4. XML

INFO 202 - 10 September 2008

Bob Glushko

Why XML Matters

Publishing

Business Processes

Programming

Metadata

Money

Plan for INFO Lecture #3

Separating Content from its Container or Presentation

Document Types

Using XML to Encode Models of Document Types

Minimal XML Syntax Tutorial

The Mother of All 202 Ideas

We say "the document is about … the photograph is about… the movie is about"

We're expressing a distinction between information as conceptual or as content: and the physical container or medium, format, or technology in which the information is conveyed

It is often essential to think abstractly about "information content" without making any assumptions or statements about the "presentation" or "rendition" or "implementation"

SEPARATING CONTENT FROM PRESENTATION IS THE MOST IMPORTANT PRINCIPLE OF INFORMATION ORGANIZATION

Three Types of "Stuff" or Kinds of Information

CONTENT – "what does it mean" information

STRUCTURE – "where is it" or "how it is organized or assembled" information

PRESENTATION – "how does it look" or "how is it displayed" information

Instances and Classes / Categories

We can treat information as a description of, or as being attached to an INSTANCE of something, a particular realization or implementation or occurrence

Or, we can treat information as a description of a CLASS or CATEGORY of things that we are treating as equivalent

It is important to be precise about which descriptive perspective you're taking

Implications for the Concept of "Document"

Separating content from presentation gives us a different definition of "document" than we might use in ordinary language:

A DOCUMENT is a purposeful and self-contained collection of information

This definition focuses on the information content, not on the physical container or medium, format, or technology in which the information is conveyed

This broader definition can now stretch to include information extracted from databases or applications, spreadsheets, printed or web forms, as well as traditional printed documents

Document Types

Any definition of "document" allows for a notion of different types or classes or categories of documents

This idea can be very intuitive and very informal, or we can be more precise and define a MODEL OF A DOCUMENT TYPE as the rules or constraints that distinguish one type from another

This expression of the model is CONCEPTUAL and is independent of the syntax and technology in which document instances are ultimately implemented

But most of the time the model is ultimately implemented in some specific syntax like XML

Models of Document Types

A model of a document type captures the distinctions between documents that make a difference

Similar types of content occur in many document models and there is often overlap in information and structural patterns

Models of document types can be very specific ("purchase order for industrial chemicals when buyer and seller are in different countries") or very abstract ("fill-in-the-blank legal form for contract")

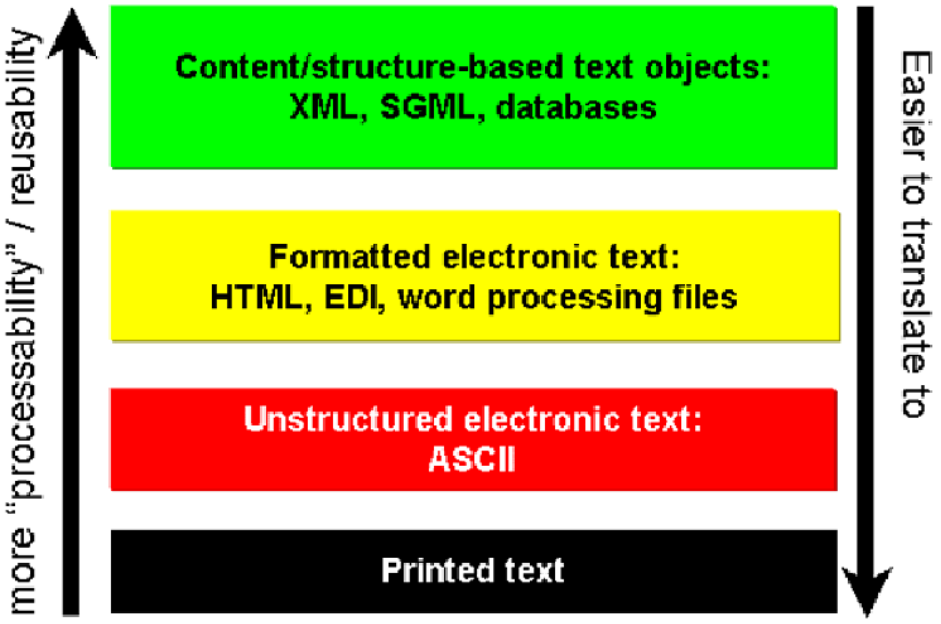
Not All Containers Are Equally Separable, Valuable or Usable

