

today: XML

course: Web Technology

Maarten Lamers

www.maartenlamers.com/WT2006/

Leiden University

XML: extensible markup language

according to Wikipedia (27/2/2006), XML:

- *[is a] general-purpose markup language for creating special-purpose markup languages, capable of describing many different kinds of data.*
- *... provides a text-based means to describe and apply a tree-based structure to information.*
- *[Its] purpose is to facilitate the sharing of data across different systems, particularly systems connected via the Internet.*

XML-based markup languages

they provide

- a standardized way to describe hierarchically structured data
- flexible data formats
- for sharing both data *and* format information

thus, they are useful to share information in a common way

XML as a standard

- a standard maintained by W3C (WWW Consortium)
- originates from February 1998
- open-source format (it uses text files)
- platform independent
- human-readable

XML as a hype

- during the Internet hype, it was a popular business buzz-word
- it was often related to other business buzz-words such as EDI, publishing, middleware, e-commerce, ...
- many people believed it solved all data integration problems (*panacea*); but it didn't
- it was hyped
- but is a very useful tool

previous lecture: HTML

- markup language that describes the structure and appearance of hypertext documents
- markup of text using tags

```
<P>this is a paragraph  
that contains a  
<A href="http://www.w3.org/">hyperlink</A>  
</P>
```

- HTML, by Tim Berners-Lee, was based on the Standard Generalized Markup Language (SGML)

SGML: standard generalized markup language

- generic markup language, or a meta-language:
standard for how to specify a *document* markup language
- SGML originated from the publishing world
- so, using SGML you can define a new markup language for your specific documents
- for example a “Media Technology Exam Markup Language”, a specific markup language for M.T. exam documents
- HTML is
 - a specific *instance* of SGML (!)
 - a specific markup language for hypertext documents

HTML and WWW

- WWW became more dynamic: more web application
- HTML is only the GUI for such applications
- also, companies wanted to integrate their applications outside the WWW:
 - different banks may want to share money transfers in a standardized way
 - building companies may want a standard data format for describing building materials that they want to buy/sell.
- not only in relation to the WWW
- also non-WWW-related data integration

representing data: HTML problems

- data integration means sharing data between computer systems
- for sharing data you must agree on a common representation for the type of data you want to share
- HTML is very suitable for representing hypertext documents
<P>, <H1>, <TITLE>, <A>, , ...
- HTML is not suitable for representing general data
Price: 19.95, Quantity: 5,
Color: Red, Height: 15 cm
- HTML was never meant to represent data!

XML

- this is where XML comes into the picture
- XML is a *light weight* (simpler) SGML
- *based on creating your own tags*, e.g.
<dessert>vla</dessert>
<sandwich_topping>hagelslag</sandwich_topping>
- it standardizes *markup for structured data*
 - for each type of data, specific tags can be made
 - these tags can be used to structure the data

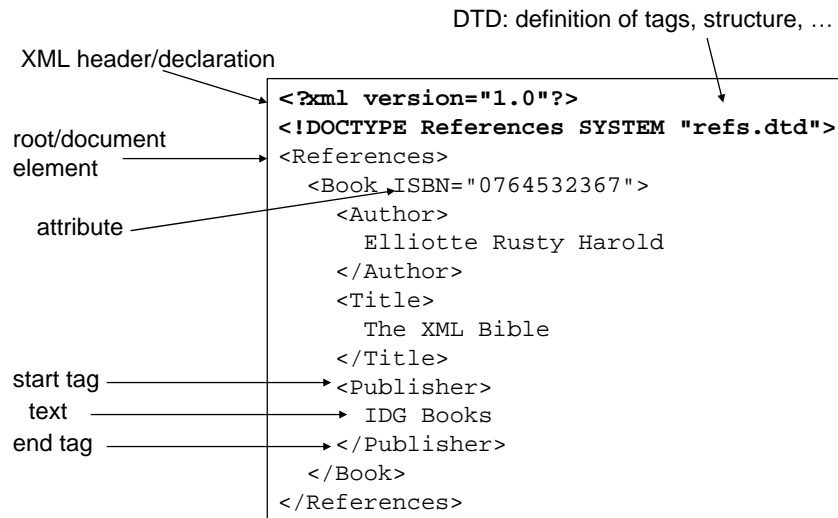
XML documents

- if you create an XML based markup language for storing data about desserts ('DML')
- then, an XML-document can describe a specific dessert by using the DML markup language
- for example:
 - a document `vla.xml` may describe vla
 - a document `icecream.xml` may describe ice cream
 - a document `jello.xml` may describe Jell-O pudding

