

# NetJava Expert

Volume 3 • Issue 1

February / March 1999

A Technical Forum for AS/400 Internet and Java Professionals

## Contents

### XML: Managing Data on the Web

by Teresa Pelkie .....1

### A Frameworks for E-business

by Don Denoncourt .....2

Emails, Pointers, Advice .....5

### A Tour of XML Applications

by Don Denoncourt .....7

### A Java GUI for DSPFFD

by Alex Garrison .....11

### Enabling HTTP Services on Your Domino Server

by Tamas Perlaky .....16

### AS/400 Servlets 101

by Sonjaya Tandon .....18

### WebSphere Application Server for AS/400

by Lisa Wellman .....23

### Putting Network Address Translation to Work

by Chris Green .....27

### Choosing the Right Internet Development Technology, Part 4—HTML

by Randy Dufault .....29

anje://HotSites .....32



## XML: Managing Data on the Web

by Teresa Pelkie

The newest addition to the selection of Web technologies is another language known as Extensible Markup Language (XML). XML is a subset of the Standardized General Markup Language (SGML), which is a markup language for describing large, complex manuals. SGML was developed in 1974 by Charles Goldberg who also developed its ancestor Generalized Markup Language (GML) for IBM in 1969.

XML, like SGML, is a metalanguage used to describe or create other markup languages. The most widely deployed SGML application is HTML, but HTML was not designed to do the things that are demanded of it today. Consequently, members of the SGML Working Group came up with a compact version of SGML that can be processed on the Web. The initial XML draft specification was presented in 1996 and became a World Wide Web Consortium (W3C) Recommendation on February 10, 1998

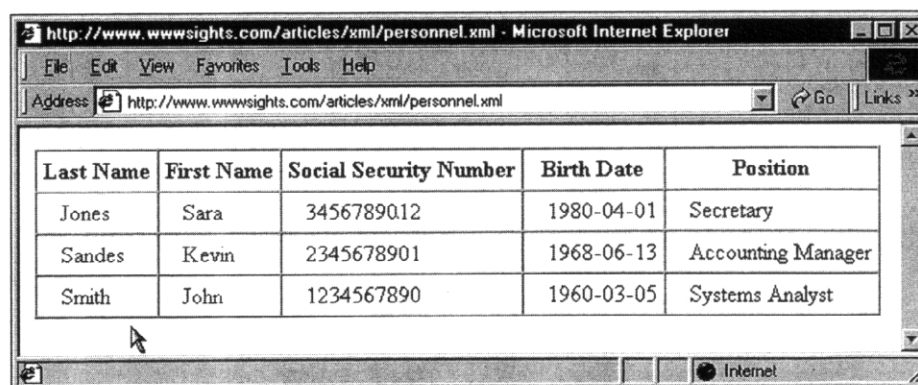
([www.w3c.org/TR/REC-xml](http://www.w3c.org/TR/REC-xml)). XML-related proposals—such as the Extensible Stylesheet Language (XSL), XLink, Xpointer (linking), XML Schema, XML-Query Language (XML-QL), and others—are still in the working stages. Nonetheless, XML is already being used in industry and on the Web, and it is gaining attention.

### How XML Works with Data

Although HTML is responsible for presenting the contents of a document on the screen, it says nothing about the meaning of the content. XML separates the content from the display by describing the content or data in a meaningful schema. XML is called extensible because it allows you to create custom tags to fit your needs. These tags describe the different parts of the document content as a label would.

For example, the HTML markup `<H1>First Central Bank</H1>` displays the

*continued on page 3*



Last Name	First Name	Social Security Number	Birth Date	Position
Jones	Sara	3456789012	1980-04-01	Secretary
Sandes	Kevin	2345678901	1968-06-13	Accounting Manager
Smith	John	1234567890	1960-03-05	Systems Analyst

Figure 1: The XML data is processed by XSL and arranged alphabetically into a table in Internet Explorer 5.0 Beta.

continued from page 1

content in large bold type but says nothing about the data itself. The XML markup <company> First Central Bank</company> describes and identifies the content and distinguishes it from other content or data in the document.