Not all information content can be completely separated from its container (sometimes the medium is the message)

But it is important to think of the information content abstractly if you can because that's the key to representing the same information in multiple formats, media, or technologies

Some information formats or representations are inherently more processable, reusable or adaptable than others

"Information IQ"



Smart Markup Creates Explicit Text Objects

How much you can do with information depends on the extent and explicitness of its internal structure or "markup"

Markup transforms a flat stream of text into a set of objects or elements that can be manipulated by other applications

HTML is rich or smart compared with EDI, RTF, or ASCII

And all of these syntaxes are poor or dumb compared with a content-oriented data model, such as those possible with XML, SGML, or a database

X12 EDI Message Fragment

```
ISA*00* *00* *08*JCPenny111 *ZZ*test22222 *971107*1220*U*00302*112240001*1*P*@
GS*PO*test111111*test22222*19971107*1220*123456789*X*003042
ST*850*4567
BEG*00*NE*10117564**19990105
REF*ZI*11
REF*1V*97-0049393
PER*OD*Chris Smith*EM*csmith@supplyworks.com
PER*RE*Tom Gerry*TE*617-861-7900x1567
DTM*074*19990120*1806
N1*BE*SupplyWorks*91*FFF
N1*BY*SupplyWorks*91*ABC1235
N1*EY*Chris Smith*92*csmith1
N2*Room 208C*SupplyWorks Corporate
N3*57 Bedford Street*Building 2
N4*Lexington*MA*02173
N1*ST*SupplyWorks*ZZ*SW1
N1*SE*EC Office Supplies*1*98629321
N2*Purchase Department
N3*123 Main Street
N4*NY*NY*03417
PO1*1*24*EA*1.86**UI*999999922234
TXI*TX*9.41
PO1*2*12*CA*25.67**UI*999999955567
TXI*TX*4.51
CTT*2*42
SE*24*4567
GE*1*123456789
IEA*1*112240001
```

EDIFACT Message Fragment

```
20000305:102'DTM+158:20000305:102'DTM+159:20000722:102'NAD+SU+9876543
NY'NAD+MI+88835::92'GIS+37'NAD+ST+72681::92'LIN+++93235494:IN'PIA+1+0
04'RFF+ON:XXX00004'QTY+79:6660:EA'DTM+51:19991225:102'DTM+52:20000304
91225:102'DTM+11:20000302:102'SCC+1++W:16'QTY+1:960:EA'DTM+158:200003
20000313:102'SCC+4++W:16'QTY+1:900:EA'DTM+158:20000320:102'QTY+1:900:
:1080:EA'DTM+158:20000403:102'QTY+1:1080:EA'DTM+158:20000410:102'QTY+
'QTY+1:630:EA'DTM+158:20000424:102'QTY+1:990:EA'DTM+158:20000501:102'
:102'QTY+1:810:EA'DTM+158:20000515:102'QTY+1:810:EA'DTM+158:20000522:
0529:102'QTY+1:810:EA'DTM+158:20000605:102'QTY+1:630:EA'DTM+158:20000
20000619:102'QTY+1:810:EA'DTM+158:20000626:102'QTY+1:810:EA'DTM+158:2
158:20000710:102'QTY+1:766:EA'DTM+158:20000717:102'SCC+2'QTY+3:12610:
:20000416:102'SCC+3'QTY+3:17485:EA'DTM+51:19991225:102'DTM+52:2000052
0:EA'UNT+73+770001'UNZ+1+77'UNB+UNOA:2+BFT:ZZ+CAI:ZZ+000305:2338+78++
```

My Information Can Beat Up Your Information [1]

Two separate and important ideas:

- Use a syntax capable of encoding information in a rich and easily processed manner
- Use it effectively

It is almost always better to make information smart so that it can be processed by simple and generic tools than to bury the intelligence into a custom processing application tuned for a particular ad hoc syntax

Try to capture your data when it is smartest, when it is created, and keep it in that smart data model as long as you can

My Information Can Beat Up Your Information [2]

Smart data is more portable and reusable by multiple applications

Smart data can always be turned into "dumb" data, but not vice versa

Programs that work with dumb data are always more brittle because they have to do more processing to detect and recover from data errors

(FInally) How XML Implements Models of Document Types

XML gives the idea of document type a more physical, formal foundation

XML has syntactic mechanisms that capture the conceptual distinctions between document types in terms of:

- ELEMENTS (the "tags") and ATTRIBUTES used to encode their content
- Rules that govern how elements and attributes are organized
- Possible values for elements and attributes

These are the VOCABULARY and the GRAMMAR of the language defined by the document type

Comparing the Syntax of HTML and XML

```
<HTML>
<BODY>
<H1>Purchase Order (#1234)</H1>
<HR>
<H2>Buyer Information</H2>
<P>Smith and Company (Buyer # AB24567)
...
</BODY>
</HTML>
```

```
<PurchaseOrder  OrderNo="1234">
<Buyer BuyerNo="AB24567">
<Name="Smith and Company"/>
<Address1>123 High Street</Address1>
<Address2>Suite 100</Address2>
<City>Anytown</City>
<State>California</State>
<ZipCode>12345</ZipCode>
</Buyer>
<Items>
...
```

How XML's Purpose Differs from HTML's

HTML's idea of using tags to mark up pieces of text according to how they should appear on the page for people to read them is very easy to understand, which is why grade school kids can create simple web pages

XML's purpose is radically different – it is *a syntax for encoding models and instances*
of "things and processes" in ways that can be handled by applications

What kinds of information is needed to encode a model of:

- A Shakespeare play?
- A chemical molecule?
- A comic strip? SAMPLE

Many more of these XML languages or applications are listed at the Cover Pages and at OASIS

XML's Design

HTML's idea of using tags to mark up pieces of text according to how they should appear on the page for people to read them is very easy to understand

XML's purpose is radically different – it is a general syntax for encoding models and instances of "things and processes"

The big difference is that the HTML document has a fixed set of element types ("tags") that it might contain, while an XML document instance is potentially unlimited in what tags it can contain

So XML is a METALANGUAGE that can be used to define new languages; HTML is an example of one

Using XML to Encode Document Type Models

Encoding a conceptual information model in XML means choosing elements or attributes as the containers for information, adding information about data types, applying naming rules, creating structures to organize repeated content components, and so on

If you've done a careful document analysis and design, the encoding stage is relatively simple and straight-forward and can even be automated in some cases

Elements

```
<BookTitle>Moby Dick</BookTitle>
<BookAuthor>Melville</BookAuthor>
<BookPublicationYear>1851</BookPublicationYear>
<BookCategory>Fiction</BookCategory>
```

```
<Book>
  <Title>Moby Dick</Title>
  <Author>Melville</Author>
  <PublicationYear>1851</PublicationYear>
  <Category>Fiction</Category>
</Book>
```

Elements are the building blocks in XML documents

They define the hierarchy or logical "containers" by enclosing content with both a begin and end tag; the hierarchy provides context for understanding the child elements

Attributes

Elements may also one or more attributes (a name - value pair) associated only with their start tag and the values must always be quoted with matching ' or "

- <PurchaseOrder number="12" >*purchase order content* </PurchaseOrder>
- The order of attributes is not significant

Elements with attributes but no content are said to be "empty" and have a different tag syntax

- <Portrait image="bob.gif"/>

Elements {and,or,vs} Attributes

```
<Book>
  <Title>Moby Dick</Title>
  <Author>Melville</Author>
  <PublicationYear>1851</PublicationYear>
  <Category>Fiction</Category>
</Book>
```

```
<Book title="Moby Dick" author="Melville"  publicationYear="1851"
category="Fiction"/>
```

```
<Book title="Moby Dick" author="Melville" category="Fiction"
publicationYear="1851"/>
```

```
<Book title="Moby Dick" author="Melville"  publicationYear="1851"  fiction="True"/>
```

XML Schemas

The formal description of a document model in XML is called its *schema*

XML schemas (lower case "s" for now) attempt to encode the conceptual model in terms of the syntactic constructs of elements and attributes

- What elements are allowed (the *vocabulary*)
- Where the are allowed – sequence, choice, occurrence and co-occurrence (the *grammar*)
- What values they can take (*datatypes*) – string, integer, decimal, etc.

