTD), for its content. The default value is set to *no*. Setting it to *yes* tells the processor there are no external declarations required for parsing the document.

Rules

- An XML declaration should abide with the following rules:
- If the XML declaration is present in the XML, it must be placed as the first line in the XML document.
- If the XML declaration is included, it must contain version number attribute.
- The Parameter names and values are case-sensitive.
- The names are always in lower case.
- The order of placing the parameters is important. The correct order is: version, encoding and standalone.
- Either single or double quotes may be used.
- The XML declaration has no closing tag i.e. </?xml>

XML Declaration Examples

XML declaration with no parameters:

<?xml >

XML declaration with version definition:

<?xml version="1.0">

XML declaration with all parameters defined:

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

XML declaration with all parameters defined in single quotes:

<?xml version='1.0' encoding='iso-8859-1' standalone='no' ?>

XML - ENCODING

- Encoding is the process of converting unicode characters into their equivalent binary representation. When the XML processor reads an XML document, it encodes the document depending on the type of encoding. Hence, we need to specify the type of encoding in the XML declaration.
- Encoding Types: There are mainly two types of encoding:

  UTF-8 and UTF-16

UTF stands for UCS Transformation Format, and UCS itself means Universal Character Set. The number 8 or 16 refers to the number of bits used to represent a character. They are either 8(one byte) or 16(two bytes). For the documents without encoding information, UTF-8 is set by default.

Following example shows declaration of encoding:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<contact-info>
 <name>Tanmay Patil</name>
<company>TutorialsPoint</company>
<phone>(011) 123-4567</phone>
 </contact-info>
```

In the above example encoding="UTF-8", specifies that 8-bits are used to represent the characters. To represent 16-bit characters, UTF-16 encoding can be used.

The XML files encoded with UTF-8 tend to be smaller in size than those encoded with UTF-16 format.

Comments

• Notes that are declared within the XML which will describe the XML elements and attributes.

• It is not interpreted by the XML processor.

<!-- XML comments -->

Two dashes in the middle of a comment are not allowed. Eg. <!-- This is a -- comment -->

Strange, but allowed:

<!-- This is a - - comment -->

XML Comments Rules

- Following rules are needed to be followed for XML comments:
- Comments cannot appear before XML declaration.
- Comments may appear anywhere in a document.
- Comments must not appear within attribute values.
- Comments cannot be nested inside the other comments.

XML - WHITE SPACES

- Whitespace is a collection of spaces,tabs, and newlines.
- They are generally used to make a document more readable.
- White-space is Preserved in XML .
- XML does not truncate multiple white-spaces (HTML truncates multiple white-spaces to one single white-space):

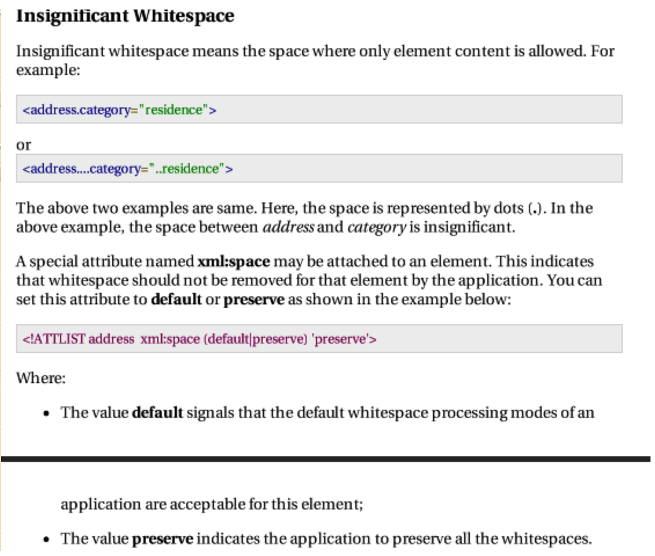
XML: Hello   Tove

XML document contain two types of white spaces (a) Significant whitespace. (b) Insignificant whitespace

Significant Whitespace

- significant whitespace occurs within the element which contain text and markup present together. for example:
- <Fname>TanmayPatil</Fname>

  and
- <Fname>Tanmay Patil</Fname>
- The above two elements are different because of the space between Tanmay and Patil. Any program reading this element in an XML file is obliged to maintain the distinction.



Insignificant Whitespace

Insignificant whitespace means the space where only element content is allowed. For example:

```
<address.category="residence">
```

or

```
<address....category="..residence">
```

The above two examples are same. Here, the space is represented by dots (.). In the above example, the space between *address* and *category* is insignificant.

A special attribute named **xml:space** may be attached to an element. This indicates that whitespace should not be removed for that element by the application. You can set this attribute to **default** or **preserve** as shown in the example below:

```
<!ATTLIST address  xml:space (default|preserve) 'preserve'>
```

Where:

- The value **default** signals that the default whitespace processing modes of an application are acceptable for this element;
- The value **preserve** indicates the application to preserve all the whitespaces.

Main Components of an XML Document

- Elements: <hello>
- Attributes: <item id="33905">
- Entities: &lt; (<)

XML Elements

- XML elements can be defined as building blocks of an XML.
- Elements can behave as containers to hold text, elements, attributes, media objects or all of these.
- Syntax: <element-name attribute1 attribute2> ....content </element-name>

Naming Rule

• XML element's name contains:
– alphanumeric characters
– special characters
• underscore (_)
• hyphen(-)
• period(.).
• Starts with letter or underscore.
• Names are case sensitive.
Eg. <name>, <address1>,<address_1>,<_dob>
If you choose a naming style, it is good to be consistent!

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

Root and child Element

• Root element is the top most element which will contain the other elements in it.
• It is the first element of the xml document.

```
<person>
      <name> Tom </name>
      <gender> Male </gender>
</person>
```

Name and gender are the child elements

Attributes

• The attributes provides additional information about the element
– <Employee id=123>

INCORRECT:
<note date=12/11/2007></note>
Correct:
<note date="12/11/2007"></note>

XML References

- References usually allow you to add or include additional text or markup in an XML document. References always begin with the symbol "&" ,which is a reserved character and end with the symbol ";".

XML has two types of references:

- Entity References: An entity reference contains a name between the start and the end delimiters. For example &amp; where amp is name. The name refers to a predefined string of text and/or markup.
- Character References: These contain references, such as &#65;, contains a hash mark ("#") followed by a number. The number always refers to the Unicode code of a character. In this case, 65 refers to alphabet "A".

Entity References

<message>salary < 1000</message>

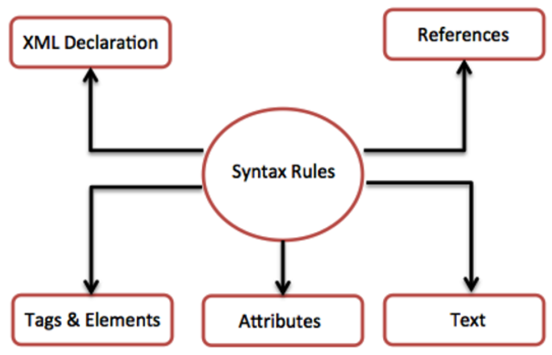
To avoid this error, replace the "<" character with an **entity reference**:

<message>salary &lt; 1000</message>

There are 5 pre-defined entity references in XML:

&lt;	< less than
&gt;	> greater than
&amp;	& ampersand
&apos;	' apostrophe
&quot;	" quotation mark

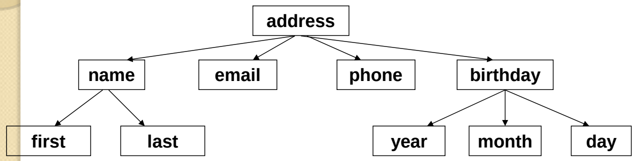
Only < and & are strictly illegal in XML, but it is a good habit to replace > with &gt; as well.



Well-Formed Documents

- An XML document is said to be well-formed if it follows all the rules.
- An XML parser is used to check that all the rules have been obeyed.
- Recent browsers such as Internet Explorer 5 and Netscape 7 come with XML parsers.
- Parsers are also available for free download over the Internet. One is Xerces, from the Apache open-source project.
- Java 1.4 also supports an open-source parser.
- A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a Document Type Definition(DTD) or XML schema

XML Files are Trees



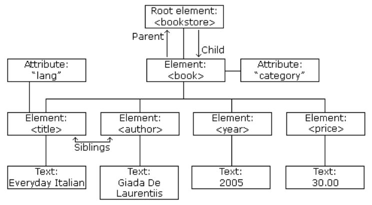
Expanded Example