XML's custom tags give meaning to the data, which enables communication, exchange, and processing of information. XML is not concerned with how the data is displayed in the browser. That is the job of XSL, a style sheet language that formats the data for visual presentation.

XML's most important use is in the exchange of data. XML tailors data for exchange of information between databases, applications, and operating systems. It allows the data to be viewed in multiple ways. Two important applications in which XML is used are messaging and financial exchange. XML's tree structure provides a navigation structure and is used for searching, filtering, and managing documents. XML also handles international character sets.

Many XML vocabularies or languages are being developed. These vocabularies are known as a Document Type Definition (DTD). A DTD is a structured format for describing an industry-specific area of data. DTDs enable developers and users to speak the same language by agreeing on a common standard.

Microsoft's Channel Definition Format (CDF), used in Internet Explorer 4.0 to define channels, was one of the first examples of an XML language. Another example is the Open Financial Exchange (OFX), which provides a standard language for banks to allow customers to manage their finances online. The Resource Description Format (RDF) for describing metadata will be used in Netscape 5.0 to store bookmark and history information. Future versions of Microsoft Office will use XML as one of two native formats for storing data.

At this point, I will explain how to create a DTD and the XML document markup by using the data in three records of a personnel database as an example. I will use XSL to display the document as shown in Figure 1. You can see this example at [www.wisights.com/articles/xml/personnel.xml](http://www.wisights.com/articles/xml/personnel.xml). You need to use Internet Explorer 5.0 Beta to see this example. When you install this browser, you will have the option to keep your present version of Internet Explorer as well.

### The XML Document

XML is case-sensitive and most reserved words are upper case. An XML document contains six types of markup:

- *Elements* are used to create the custom tags, which describe the data. Elements form the schema of the document. All elements have an opening and closing tag such as <element></element>. Tags must be closed, nested in order, and follow a hierarchical structure. Empty elements that do not contain any data or other tags, such as an image tag, are represented as <img/>. Only one root element is allowed in a document.
- *Attributes* are name/value pairs that provide an element with additional information. Attribute values must be in quotes.

- *Comments* begin with <!-- and end with -->.
- *Entities* are keywords, reserved characters, repeated text, or content from an external file.
- *Processing Instructions (PIs)* are instructions to the browser that start with <? and end with ?>.
- *CDATA* tells the XML parser to ignore the reserved characters.

### Document Type Definition

The DTD lists the tags, attributes, and entities that are allowed in the document, the conditions under which they can be used, and the nesting structure of the tags. Many documents can reference the same DTD. An XML document does not have to contain a DTD; it can function the same without it. An XML document that follows all the markup rules is said to be well-formed. An XML document that is well-formed and contains a DTD is said to be valid.

Figure 2 shows the source code for my XML example. Section A states that this is an XML document. The DTD is named personnel (Section B), and the declarations that follow are enclosed in brackets (Sections B through Section G). The first element declaration, personnel, declares the root element or node for the entire document (Section C). Personnel can contain only one child element or node, employee. The plus sign (+) indicates that employee can be repeated. The element employee is declared next (Section D), and it is allowed to contain four child elements or nodes: name, birthdate, ssn, and position. The question mark (?) after birthdate indicates that it is optional. An attribute declaration for the element employee is next (Section E). The name of this attribute is department and it can take three values: mis, accounting, or personnel. Accounting is positioned again at the end to indicate that this is the default. The declaration for the element called name is next (Section F) and it contains two child elements or nodes. The data types for the element are assigned next in the same order in which they were declared (Section G). All of these elements can contain parsed character data. Notice that three elements, personnel, employee, and name do not contain any data, only other elements.