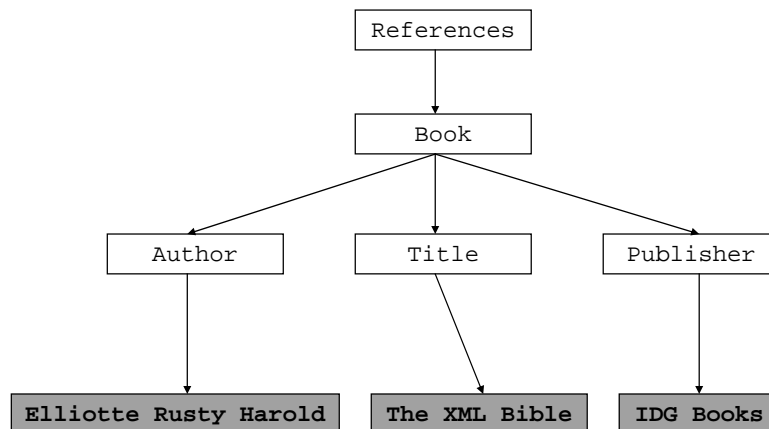
XML and meaning

- *XML doesn't care how the document is used*
- unlike HTML, which is meant for hypertext document presentation
- XML does not specify the *meaning* of data (!) only its structure
- `<superhero>Maarten</superhero>` does not specify what it means to be a superhero, it just states that item Maarten is of type superhero

an autopsy: file refs.xml



tree structure of file refs.xml



XML document anatomy

- an XML document is structured as a data tree
- each XML document describes a piece of data (!)

- an XML document that satisfies the *basic syntax* rules of XML is called *well-formed*
 1. each element must have open tag and close tag
 2. elements must be nested properly
 3. ...

Maarten says

if

your data can be expressed as a tree

and

you want to communicate it outside your application

then

consider using XML for this

DTD: document type definition

- XML documents can define their own structure
- a *Document Type Definition* (DTD) file defines a specific structure
- for example `dessert.dtd` could define a structure for describing different desserts.

- it specifies *which* tags can be places *where*
- therefore, it is comparable to a language grammar
- "Maarten teaches class today."
"Teaches Maarten today class."

- just as a sentence may or may not conform to the English grammar, an XML document may or may not conform to its DTD

an example DTD file: refs.dtd

The diagram shows a DTD file named `refs.dtd` with several annotations explaining the symbols used in the declarations:

- zero or more (*)**: Points to the asterisk in `<!ELEMENT References (Book|Article)*>`
- optional (?)**: Points to the question mark in `<!ELEMENT Book (Author+,Title,Publisher?)>`
- one or more (+)**: Points to the plus sign in `<!ELEMENT Book (Author+,Title,Publisher?)>`
- or (|)**: Points to the vertical bar in `<!ELEMENT References (Book|Article)*>`
- data (in text format)**: Points to `#PCDATA` in `<!ELEMENT Author (#PCDATA)>`
- followed by (,)**: Points to the comma in `<!ELEMENT Article (Author+,Title,Magazine)>`

```
<!ELEMENT References (Book|Article)*>
<!ELEMENT Book (Author+,Title,Publisher?)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
<!ELEMENT Article (Author+,Title,Magazine)>
<!ELEMENT Magazine (Name, Year?, Number?)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Year (#PCDATA)>
<!ELEMENT Number (#PCDATA)>
```

