XML

CIS-3010
Database Systems
Peter C. Chapin

Markup Languages

- “Plain text” documents with special commands
  - PRO
    - Plays well with version control and other program development tools.
    - “Easy” to manipulate with scripts and third party programs.
    - WYSIWYG in the sense that no control codes are hidden.
  - CON
    - Requires additional processing to create final output
    - Can be hard to learn
    - Not WYSIWYG in the sense that the document doesn't look like its final form.

Example: nroff

- Used for Unix manual pages...
  - ```
    .SH NAME
    watch \- watch for a user to log in or out of the system
    .SH SYNOPSIS
    .BR "watch " [-d " delay"] [-s|-q] [-f " logfile" ] "
    username"

    .SH DESCRIPTION
    .B watch
    is used to watch for the log in and log out activity of a
    particular user. It can record its findings to a file
    and/or display information to stdout.

    .SH OPTIONS
    .TP
    .B \-d delay
    Set the delay time in seconds. This is the time
    .B watch
    sleeps between times when it examines the system.
    ```
    ```

Example: WGML

- Used by the Open Watcom project
  - ```
    .H1. Design Details
    :P.

    .H2. Copy-On-Write?
    :P.
    This implementation of
    :CLASS.std::~string
    does not use a copy-on-write or a reference counted
    approach. Instead every string object maintains its own
    independent representation. This was done, in large
    measure, to simplify the implementation so that a
    reasonable
    :CLASS.std::~string
    could be offered quickly. However there are a number of
    difficulties with making
    :CLASS.std::~string
    reference counted and it is worth reviewing those issues
    here.
    ```
    ```

