

The Inside Track November 2000

Did you see the Blondie comic on September 11, 2000? Check it out on www.blondie.com. In case you don't have a browser just handy, I'll sketch it out for you. Frame 1, Dagwood is in bed and Blondie tries to wake him up with a gentle call "DAGWOOD!" Frame 2, Dagwood sleeps on, so Blondie calls more insistently, "**DAGWOOD BUMSTEAD!!**" Frame 3, Blondie is having no luck waking Dagwood. Blondie is fed up, so she calls "**DAGWOOD BUMSTEAD DOT COM!!**" Dagwood flies up vertically off the mattress saying "**OMIGOSH! THAT MEANS BUSINESS!!**"

So, after 70 years, Blondie and Dagwood have entered the Internet age. Have you? And of course while .COM indeed means business, there is no more important technology to the future of business on the Internet than XML. Yes, XML. Say it again, memorize it, make it your friend, and love it from the depths of your little programming heart. XML, it just rolls off the tongue!!

OK, so you've heard about XML (hopefully not for the first time just now) the Extensible Markup Language and you've got an idea XML is important, everyone says it is. Let's have some details.

XML is very simple. XML is complex. No contradiction here. Think about computers, they push around things called bits, just zeros and ones, nothing simpler in existence. It's the same with XML. XML is very simple, it's what you do with it that makes it complex. XML and HTML (Hyper Text Markup Language) both derive from SGML (Standard Generalized Markup Language). Like HTML, XML is text. XML looks a lot like HTML. XML uses a tag-like structure similar to HTML. Unlike HTML, XML is very carefully defined. Because XML is so carefully defined, it is easy for a program to parse. Because it's so simple, XML can be hard to understand. When was the last time you converted an integer represented as 32 binary digits to decimal in your head?

If you have a chunk of text that follows the rules of XML, you have an XML document. Here is one now:

```
<PEOPLE Type="not real">
  <PERSON FirstName="Blondie" LastName="Bumstead" />
  <PERSON FirstName="Dagwood" LastName="Bumstead" />
</PEOPLE>
```

If you want the formal definition of XML, I suggest you check out www.w3.org/xml. I hope you have several hours of free time! For an incomplete and less rigorous definition I'll give you some rules:

- 1) An element begins with "<", an element name, zero or more attributes and ">", for example "<PEOPLE>".
- 2) An element ends with "</", the same element name used to begin the element and ">", for example "</PEOPLE>".
- 3) A "child" element may be inserted between its parent element's start and end. No part of a child element may extend beyond its parent's end. This means you can't write "<A>".
- 4) If an element has no children there is an abbreviation possible. Such an element will start with "<", have an element name, have zero or more attributes and end with ">", for example "<PERSON Name='Karl' />".
- 5) An attribute consists of an attribute name, "=" and an attribute value.
- 6) An attribute value begins with a quote (single or double), has zero or more characters not including the same quote, or ampersand and ends with a quote matching the one that started the attribute value. Use "&", "'" and """ in attribute values to represent the ampersand, single quote and double quote characters, respectively. For instance "<COMPANY Name='"Red" Smith&s Software'"> would produce a company name of "Red" Smith's Software.

- 7) Element and attribute names are made up from letters, digits, periods, dashes, underscores and colons.
- 8) XML is case sensitive. Beginning and ending element names must match exactly. The attribute names "Name" and "NAME" are not the same attribute.
- 9) Every XML document must have a single "root" element.

That's about 95% of the important rules for XML. You can pick up the rest by imitation. I was once told by a professional instructor that the best programmers were excellent thieves. So look for examples and steal!

To make these rules a bit more concrete, let's look at a slightly larger example.

```
<PEOPLE Type="not real">
  <FAMILY LastName="Bumstead">
    <PERSON FirstName="Blondie" Gender="Female"/>
    <PERSON FirstName="Dagwood" Gender="Male"/>
    <CHILDREN>
      <PERSON FirstName="Alex" Gender="Male"/>
      <PERSON FirstName="Cookie" Gender="Female"/>
    </CHILDREN>
    <PET FirstName="Daisy" Type="Dog"/>
    <NEIGHBOR LastName="Wooley"/>
  </FAMILY>
  <FAMILY LastName="Wooley">
    <PERSON FirstName="Tootsie" Role="Wife"/>
    <PERSON FirstName="Herb" Role="Husband"/>
  </FAMILY>
</PEOPLE>
```

The element with name "PEOPLE" is the root of this XML document. It has one attribute with a name of "Type" and the attribute's value is "not real". There are two child elements both "FAMILY". The first FAMILY consists of PERSON's with attributes of FirstName and Gender. The FAMILY also contains CHILDREN, a PET and a NEIGHBOR. The second FAMILY consists of two PERSON's with attributes FirstName and Role.

Pretty simple! Cut-and-paste the chunk of XML above into Notepad and save it with a name ending in .XML, e.g. People.XML. Now open or drag-and-drop that file in Microsoft's Internet Explorer. Or better yet, download XML Notepad from:

<http://msdn.microsoft.com/xml/notepad/intro.asp>.

