UNIT IV

XML Introduction

What Is XML?

- eXtensible Markup Language
- A simplified version of SGML
- Maintains the most useful parts of SGML
- Designed so that SGML can be delivered over the Web
- More flexible and adaptable than HTML
- XHTML -- a reformulation of HTML 4 in XML 1.0

Difference between XML and HTML

XML was designed to carry data, not displaying data

- XML is not a replacement for HTML.
- Different goals:

  XML was designed to describe data and to focus on what data is.
  HTML was designed to display data and to focus on how data looks.
- HTML is about displaying information, XML is about describing information.

HTML and XML,

XML stands for eXtensible Markup Language

HTML is used to mark up text so it can be displayed to users

XML is used to mark up data so it can be processed by computers

HTML describes both structure (e.g. <p>, <h2>, <em>) and appearance (e.g. <br>, <font>, <i>)

XML describes only content, or “meaning”

HTML uses a fixed, unchangeable set of tags

In XML, you make up your own tags

HTML and XML,

- HTML and XML look similar, because they are both SGML languages (SGML = Standard Generalized Markup Language)
  - Both HTML and XML use elements enclosed in tags (e.g. <body>This is an element</body>)
  - Both use tag attributes (e.g., <font face="Verdana" size="+1" color="red">)
  - Both use entities (&lt;, &gt;, &amp;, &quot;, &apos;)
- More precisely,
  - HTML is defined in SGML
  - XML is a (very small) subset of SGML

HTML and XML,

- HTML is for humans
  - HTML describes web pages
  - You don’t want to see error messages about the web pages you visit
  - Browsers ignore and/or correct as many HTML errors as they can, so HTML is often sloppy
- **XML is for computers**
  - XML describes data
  - The rules are strict and errors are not allowed
    - In this way, XML is like a programming language
  - Current versions of most browsers can display XML
    - However, browser support of XML is spotty at best

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>
```

An example of XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Why Is XML Important?

- Plain Text
  - Easy to edit
  - Useful for storing small amounts of data
  - Possible to efficiently store large amounts of XML data through an XML front end to a database
- Data Identification
  - Tell you what kind of data you have
  - Can be used in different ways by different applications

Why Is XML Important?

- Stylability
  - Inherently style-free
  - XSL---Extensible Stylesheet Language
  - Different XSL formats can then be used to display the same data in different ways
- Inline Reusabiliy
  - Can be composed from separate entities
  - Modularize your documents without resorting to links

Why is XML important?

- Linkability -- XLink and XPointer
  - Simple unidirectional hyperlinks
  - Two-way links
  - Multiple-target links
  - “Expanding” links
- Easily Processed
  - Regular and consistent notation
  - Vendor-neutral standard

Why is XML important?

- Hierarchical
  - Faster to access
  - Easier to rearrange

Example XML document

```xml
<?xml version="1.0"?>
<weatherReport>
  <date>7/14/97</date>
  <city>North Place</city>, <state>NX</state>
  <country>USA</country>
  High Temp: <high scale="F">103</high>
  Low Temp: <low scale="F">70</low>
  Morning: <morning>Partly cloudy, Hazy</morning>
  Afternoon: <afternoon>Sunny &amp; hot</afternoon>
  Evening: <evening>Clear and Cooler</evening>
</weatherReport>
```

From: **XML: A Primer**, by Simon St. Laurent

Overall structure

- An XML document may start with one or more processing instructions (PIs) or directives:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="ss.css"?>
```

- Following the directives, there must be exactly *one* root element containing all the rest of the XML:

```
<weatherReport>
   ...
</weatherReport>
```

XML building blocks

- Aside from the directives, an XML document is built from:
  - elements: high in <high scale="F">103</high>
  - tags, in pairs: <high scale="F">103</high>
  - attributes: <high scale="F">103</high>
  - entities: <afternoon>Sunny &amp; hot</afternoon>
  - character data, which may be:
    - parsed (processed as XML)--this is the default
    - unparsed (all characters stand for themselves)