Example: LaTeX

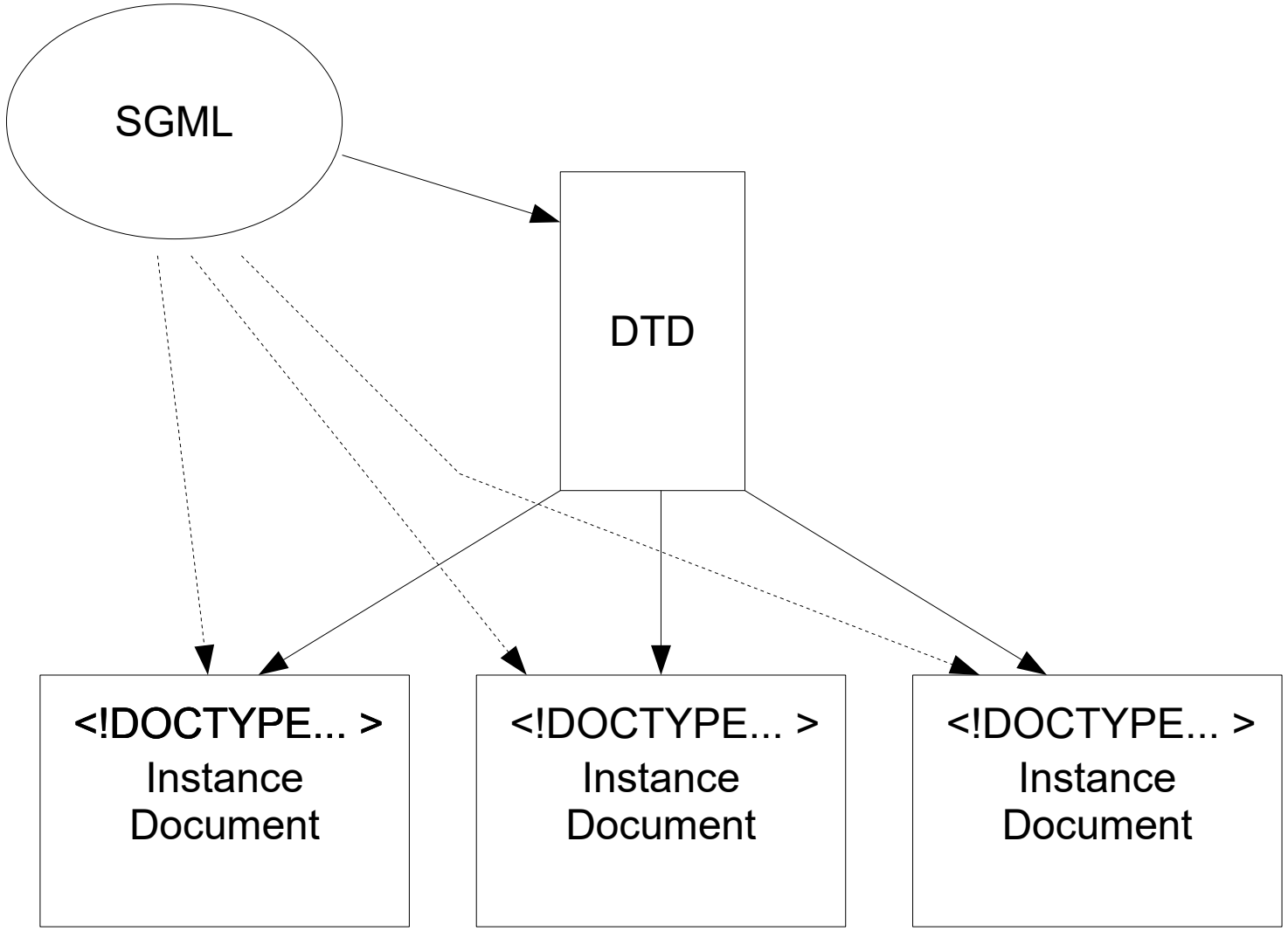
- Used extensively for technical literature.
  - ```
    \begin{document}
    \title{CIS--3152 Lab \#4\\UDP and the Domain Name System}
    \author{\copyright\ Copyright 2009 by Peter C. Chapin}
    \date{Last Revised: January 31, 2009}
    \maketitle

    \section*{Introduction}
    In this lab you will explore UDP and the domain name system
    by writing a program that does DNS queries manually.
    Although there are well known library functions for sending
    queries to a name server (\texttt{gethostbyname()} and \
    texttt{gethostbyaddr()}), you will not use those functions
    in this lab. By constructing the DNS query and by
    interpreting the response yourself you will gain more
    insight into how the domain name system works. In addition,
    since most DNS transactions use UDP, this lab will give you
    some exposure to this important, low level transport
    protocol.
    ```
    ```

SGML

- Standard Generalized Markup Language
  - A markup language to define other markup languages (a “meta markup language”).
    - ISO standard since the 1980s.
    - Powerful and expressive.
    - Complicated and difficult to implement fully.
  - How it works...
    - Use SGML to write a “document type definition” (DTD) for your markup language.
    - DTD describes the vocabulary of the markup.
    - SGML rules apply in the marked up documents too.

How It Works


SGML
DTD
<!DOCTYPE... >
Instance
Document
<!DOCTYPE... >
Instance
Document
<!DOCTYPE... >
Instance
Document

“The Look”

- All SGML documents have a similar look.
  - <body>
    Mark up is in the form of tags surrounding blocks of material.
    Start tags can be decorated with attributes. My name is &author;. “Entities” are prefixed with &.
    </body>
  - The precise names used in the tags depends on the DTD.

HTML

- If “the look” seems familiar, don’t be surprised.
  - HTML4 is an SGML application.
    - Meaning that HTML is described by an SGML document type definition.
      - Kept in a vault at the W3C!
      - http://www.w3.org/TR/REC-html40/sgml/dtd.html
  - SGML DTD's are a formal specification of what instance documents look like.
    - Documents either do or do not conform to this specification. There is no room for ambiguity.

Problem

- Many people wanted more from HTML
  - Mathematical formula
  - Music
  - Chemical formula
  - etc, etc...
- W3C didn't want to add everything to HTML
  - HTML would become huge!
- Instead we need a way to let people define their own mark up languages.

SGML?

- Why not SGML?
  - Already existed.
  - Already standard.
  - Already used by HTML
- BUT...
  - SGML processing programs are large and complex.
  - W3C wanted to allow for smaller (hand held?) systems.
  - Needed something different.

XML

- eXtensible Markup Language
  - Essentially SGML “light”
    - Same sort of DTDs
    - Same sort of “look” in the instance documents.
    - Simplified semantics.
      - Features removed
      - Other features given fewer options.
    - Easier to process.
  - XML is to SGML as Java is to C++.

XHTML v1.0

- ... a reformulation of HTML4 as an XML application.
  - Defined by an XML DTD
    - Looks very similar to the SGML DTD
  - Instance documents look very similar
    - More limitations because XML is more limited than SGML.
- Now users can define other XML markups as needed
  - Resulting markup languages are easier to handle than full SGML based languages.

Markup Languages to Know

- Many XML markup languages exist.
  - MathML
    - Allows you to describe and display mathematical formula.
  - SVG (Scalable Vector Graphics)
    - Allows you to describe vector graphics with an XML vocabulary.
  - XSL (XML Style Language)
    - Allows you to describe transformations from one XML vocabulary to another.
  - ODF (Open Document Format)
    - Office documents used by OpenOffice.org.
  - Many others... define your own!

Elements vs Tags

- `<p level=”1”>This is text</p>`
  - The whole thing is called an “element.”
  - The `<p level=”1”>` is the “start tag.”
  - The `</p>` is the “end tag.”
  - The `level=”1”` is an attribute (with its value).
- Most of the time you talk about “elements.”
  - Many people call them “tags” incorrectly.
    - It makes XML specialists cringe.

More Terminology

- Consider the following
  - ```
    <book>
      <title>XML is cool</title>
      <author>Jill
              <bold>Jones</bold></author>
    </book>
    ```
    - The `title` element has text-only content.
    - The `book` element has element-only content.
    - The `author` element has mixed content.
  - In general white space in text-only or mixed content is *significant*.
    - Each XML application decides what to do with it.
    ```