Why We Need Schemas

If you can represent these rules that define a document type in a form that is "computable" or "processable" then:

- It can guide the creation of valid document instances in editors like XML Spy or Oxygen, or when information is exported from a database or other application
- It can be a model for application programming in the development of Web forms or other GUIs or can be a template for objects in other programming environments
- It can communicate the model to others who need to create or receive document instances

XML Schemas and Schema Languages

XML has several schema languages that differ in how completely that can encode a document type's conceptual model

The most common of these are Document Type Definitions (DTDs) and XML Schemas (XSD) -- examples to follow

No schema language is perfect; there is always some compromise between:

- Expressiveness – the range of models that can be described
- Functionality – the set of features used to define a model
- Usability – the ease with which a model can be defined
- Reusability – how readily a model or parts of models can be included in another one
- ...and a range of other "ilities"

DTD for Simple Event Calendar

```
<!ELEMENT Calendar (Organization, TimePeriod, Events)>
<!ELEMENT Organization (#PCDATA)>
<!ELEMENT TimePeriod (#PCDATA)>
<!ELEMENT Events (Event+)>

<!ELEMENT Event (Title, Description?, Speaker?, DateTime, Location)>
<!ATTLIST Event
        type (Lecture | Workshop) #IMPLIED>

<!ELEMENT Title (#PCDATA)>
<!ELEMENT Description (#PCDATA | Keyword)*>

<!ELEMENT Keyword (#PCDATA)>
<!ELEMENT Speaker (Name, Affiliation)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Affiliation (#PCDATA)>
<!ELEMENT DateTime (#PCDATA)>
<!ELEMENT Location (#PCDATA)>
```

XML Schema for Simple Event Calendar

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<xs:element name="Calendar">
<xs:complexType>
  <xs:sequence>
    <xs:element name="Organization" type="xs:string"/>
    <xs:element name="TimePeriod" type="xs:string"/

    <xs:element name="Events">
     <xs:complexType>
      <xs:sequence>
        <xs:element name="Event" maxOccurs="unbounded">
      <xs:complexType>
      <xs:sequence>
        <xs:element name="Title" type="xs:string"/>
        <xs:element name="Description" minOccurs="0">
          <xs:complexType mixed="true">
            <xs:choice minOccurs="0" <MaxOccurs="unbounded">
             <xs:element name="Keyword" type="xs:string"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
        <xs:element name="Speaker" minOccurs="0">
 ...
```

Why Transform?

You have information that is too "smart" for the web or end users to handle (for example, a database that will be queried, a complex information model for which simpler views are needed for some users)

You need to RE-PURPOSE information – extracting and / or formatting the same piece of information is many different ways, producing a different document type targeted for a different user or purpose

You have to support a variety of output devices that have different capabilities– often called RE-PACKAGING or TAILORING

You need to conform to a structural or formatting standard that is different from your company or organization's information model

Your "web service" needs to convert an "inbound" non-XML document to XML, or convert XML to a non-XML format for the "outbound" document

Technical Example of Re-purposing – Product Database / Catalog

The XML instance: product-db.xml

The target HTML instance viewed as an internal database: product-db.htm

The target HTML instance viewed as a catalog for external customers: product-catalog.htm

The XSLT transformation for the database: product_db.xsl

The XSLT transformation for the catalog: product_catalog.xsl

The Architecture of XML Transformation

A particular transformation may apply to more than one document – it might be used to enforce standards for all instances of a document type

```
<?xml-transform type="text/xsl"
href="standard_style_for_this_doctype.xsl"?>
```

A given document instance may have different transforms applied to it in different contexts (like for different audiences, output devices, etc)

```
<?xml-transform type="text/xsl" href="transform_for_pda.xsl"?>
<?xml-transform type="text/xsl" href="transform_for_verbose_mode.xsl"?>
```

A transformation may turn one XML file into one or more output files (like the XML source file for each lecture being transformed into a set of linked HTML slides)

Assignment 2 - "Getting Familiar With XML"

Use an XML editor and learn how an XML instance, schema, and transform are "hooked in"

Next week's section will be "hands-on" with specific help on XML syntax and mechanics

Due on 22 September

You'll use XML in Assignments 3 and 7

An Admonition and Preview of Lecture #5

The best thing about XML is the ease with which anyone can create a new "language" or "vocabulary"

The worst thing about XML is the same as the best thing: the ease with which anyone can create a new ;anguage or vocabulary

Readings for INFO Lecture #5

George Lakoff, Women, Fire, and Dangerous Things, Chapters 1 and 2 (pages 5-57)

Robert J. Glushko, Paul Maglio, Teenie Matlock and Lawrence Barsalou, "Categorization in the Wild," Trends in Cognitive Sciences, 12(4): 129-135, April 2008