XML Elements

- XML Elements are Extensible

  XML documents can be extended to carry more information

- XML Elements have Relationships

  Elements are related as parents and children

- Elements have Content

  Elements can have different content types: **element** content, **mixed** content, **simple** content, or **empty** content and **attributes**

- XML elements must follow the naming rules

Elements and attributes

- Attributes and elements are somewhat interchangeable
- Example using just elements:

```
<name>
   <first>David</first>
   <last>Matuszek</last>
</name>
```

- Example using attributes:

```
<name first="David" last="Matuszek"></name>
```

- You will find that elements are easier to use in your programs--this is a good reason to prefer them
- Attributes often contain metadata, such as unique IDs
- Generally speaking, browsers display only elements (values enclosed by tags), not tags and attributes

XML Attributes

- Located in the start tag of elements
- Provide additional information about elements
- Often provide information that is not a part of data
- Must be enclosed in quotes
- Should I use an element or an attribute?

  metadata (data about data) should be stored as attributes, and that data itself should be stored as elements

Well-formed XML

- Every element must have *both* a start tag and an end tag, e.g. <name> … </name>
  - But empty elements can be abbreviated: <break />.
  - XML tags are case sensitive
  - XML tags may not begin with the letters xml, in any combination of cases
- Elements must be properly nested, e.g. *not* <b><i>bold and italic</b></i>
- Every XML document must have one and only one root element
- The values of attributes must be enclosed in single or double quotes, e.g. <time unit="days">
- Character data cannot contain < or &

Entities

- Five special characters must be written as entities:

&amp;	for	&amp;	(almost always necessary)
&lt;	for	&lt;	(almost always necessary)
&gt;	for	&gt;	(not usually necessary)
&quot;	for	"	(necessary inside double quotes)
&apos;	for	'	(necessary inside single quotes)

- These entities can be used even in places where they are not absolutely required
- These are the *only* predefined entities in XML

XML declaration

The XML declaration looks like this:

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

- The XML declaration is not required by browsers, but *is* required by most XML processors (so include it!)
- If present, the XML declaration must be first--*not even whitespace* should precede it
- Note that the brackets are <? and ?>
- version="1.0" is required (this is the *only* version so far)
- encoding can be "UTF-8" (ASCII) or "UTF-16" (Unicode), or something else, or it can be omitted
- standalone tells whether there is a separate DTD

Processing instructions

- PIs (Processing Instructions) may occur anywhere in the XML document (but usually first)
- A PI is a command to the program processing the XML document to handle it in a certain way
- XML documents are typically processed by more than one program
- Programs that do not recognize a given PI should just ignore it
- General format of a PI: `<?`*target instructions*`?>`
- Example: `<?xml-stylesheet type="text/css" href="mySheet.css"?>`

Comments

- <!-- This is a comment in both HTML and XML -->
- Comments can be put anywhere in an XML document
- Comments are useful for:
  - Explaining the structure of an XML document
  - Commenting out parts of the XML during development and testing
- Comments are not elements and do not have an end tag
- The blanks after <!-- and before --> are optional
- The character sequence -- cannot occur in the comment
- The closing bracket *must* be -->
- Comments are not displayed by browsers, but can be seen by anyone who looks at the source code

Names in XML

- Names (as used for tags and attributes) must begin with a letter or underscore, and can consist of:
  - Letters, both Roman (English) and foreign
  - Digits, both Roman and foreign
    - . (dot)
    - - (hyphen)
    - _ (underscore)
    - : (colon) should be used only for namespaces
  - Combining characters and extenders (not used in English)

Namespaces

- Recall that DTDs are used to define the tags that can be used in an XML document
- An XML document may reference more than one DTD
- Namespaces are a way to specify which DTD defines a given tag
- XML, like Java, uses qualified names
  - This helps to avoid collisions between names
  - Java: myObject.myVariable
  - XML: myDTD:myTag
  - Note that XML uses a colon (:) rather than a dot (.)