XML Schema

- another language for describing the structure of XML documents; an alternative to DTD
- *XML Schema* can also describe which *type* data is
- remember: using DTD, all data is treated as text

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="country">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string" />
        <xs:element name="population" type="xs:decimal" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

DTD / XML Schema

- both are a so-called "*XML schema language*"
- for describing specific XML structure, a.k.a. *schema*
- whether you use *DTD* or *XML Schema* to describe a specific XML structure, does not change your XML document that actually contains the data
- schema:
in the following slides, whenever we say "*schema*" we mean a description of a specific XML structure, regardless of whether that description is made in *DTD* or *XML Schema*.

schema

- makes any XML document *self-describing*
- because when mentioned in the XML document, it specifies *which* tags can/must be places *where*, thus letting the XML document describe its own structure
- an XML document that conforms to its schema is called *valid*
- if an XML document does not conform to its schema, it is not *valid*
- an *XML parser* is a program that can read any XML document and can *validate* it against its schema file
- *well-formed* and *valid* are different things

intermezzo: parser

- in computer technology, a *parser* is a program, that receives input and breaks it up into parts, according to some underlying structure that is agreed upon. Usually, the parts are then passed on to some other program (or program part).
- for example, an *English parser* may break up fragments of English text into the nouns, verbs, etcetera
- for example, a *C++ parser* may break up C++ program code into its components (loops, statements, functions, variables, etc)
- a parser may also check to see that all input that is required has been provided, and that the structure of the input is correct

XML and programming

- suppose you want to write a program that accesses XML documents, then your program must be able to parse XML files and validate them using a schema file (DTD or XML Schema)
- a programming interface is a pre-described way for your code to access some external data source
- several standard programming interfaces exist, most notably:
 - DOM: the Document Object Model
 - SAX: the Simple API for XML
- DOM is like the JavaScript tree
- SAX is event-based: a parser never builds an XML tree, but launches events whenever it encounters a known XML element in the document; your code can then react to these events

XML parser

- any XML parser can read any XML document, with any schema file, and validate it
- any application that supports XML should be able to read and validate any XML document
- none of such applications needs to know about each other's data format (structure) in advance...
- this is because the XML document can be self-describing by way of its association to a schema file

data integration

- the previous slide implies that the *syntax* (structure) is defined
- but not the *semantics* (meaning)
- XML does not care how you want to use the data, or what for
- for *meaningful* communication, you need a priori agreement on what the data means
- try to understand this!
- that's why XML does not solve the data integration problem, as mentioned earlier today

example: XML and meaning

- what does this XML-file describe?

```
<?xml version="1.0"?>
<!DOCTYPE References SYSTEM "pjlk.dtd">
<zwxy>
  <tc5aq>
    98
  </tc5aq>
</zwxy>
```

- in what unit is my height expressed?

```
<person>
  <height>
    78.3
  </height>
</person>
```

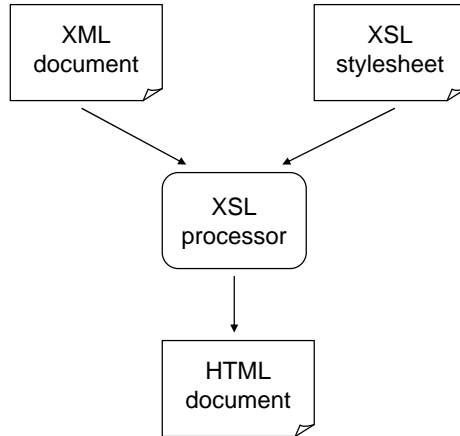
data integration

- XML raises the data-integration problem to a higher level:
 - it helps you specify how data is communicated
 - what the data means, or how it should be processed, is not specified in XML. It remains to be agreed on by all parties
- there are many standardized schema's for different application domains:
 - web-publishing, banking, chemistry, mathematics, graphics, news-providers, pharmacy, museums, ...

extensible stylesheet language (XSL)