I'm sure you noticed that the Bumstead family identified members by Gender and the Wooley family identified family members by Role. Just like in real life, various companies might refer to its workers as "employee", "team member", "associate", "staff" or any of a number of other names and let's not even consider other countries!! One solution might be to have standards, but perhaps even better is XSL (Extensible Stylesheet Language), or its big brother XSLT (XSL for Transformations), which allows you to transform one XML document into another XML document or an entirely different type of document. We'll definitely be looking at XSL/XSLT in a future column. As a quick example, here is a chunk of XSL (note that XSL is just a special type of XML document!) that will change a PERSON's Role attribute to an appropriate Gender attribute.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <xsl:apply-templates select="*" />
  </xsl:template>
```

```

        <xsl:template match="*|@*|text()|cdata()|comment()|pi() ">
            <xsl:copy><xsl:apply-templates
select="*|@*|text()|cdata()|comment()|pi() "/></xsl:copy>
        </xsl:template>

        <xsl:template match="PERSON/@Role">
            <xsl:attribute name="Gender">
                <xsl:choose>
                    <xsl:when test=". [.='Husband'] ">Male</xsl:when>
                    <xsl:when test=". [.='Wife'] ">Female</xsl:when>
                </xsl:choose>
            </xsl:attribute>
        </xsl:template>

</xsl:stylesheet>

```

The line containing PERSON/@Role selects the Role attribute of a PERSON. The following lines replace it with a new attribute named Gender with values of Male and Female instead of Husband and Wife, respectively. Neat! If you want to try this out, there are a couple of steps to follow. First add the following two lines to the top of People.xml:

```

<?xml version="1.0"?>
<?xml:stylesheet type="text/xsl" href="people.xsl"?>

```

These are called processing instructions. The second line refers to "People.xsl". This is the XSL file that will be applied to the XML document when it is loaded. So cut-and-paste the XSL above into a file called People.xsl" in the same folder as People.xml. You'll want to use Internet Explorer, IE, 5.0 or later. To see the result of an XSL transformation you have two options. One is to load a couple of nice extensions to IE found at:

<http://msdn.microsoft.com/downloads/webtechnology/xml/iexmltIs.exe>

Because another use of XSL is to transform XML into HTML, normally you don't want to see any XML output from an XSL transformation. The transformation performed above does not display anything because it only produces new XML! After all, when you run a program you don't expect the source code to showup in the output do you? The extensions above allow you to view the output of an XSL transformation and to validate an XML document. So, once you have installed the extension, you can right-click on the browser page, select "View XSL Output" and you can see the new XML with Role transformed into Gender.

Another way to view this output would be to put it into a browser window but then we'd have to tell IE how to format the XML, a somewhat tedious task. Another alternative is to save the output from the XSL transformation into a new XML file and use the default formatting that IE provides. That's what the following VBScript does (the result is in "People2.xml"):

Option Explicit

```

Dim xmlDoc      'As DOMDocument
Dim xslDoc      'As DOMDocument
Dim xmlOut      'As String
Dim fso         'As FileSystemObject
Dim xmlFile     'As TextStream

Set xmlDoc = CreateObject("Microsoft.XMLDOM")
Set xslDoc = CreateObject("Microsoft.XMLDOM")

```

```
xmlDoc.Load "People.xml"  
xslDoc.Load "People.xsl"  
  
xmlOut = xmlDoc.transformNode(xslDoc)  
  
Set fso = CreateObject("Scripting.FileSystemObject")  
  
Set xmlFile = fso.CreateTextFile("People2.xml")  
xmlFile.Write xmlOut  
xmlFile.Close
```

Now you know a bit about how XML looks. Perhaps you have begun to see that XML is a really, really good way of describing data! We haven't even touched on DTD's, XML Schema's and SOAP, among other XML related technologies. I guess you'll have to trust me or find out for yourself just how true a statement that is.

As some evidence for XML's value, consider these points:

- 1) The XML 1.0 standard was published in February of 1998, see www.w3.org/xml.
- 2) XML has been supported since Internet Explorer 4.0 and will be supported by Netscape Navigator when 6.0 releases.
- 3) XML is a core technology in Great Plains new release 6.0 of eCommerce, see http://www.greatplains.com/search/document.asp?link=pressreleases/e-commerce_r6.htm.
- 4) Several industry groups, see www.BizTalk.org, www.openapplications.org (formed in 1995, Great Plains is a member) and www.xml.org (formed in 1993), exist to promote XML and develop standard XML documents to facilitate exchange of data via XML.
- 5) XBRL (Extensible Business Reporting Language) is an XML based standard for exchange of financial information, see www.xbrl.com. Great Plains is a member of this organization.
- 6) XML will be a core technology in Great Plains next generation products!

XML love it, but never leave it.

Leaving you for now,
Karl Gunderson
Technical Evangelist