XML Restrictions

- As compared to SGML
  - All elements *must* have an end tag.
    - This is text.
      - The above is wrong. There is no tag.
      - SGML markups can define elements with optional end tags.
      - The above is okay in HTML because is optional there.
  - Element names are case sensitive.
    - This is text.</P>
      - The above is wrong. The name used in the end tag does not match that used in the start tag.
      - SGML markups can define case insensitive elements. HTML does this.

More XML Restrictions

- As compared with SGML
  - Attribute values *must* be quoted.
    - <table border=1> ... </table>
      - The above is wrong because the attribute value “1” is not quoted.
      - SGML allows unquoted attributes if they are not ambiguous.
  - Attribute values *must* be present.
    - <table border> ... </table>
      - The above is wrong because the attribute border does not have a value given.
      - SGML allows this syntax in certain cases.
- Restrictions make processing easier.

Minimization End Tags

- Always requiring an end tag is a burden.
  -
</br>
    - HTML defines a br element for break. In XHTML the end tag *must* be present (in HTML it is optional).
  -
    - XML defines the above minimization form that combines both tags in one.
    - Strictly this is an XML feature and thus not allowed in HTML (SGML) documents.
    - Web browsers are forgiving about this.
- These restrictions and features apply to *all* XML markup languages.

“Well Formed” vs “Valid”

- Well Formed...
  - An XML document is *well formed* if it obeys the syntax requirements of XML.
- Valid...
  - An XML document is *valid* if it is well formed and if it obeys the requirements of a particular *schema*.
    - A schema defines what elements are allowed, their allowed attributes, and the relationships between elements. DTDs are a kind of schema.
- Which do we want?
  - It depends on the application.

Character Sets

- XML is a modern markup language.
  - Unicode is the default character set.
  - UTF-8 encoding is the default document encoding.
- Implications...
  - XML documents might contain “arbitrary” binary data (strictly speaking they are not ASCII files).
    - If the document is in Chinese it could be loaded with non-ASCII bytes.
  - Should be handled as such.
    - application/xml is the MIME type used, not text/xml.

Entities

- Certain characters can't be represented directly.
  - A literal < character
    - Must be represented as <
      - This is called a “character entity.” Note the semicolon at the end.
  - A literal & character
    - Must be represented as &
  - Can also encode arbitrary characters this way.
    - Character with Unicode code point U+ABCD can be represented as ꯍ
      - You could also just store the Unicode character directly in the file!
- XML DTDs can define other entities.

What Makes XML Cool?

- Generic way to described structured data.
  - Many documents contain structure. Plain text obscures this structure.
  - *XML allows you to expose the structure so that programs can “see” and manipulate it.*
- Good for...
  - Managing complex documents.
  - **Storing and sharing data**.
  - Dealing with structured information in general.

Demonstration

XHTML Skeleton