Namespaces and URIs

- A namespace is defined as a unique string
  - To guarantee uniqueness, typically a URI (Uniform Resource Indicator) is used, because the author “owns” the domain
  - It doesn't have to be a “real” URI; it just has to be a unique string
  - Example: http://www.matuszek.org/ns
- There are two ways to use namespaces:
  - Declare a default namespace
  - Associate a prefix with a namespace, then use the prefix in the XML to refer to the namespace

Namespace syntax

- In *any* start tag you can use the reserved attribute name xmlns:

  `<book xmlns="http://www.matuszek.org/ns">`

  - This namespace will be used as the default for all elements up to the corresponding end tag
  - You can override it with a specific prefix

- You can use almost this same form to declare a prefix:

  `<book xmlns:dave="http://www.matuszek.org/ns">`

  - Use this prefix on *every tag and attribute* you want to use from this namespace, including end tags--it is *not* a default prefix

  `<dave:chapter dave:number="1">To Begin</dave:chapter>`

- You can use the prefix in the start tag in which it is defined:

  `<dave:book xmlns:dave="http://www.matuszek.org/ns">`

Review of XML rules

- Start with <?xml version="1"?>
- XML is case sensitive
- You must have exactly one root element that encloses all the rest of the XML
- Every element must have a closing tag
- Elements must be properly nested
- Attribute values must be enclosed in double or single quotation marks
- There are only five predeclared entities

Another well-structured example

```
<novel>
   <foreword>
      <paragraph> This is the great American novel.
      </paragraph>
</foreword>
   <chapter number="1">
      <paragraph>It was a dark and stormy night.
     </paragraph>
      <paragraph>Suddenly, a shot rang out!
      </paragraph>
   </chapter>
</novel>
```

XML Building blocks--Prolog

- The part of an XML document that precedes the XML data

- Includes

  A declaration: version [, encoding, standalone]

  An optional DTD (Document Type Definition)

- Example

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
```

XML Syntax

- All XML elements must have a closing tag
- XML tags are case sensitive
- All XML elements must be properly nested
- All XML documents must have a root tag
- Attribute values must always be quoted
- With XML, white space is preserved
- With XML, a new line is always stored as LF
- Comments in XML: `<!-- This is a comment -->`

Displaying XML

- XML documents do not carry information about how to display the data
- We can add display information to XML with
  - CSS (Cascading Style Sheets)
  - XSL (eXtensible Stylesheet Language) --- preferred

XML Application1—Separate data

XML can Separate Data from HTML

- Store data in separate XML files
- Using HTML for layout and display
- Using Data Islands
- Data Islands can be bound to HTML elements

Benefits:

Changes in the underlying data will not require any changes to your HTML

XML Application2—Exchange data

XML is used to Exchange Data

- Text format
- Software-independent, hardware-independent
- Exchange data between incompatible systems, given that they agree on the same tag definition.
- Can be read by many different types of applications

Benefits:

- Reduce the complexity of interpreting data
- Easier to expand and upgrade a system

XML Application3—Store Data

XML can be used to Store Data

- Plain text file
- Store data in files or databases
- Application can be written to store and retrieve information from the store
- Other clients and applications can access your XML files as data sources

Benefits:

Accessible to more applications

XML Application4—Create new language

XML can be used to Create new Languages

- WML (Wireless Markup Language) used to markup Internet applications for handheld devices like mobile phones (WAP)
- MusicXML used to publishing musical scores

Advantages of XML

- XML is text (Unicode) based.
  - Takes up less space.
  - Can be transmitted efficiently.
- One XML document can be displayed differently in different media.
  - Html, video, CD, DVD,
  - You only have to change the XML document in order to change all the rest.
- XML documents can be modularized. Parts can be reused.

Conclusion

- XML is a self-descriptive language
- XML is a powerful language to describe structure data for web application
- XML is currently applied in many fields
- Many vendors already supports or will support XML