

XML



Heba Owis

May 26, 2024

Agenda

Basics of the XML

Sample XML languages / applications

Why XML (1) ?

Electronic data interchange is critical in today's networked world and needs to be standardized

- Examples:
 - Banking: funds transfer
 - Education: e-learning contents
 - Scientific data
 - Chemistry: ChemML, ...
 - Genetics: BSML (Bio-Sequence Markup Language), ...

Each application area has its own set of standards for representing information

XML has become the basis for all new generation data interchange formats (markups)

Why XML (2)

Earlier electronic formats were based on plain text with line headers indicating the meaning of fields

- Does not allow for nested structures, no standard “type” language
- Tied too closely to low level document structure (lines, spaces, etc)

Each XML based standard defines what are valid elements, using XML type specification languages (i.e. grammars) to specify the syntax

- E.g. DTD (Document Type Descriptors) or XML Schema
- Plus textual descriptions of the semantics

XML allows new tags to be defined as required

A wide variety of tools is available for parsing, browsing and querying XML documents/data

Design rationale for XML (1)

1. XML must be easily usable over the Internet
2. XML must support a wide variety of applications
3. It must be easy to write programs that process XML documents
4. The number of optional features in XML must be kept small
5. XML documents should be clear and easily understood
6. The XML design should be prepared quickly
7. The design of XML must be exact and concise
8. XML documents must be easy to create
9. Keeping an XML document size small is of minimal importance

XML is a formalism to create markup languages

Markup

- text added to the data content of a document in order to convey information about data

Marked-up document contains

- data and
- information about that data (markup)

Markup language

- formalized system for providing markup

Definition of markup language specifies

- what markup is allowed
- how markup is distinguished from data
- what markup means ...

2 ways to look at the XML universe

(1) XML as formalism to define vocabularies (also called applications)

Example DTD :

```
<!ELEMENT page (title, content, comment?)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT content (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
```

Exemple of an XML document:

```
<page>
  <title>Hello XML friend</title>
  <content>
    Here is some content :)
  </content>
  <comment>
    Written by DKS/Tecfa,
  </comment>
</page>
```

(2) XML as a set of languages for defining:

Contents

Graphics

Style

Transformations and queries

Data exchange protocols

.....

Kinds of XML-based languages (1)

XML-related languages can be categorized into the following classes:

- **XML accessories, e.g. XML Schema**
 - Extends the capabilities specified in XML
 - Intended for wide, general use
- **XML transducers: e.g. XSLT**
 - Converts XML input data into output
 - Associated with a processing model
- **XML applications, e.g. XHTML**
 - Defines constraints for a class of XML data
 - Intended for a specific application area

Kinds of XML-based languages (2)

Less formally speaking: ways to use XML

Behind the scenes as a standard and easily transformed format for information

As a transfer syntax, to exchange information in a machine-parsable form

As a method of delivery direct to the user, usually in combination with a stylesheet

XML information structures (1)

Example 1: A possible book structure

Book

- FrontMatter

 - BookTitle

 - Author(s)

 - PubInfo

- Chapter(s)

 - ChapterTitle

 - Paragraph(s)

- BackMatter

 - References

 - Index

XML information structures (2)

Premise: A text is the sum of its component parts


- A **<Book>** could be defined as containing:
<FrontMatter>, **<Chapter>**s, **<BackMatter>**
- **<FrontMatter>** could contain:
<BookTitle> **<Author>**s **<PubInfo>**
- A **<Chapter>** could contain:
<ChapterTitle> **<Paragraph>**s
- A **<Paragraph>** could contain:
<Sentence>s or **<Table>**s or **<Figure>**s ...

Components chosen for book markup language should reflect anticipated use

XML information structures (3)

A corresponding XML fragment
(based on a corresponding XML application):

```
<Book>  
  <FrontMatter>  
    <BookTitle>XML Is Easy</BookTitle>  
    <Author>Tim Cole</Author>  
    <Author>Tom Habing</Author>  
    <PubInfo>CDP Press, 2002</PubInfo>  
  </FrontMatter>  
  <Chapter>  
    <ChapterTitle>First Was SGML</ChapterTitle>  
    <Paragraph>Once upon a time ...</Paragraph>  
  </Chapter>  
</Book>
```



The diagram illustrates the XML structure with two yellow callout bubbles. The first bubble, labeled 'begin element', points to the opening tag of the <BookTitle> element. The second bubble, labeled 'end element', points to the closing tag of the <BookTitle> element.

XML information structures (4)

Example 2: Movies

Elements can have attributes

```
<movies>
```

```
  <movie genre="action" star="Halle Berry">
```

```
    <name>Catwoman</name>
```

```
    <date>(2004)</date>
```

```
    <length>104 minutes</length>
```

```
  </movie>
```

```
  <movie genre="horror" star="Halle Berry">
```

```
    <name>Gothika</name>
```

```
    <date>(2003)</date>
```

```
    <length>98 minutes</length>
```

```
  </movie>
```

```
  <movie genre="drama" star="Halle Berry">
```

```
    <name>Monster's Ball</name>
```

```
    <date>(2001)</date>
```

```
    <length>111 minutes</length>
```

```
  </movie>
```

```
</movies>
```



attribute

What is an XML document ?