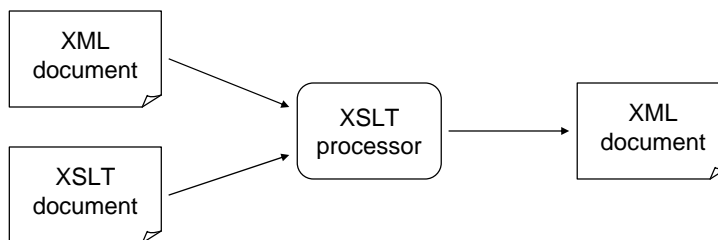
- a document written in XSL is called a *stylesheet*
- it defines XML document presentation and transformation
- XSL stylesheets can describe how XML data should be presented to a user: which XML data fields to display, where and how
- example transformations
 - XML document → HTML document
 - XML document → RTF document
 - XML document → PDF document
 - XML document → ...

XSL example

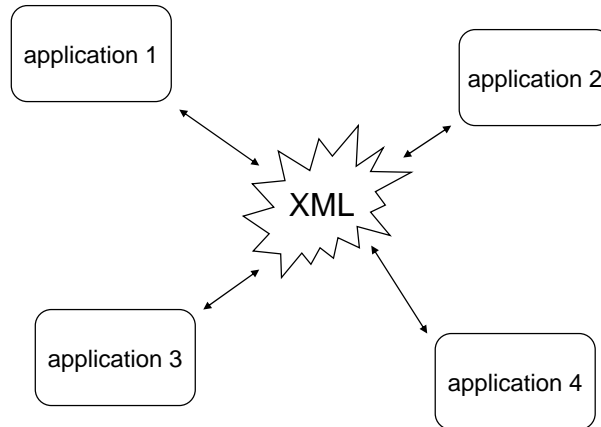


XSL transformations (XSLT)

- a sub-language of XSL
- an XSLT document describes how to change the structure of an XML document into that of another XML document with a different structure
- or: *how to transform the tree structure of XML document into tree structure of new XML document*



XML as universal data hub?



XML as universal data hub?

pro's:

- independent standard, not controlled by one company
- easy transformation between different types XML documents
- human readable
- standard interfaces available for programming

cons:

- it does not solve the data integration problem, but raises it to a higher level:
 - you must still agree on the exact form of your data, but that is made easier through XML and schema languages
 - you should still agree on the *meaning* of the data

applications

XML was successfully integrated into

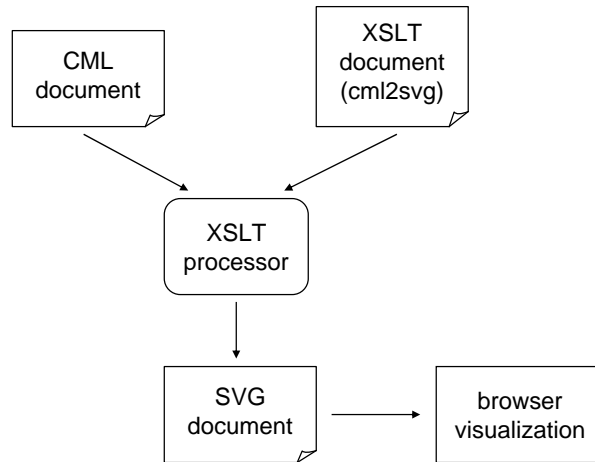
- database technology
- middleware (programs that mediate data between different applications)
- e-commerce
- publishing
- the world wide web
- chemistry, mathematics, graphics
- ...

the big question is: can we agree on common schema's !?

example XML applications: CML

- Chemical Markup Language (CML)
 - describes chemical compounds in all detail
 - widely used in chemistry field
 - <http://www.xml-cml.org/>
 - DTD: http://www.xml-cml.org/dtd/cml1_0_1.dtd
- Scalable Vector Graphics (SVG)
 - XML based language for vector-based web-graphics
 - <http://www.w3.org/Graphics/SVG/>

example XML applications: cml2svg



- <http://www.adobe.com/svg/demos/cml2svg/html/>

announcement

because of different class hours,

lab assistance of Tuesday March 14th is moved to

Thursday March 16th, 10:00 – 13:00h