```
<?xml version = "1.0" ?>
<address>
   <name>
       <first>Alice</first>
       <last>Lee</last>
    </name>
    <email>alee@aol.com</email>
    <phone>123-45-6789</phone>
    <birthday>
       <year>1983</year>
       <month>07</month>
       <day>15</day>
   </birthday>
</address>
```

XML Trees

- An XML document has a single root node.
- The tree is a general ordered tree.
  - A parent node may have any number of children.
  - Child nodes are ordered, and may have siblings.
- Preorder traversals are usually used for getting information out of the tree.

XML Tree



```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

XML Elements are Extensible

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

produce this output:

```
MESSAGE
To: Tove
From: Jani

Don't forget me this weekend!
```

Author of the XML document added some extra information to it:

-
  ```
  <note>
    <date>2008-01-10</date>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>
  ```
- application will still be able to find the <to>, <from>, and <body> elements in the XML document and produce the same output.
- Xml can be extended without breaking applications.

XML Attributes

Attribute values must always be quoted. Either single or double quotes

`<gangster name="George &quot;Shotgun&quot; Ziegler">`

XML Elements vs. Attributes

```
<person gender="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

Can be written as:

```
<person>
  <gender>female</gender>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

- There are no rules about when to use attributes or when to use elements in XML.

Avoid XML Attributes?

- Some things to consider when using attributes are:
- attributes cannot contain multiple values (elements can)
- attributes cannot contain tree structures (elements can)
- attributes are not easily expandable (for future changes)
- Don't end up like this: <note day="10" month="01" year="2008" to="Tove" from="Jani" heading="Reminder" body="Don't forget me this weekend!"> </note>

Applications of XML

• A few example of current platforms and applications making of XML heavily are

– Cell Phones

– File Converters

– Voice XML

– Web Publishing

– Web Searching and automating Web Task

– Meta data applications

Benefits of XML

• XML format is not a computer language. It is highly human readable. It is not difficult to code compared to HTML(XML tags are highly user defined).

• XML is highly compatible with Java and is hundred percent portable.

• Any application regardless of the platform or architecture can us the XML data

• XML is highly extendable - can create our own tags or can use t tags created by others.

• Complex data searches can be easily performed in XML documents.

• XML data can be made to display on different types of devices.

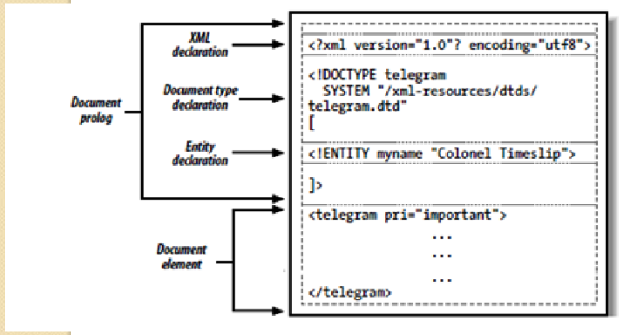
Drawbacks of XML

- Lack of adequate processing applications to process the XML.
- XML is not specific to any platform, and it has a neutral platform requirement
- All the XML standards are not yet fully standardized.
- Users have reported problems with the parser and there are problems with XML and HTTP which are still being resolved.
- XML flexibility could be some times its biggest disadvantage.

Different tags in XML

Object	Purpose
Empty element	An element without any containing data
Container Element	Group elements with character data
Declaration	Add new parameter, entity to the processing environment
Processing Instruction	Feed special instruction about XML
Comment	Add additional information that will be ignored by XML processer
CDATA	Character data that will not parsed and which preserves special characters in it
Entity Reference	Command the parser to insert text stored somewhere.

Structure of XML



```
<?xml version="1.0"? encoding="utf8">
<!DOCTYPE telegram
  SYSTEM "/xml-resources/dtds/
telegram.dtd"
[
<!ENTITY myname "Colonel Timeslip">
]>
<telegram pri="important">
                    ...
                    ...
                    ...
</telegram>
```

Example: XML News

```
<?xml version="1.0" encoding="UTF-8"?>
<nitf>
  <head>
    <title>Colombia Earthquake</title>
  </head>
  <body>
    <headline>
      <hl1>143 Dead in Colombia Earthquake</hl1>
    </headline>
    <byline>
      <bytag>By Jared Kotler, Associated Press Writer</bytag>
    </byline>
    <dateline>
      <location>Bogota, Colombia</location>
      <date>Monday January 25 1999 7:28 ET</date>
    </dateline>
  </body>
</nitf>
```