An XML document is a content marked up with XML (can be a file, a string, a message content or any other sort of data storage)

There are 2 levels of conforming documents:

- Well-formed: respects the XML syntax
- Valid: In addition, respects one (or more) associated grammars (schemas).

What is a well-formed XML document (1) ?

- Well-formed documents follow basic syntax rules e.g.
- there is an XML declaration in the first line
- there is a single document root
- all tags use proper delimiters
- all elements have start and end tags
 - But can be minimized if empty: `
` instead of `
</br>`
- all elements are properly nested

```
<author> <firstname>Mark</firstname>
      <lastname>Twain</lastname> </author>
```
- appropriate use of special characters
- all attribute values are quoted:

```
<subject scheme="LCSH">Music</subject>
```

What is a well-formed XML document (2) ?

Good example

```
<addressBook>
  <person>
    <name> <family>Wallace</family> <given>Bob</given> </name>
    <email>bwallace@megacorp.com</email>
    <address>Rue de Lausanne, Genève</address>
  </person>
</addressBook>
```


What is a well-formed XML document (3) ?

Bad example

```
<addressBook>
  <address>Rue de Lausanne, Genève <person></address>
  <name>
    <family>Schneider</family> <firstName>Nina</firstName>
  </name>
  <email>nina@nina.name</email>
</person>
<name><family> Muller </family> <name>
</addressBook>
```

What is a valid XML document (4) ?

Parser (i.e. that program that reads the XML) can check markup of individual document against rules expressed in a schema (DTD, XML Schema, etc.)

Typically, a schema (grammar):

- Defines available elements
- Defines attributes of elements
- Defines how elements can be embedded
- Defines mandatory and optional information

Authoring tools usually can enforce rules of DTD/Schema while document is edited

Document Type Definitions (DTDs 1)

XML document types can be specified using a DTD

DTD constraints structure of XML data

- What elements can occur
- What attributes can/must an element have
- What subelements can/must occur inside each element, and how many times.

DTD does not constrain data types

- All values represented as strings in XML

DTD definition syntax

- `<!ELEMENT element (subelements-specification) >`
- `<!ATTLIST element (attributes) >`
- ... more details later

Valid XML documents refer to a DTD (or other Schema)

Document Type Definitions (DTDs 2)

External Public DTD Declaration

```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<!DOCTYPE test PUBLIC "-//Webster//DTD test V1.0//EN"  
<test> "test" is a document element </test>
```

test = name
of the root
element

Application
should
know DTD

External DTD Declaration referring to a file or a URL

```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<!DOCTYPE test SYSTEM "test.dtd">  
<test> "test" is a document element </test>
```

DTD is
defined in
file test.dtd

Internal DTD Declaration

```
<!DOCTYPE test [  
  <!ELEMENT test EMPTY> ]>  
<test/>
```

DTD is
defined
inside XML

XML Schemas

XML Schema is a more sophisticated schema language which addresses the drawbacks of DTDs. Supports

- Typing of values
 - E.g. integer, string, etc
 - Also, constraints on min/max values
- User-defined, complex types
- Many more features, including
 - uniqueness and foreign key constraints, inheritance

XML Schema is itself specified in XML syntax, unlike DTDs

- More-standard representation, but verbose

XML Scheme is integrated with namespaces

BUT: XML Schema is significantly more complicated than DTDs.

XML Namespaces (1)

Various XML languages can be mixed

- However there can be a naming conflict, different vocabularies (DTDs) can use the same names for elements ! How to avoid confusion ?

Namespaces:

- Qualify element and attribute names with a label (prefix):

unique_prefix:element_name

- An XML namespace is a collection of names (elements and attributes of a markup vocabulary)
- identified by xmlns:prefix="URL reference"

xmlns:xlink="http://www.w3.org/1999/xlink"

XML Namespaces (2)

Example: Use of XLinks requires a namespace definition

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<STORY xmlns:xlink="http://www.w3.org/1999/xlink">
  <Title>The Webmaster</Title>
  .....
  <INFOS> <Date>30 octobre 2003 - </Date>
    <Author>DKS - </Author>
    <A xlink:href="http://jigsaw.w3.org/css-validator/check/referer"
      xlink:type="simple">CSS Validator</A> </INFOS>
</STORY>
```

The STORY Element
May contain xlink names

Title belong to
default name space

href belongs to
xlink name space

Processing instructions

XML is read by machines

Processing instructions (PI) can tell a program how to deal with contents of a given XML document

E.g. to tell a web browser to use a stylesheet with an XML content, the following PI is used:

```
<?xml-stylesheet type="style" href="sheet" ?>
```

- *Style* is the type of style sheet to access and *sheet* is the name and location of the style sheet.

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

```
<?xml-stylesheet href="stepbystep.css«  
    type="text/css"?>
```


XML + XSLT (1)

- XSLT is a transformation language that can translate from XML to anything
- Also, works well with Mozilla/Firefox and IE 6 / 7

XML Source

