Overview

- XML and SGML
- A document
  - Content
  - Tagged Content
  - Valid Tagged Content
- Defining a DTD
  - Elements
  - Attributes
  - Entities
- Simple Examples
- Design Methods
XML and SGML

- SGML introduced the notion of a “Document Type Definition” or DTD
- XML is a simplified and at the same time extended form of SGML
- SGML (and XML) documents include tags or copymarks
- Documents containing tags that are correctly nested and written are called “well formed”
- Documents whose elements are as specified in a DTD are called “valid”

Content, Tags, and Elements

- Content
  Understand the pressures that are moving e-business forward
  Appreciate the impact of bit businesses versus atom business, national versus global markets
- Tags <xyz> = starttag  </xyz> = endtag
  <LIST><ITEM></ITEM><ITEM></ITEM></LIST>
- Elements (tags + content)
  <LIST><ITEM>Understand the pressures that are moving e-business forward as a new mode of doing business</ITEM><ITEM>Appreciate the impact of bit businesses versus atom business, national versus global markets, and customer driven manufacturing on the conduct of business</ITEM></LIST>
More Formally

• an element in XML is a starttag, content, and endtag
• a starttag is element name between angle brackets with one or more attributes
• an endtag is an element name between <> and >
• an empty element may be specified as an element name between < and />

A Document

Course on E-Business

by: Michael B. Spring, University of Pittsburgh

General Description

The Internet is providing new ways of communicating and of doing business. There are many facets to the developments and many technologies appearing and disappearing in the rush to develop this new area.

The objectives of this course will be to:
Understand the pressures that are moving e-business forward as a new mode of doing business
Appreciate the impact of bit businesses versus atom business, national versus global markets, and customer driven manufacturing on the conduct of business
A Document with Tags

<TITLE>Course on E-Business</TITLE>
.AUTHOR>Michael B. Spring</AUTHOR>
.INST>University of Pittsburgh</INST>
.CHAP.<CTITLE>General Description</CTITLE>
.PARA>The Internet is providing new ways of communicating and of doing business. There are many facets to the developments and many technologies appearing and disappearing in the rush to develop this new area.</PARA>
.PARA>The objectives of this course will be to:
.LIST.<ITEM>Understand the pressures that are moving e-business forward as a new mode of doing business</ITEM>
.ITEM>Appreciate the impact of bit businesses versus atom business, national versus global markets, and customer driven manufacturing on the business</ITEM>
.LIST></CHAP></MYDOC>
XML Markup

- an XML document begins with the XML declaration
  - `<?xml version="1.0"?>`
  - n.b. this is actually a processing instruction
- Following the declaration, an XML document specifies the DTD or schema which it follows
  - `<!DOCTYPE name [........]>`
  - A document that only claims to be well formed need not specify a DTD
  - Namespaces actually allow much more flexibility in nesting DTD definitions

The DTD

- A DTD can be:
  - PRIVATE – specified in the document
  - SYSTEM – specified on the system
  - PUBLIC – specified in some public registry
- This presentation deals only with private
- the name you give to a document type must be the same as the root element
  - the DTD is specified via the doctype element
  - `<!DOCTYPE name [........]>`
Content Modeling

- The DTD defines a model of the document content
- Within the [] of the <!DOCTYPE > declaration, the designer specifies the content of the document in terms of:
  - Elements
  - Attributes
  - Entities
  - Data types
  - Notations
- The most important are the element definitions

What the DTD does

- the document type definition defines the legitimate markup structure for a document.
- For each element, the DTD specifies
  - element content, if any
  - attributes of element
  - the allowable sub-elements including
    - ordering information
    - occurrence information
The DTD components

• <!ELEMENT – defines the content model for a given element
• <!ATTLIST – defines the attributes for a specified element, possible values, and defaults
• <!ENTITY – defines the entities that can be referred to in the document using entity references.
• <!NOTATION – defines, like entities means for handling non-SGML notations

Element Declaration

• groups
  • () parentheses define a group
• sequence connectors
  • , indicates in the specified order
  • | indicated a choice
  • & (SGML only) indicates elements may be in any order
• occurrence indicators
  • nothing indicates a single instance is required
  • ? Indicates optional
  • * indicates optional and repeatable
  • + indicates required and repeatable
More Detail