```
<?xml version="1.0"?>
<page>
  <title>Hello friend</title>
  <content>Here is some content :)
</content>
  <comment>Written by DKS/Tecfa,
adapted from S.M./the Cocoon
samples </ comment>
</page>
```

Translated into HTML (as an example)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<html><head><title>Hello
friend</title></head>
<body bgcolor="#ffffff">
  <h1 align="center">Hello
friend</h1>
  <p align="center"> Here is some
content :) </p>
  <hr> Written by DKS/Tecfa,
adapted from S.M./the Cocoon
samples
</body></html>
```

XML + XSLT (2)

The XSLT stylesheet used for the translation:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="page">
.....
    <html> <head> <title> <xsl:value-of select="title"/> </title> </head>
      <body bgcolor="#ffffff"> <xsl:apply-templates/> </body>
    </html>
</xsl:template>

<xsl:template match="title">
  <h1 align="center"> <xsl:apply-templates/> </h1>
</xsl:template>

<xsl:template match="content">
  <p align="center"> <xsl:apply-templates/> </p>
</xsl:template>

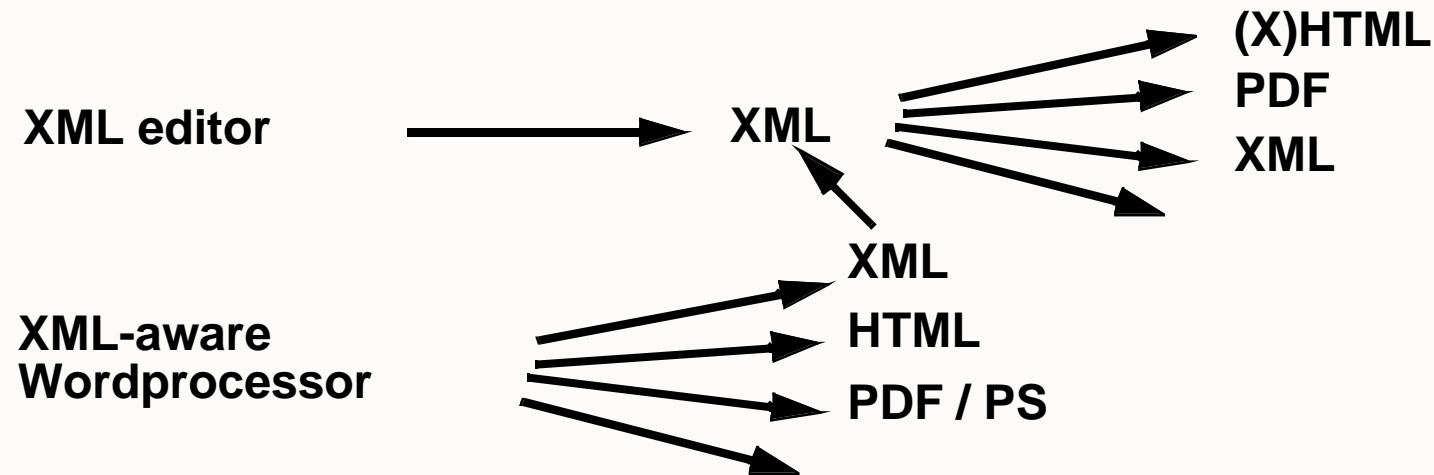
<xsl:template match="comment">
  <hr /> <xsl:apply-templates/>
</xsl:template>
</xsl:stylesheet>
```

XML in the documentation world

XML is popular in the documentation world

Specialized vocabularies to write huge documents (e.g. DocBook or DITA)

Domain-specific vocabularies to enforce semantics, e.g. legal markup, news syndication



XML Query languages

XPath (also part of XSLT)

- 13 axes (navigation directions in the tree)
 - child (/), descendant (//), following-sibling, following...
- NameTest, predicates
- E.g,
`doc("bib.xml")//book[title="Harry Potter"]/ISBN`

XQuery (superset of XPath)

- FLWOR expression

```
for $x in doc("bib.xml")//book[title
    = "Harry Potter"]/ISBN,
    $y in doc("imdb.xml")//movie
where $y//novel/ISBN = $x
return $y//title
```

Data integration and exchange languages

Web services (SOAP, WSDL, UDDI)

- Amazon.com, eBay, ...

Domain specific data exchange schemas (>1000)

- legal document exchange languages
- business information exchange ...

RSS XML news feeds

- CNN, slashdot, blogs, ...

More !!

The languages presented before are just a subset of the XML galaxy !

....

In this course we mainly will deal with:

- The XML formalism, editing XML content
- Defining DTDs
- Associating CSS stylesheets
- Transforming XML data with XSLT

Summary

XML has a wide range of applications

XML is just a formalism (meta-language), *unlike* HTML

The W3C framework includes

- General purpose (accessory, transducing, ..) languages such as XML Schema, XSLT, XPath, XQuery, Xlink, RDF, ...
- Useful languages for contents (vector graphics, multimedia animation, formulas

Other organizations

- Define domain-specific vocabularies
- Define alternative XML-based general purpose languages

XML is mostly used “behind the scene”

- e.g. for communication between web services or to store contents of Office files.
- In web pages XML «survives» well as SVG and MathML.