- Element names must consist of at least one letter
  - if more than two characters long, they may start with a `_` or a `:
  - letters, digits, hyphens, period, and underscores are allowed in the body of the name
  - spaces and tabs are not allowed

Attribute Definition

- The second type of type declaration is attribute definition, it takes the general form
  - `<!ATTLIST gi name value/range default>`
  - `<!ATTLIST memo status ("dft" | "fnl") "fnl">`
- Given this in the DTD, in text we could see a value after = and in `""`
  - `<memo status = "draft">`
- value range must either be a group, or a reserved word (see next slide)
Attribute Reserved Words

• The reserved words can be:
  • CDATA -- character data
  • NUMBER -- a number
  • NAME -- a name string
  • NMTOKENS -- names that can begin with a number
  • NMTOKENS -- names that begin with a number
  • ID -- must be a valid and unique name within the scope of the
document; ID attributes should be named consistently -- some would
say they should be called id
  • IDREF -- need not be unique but must match a value of an ID in the
document.

Default Values

• Default values may be specified as one member of the set.
• They may also include the following:
  • #REQUIRED -- must be supplied
  • #IMPLIED -- is optional and will be supplied by the system if absent
  • #CURRENT -- is the most recent value
• This allows definitions like
  • <!ATTLIST fig figtag ID #IMPLIED>
  • and
  • <!ATTLIST figref reffig IDREF #IMPLIED>
Entity Definition

- `<!ENTITY SIS "School of Information Sciences">`
  - allows &SIS; in the text.
- Character references are like entity definitions
  - an entity reference for a character might be `&amp;`;
  - a character reference might be `&#32;` or `&#x0D;`;

Special Entity References

- if an entity is a processing instruction, the keyword PI is inserted in the definition between the entity and the string literal. For example
  - `<!ENTITY dothis PI "newpage recto">`
- if an entity is to be allowed in a parameter literal, the entity must be defined using a `%` indicator. For example
  - `<!ENTITY % myref "some string">`
  - this allows us to resolve `<!ENTITY another "%myref; some other string">`
Content

- SGML character data may be defined as CDATA or RCDATA or PCDATA.
  - CDATA is simply that -- Character data
  - RCDATA is character data that may contain entity references
  - PCDATA is character data that may be fully parsed

Example

- `<!DOCTYPE letter [
  <!ELEMENT letter (adrs, sal, body, sig)>`
- `<!ELEMENT adrs (name, str, city)>`
- `<!ELEMENT name (first, last)>`
- `<!ELEMENT body (p*)>`
- `<!ELEMENT p (#PCDATA)>`
- `<!ELEMENT first (#PCDATA)>`
- `<!ELEMENT last (#PCDATA)>`
  - ...
  - ...
  - ]>`
**DTD Design**

- Building a DTD requires the traditional Systems Analysis and Design
  - Requirements Gathering
  - Analysis of Data
  - Modeling of Application
  - Implementation
  - Validation

**Requirements**

- Articulate the goals of the project
- scope of the documents included
- goal of the effort -- document validation, author productivity, multiple delivery formats
- proposed document processing architecture
- individuals to be involved in decision making
Analysis

- Analyze the requirements of the document
  - Identify and define the basic components
  - Classify the components into logical groups
  - Validate the components and the classification

Design the DTD

- Select the components that should be modeled
- Build top level element and attribute models
- Build middle level element and attribute models
- Build low level element and attribute models
- Populate the model with places where choices will be made
- Establish the linkages with the outside world
Validation

- Validate the model -- technically and semantically
- Review the relationships between the developed DTD and other existing DTDs
- Test the DTD to determine if it meets the Goals of the project
  - Does it do what we set out to do
- Implement the model providing appropriate end user training

ELM (enables lucid models)

- typical tree diagram
- boxes represent elements
  - rectangle means element required
    - + sign on vertical link indicates required and repeatable
    - * means optional and repeatable
    - ? means optional
  - ovals indicate data e.g. PCDATA
- horizontal bracket means all elements at level in order
- diagonal lines specify choice from set