This is a simple DTD. There are many more rules and conditions that can be applied. My XML example (Figure 2) shows the DTD at the beginning of the document. The DTD can be placed at the beginning of an XML document or it can be referenced in an external file. If the DTD were referenced from an external file it would be written as follows:

```
<!DOCTYPE personnel SYSTEM  
"personnel.dtd">
```

### The Document Element

The document element is the root element or node for the entire document. The document element in my example is personnel as shown in Section I of Figure 2. All of the content and other markup in this document is contained between the <personnel></personnel> tags. The tags and

```

<?xml version="1.0"?>
<DIV xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <TABLE BORDER="1" cellspacing="0" cellpadding="4">
    <THEAD STYLE="background-color:#CCFFFF; color:navy">
      <TH>Last Name</TH>
      <TH>First Name</TH>
      <TH>Social Security Number</TH>
      <TH>Birth Date</TH>
      <TH>Position</TH>
    </THEAD>
    <xsl:for-each
      select="personnel/employee"
      order-by="+ name/last">
      <TR STYLE="background-color:#ffffee">
        <TD STYLE="padding-left:1em">
          <SPAN>
            <xsl:value-of select="name/last"/>
          </SPAN>
        </TD>
        <TD STYLE="padding-left:1em">
          <SPAN>
            <xsl:value-of select="name/first"/>
          </SPAN>
        </TD>
        <TD>
          <SPAN>
            <xsl:value-of select="ssn"/>
          </SPAN>
        </TD>
        <TD STYLE="padding-left:1em">
          <SPAN>
            <xsl:value-of select="birthdate"/>
          </SPAN>
        </TD>
        <TD STYLE="padding-left:1em">
          <SPAN>
            <xsl:value-of select="position"/>
          </SPAN>
        </TD>
      </TR>
    </xsl:for-each>
  </TABLE>
</DIV>

```

Figure 3: The XSL document (personnel.xml) uses style sheets and instructions to process and display data.

IBM, Lotus, Infoseek, and Silicon Graphics are among the many members of this working group. XML will work with HTML to separate the interface from the underlying data. In the same way that HTML provides a universal format to view information, XML offers a universal method to describe and manage data.

**Expert**

Teresa Pelkie is the owner of WWW Sights in the San Diego, California, area. She can be reached at [teresa@wwwsights.com](mailto:teresa@wwwsights.com).

## Emails, Pointers, Advice

### Pointer

#### Java's Static\_INITIALIZER and the Singleton Design Pattern

Object-oriented programming languages such as Java can easily benefit from reusing coding patterns. The authors of *Design Patterns: Elements of Reusable Object-Oriented Software* advocate a pattern called Singleton. A Singleton works very well in the circumstance where you only ever want one instance of an object. The obvious example for a Singleton is an IBM AS/400 Toolbox for Java connection object. The Java implementation for a Singleton AS/400 connection uses static fields and a special function called the static initializer function, as shown in the Java class in Figure 1. The fields of a Java class that are modified with the static keyword are fields whose value is shared among all instances of that class. They are similar to data areas in an RPG program because the same value is shared. In Figure 1, all the fields are modified with the static keyword. Note that all but the AS400 field have their initial values specified. So, where does the AS400 object get initialized? It's initialized in the static initializer. The static initializer looks like a function because it has curly braces, but it has

```

import com.ibm.as400.access.*;
class Profile {
  static String USER = "DENONCOURT";
  static String PASS = "secret";
  static String URL = "mc url";
  static AS400 as400;
  static {
    as400 = new AS400(Profile.URL,
      Profile.USER, Profile.PASS);
  }
}

```

Figure 1: An AS/400 Profile class is an excellent use for the Singleton design pattern.

no function name; it only has the static keyword. If the static initializer has no function name, then how are you supposed to call it? You don't. The Java Virtual Machine (JVM) calls it the first time its class is used in an application. That means that I can simply use `Profile.as400` and, the first time, the JVM invokes the static initializer. On subsequent

structure are built according to the DTD. Notice that the order in which the DTD declares the elements is the same order in which they appear in the document element. The employee element has a department attribute but contains no content of its own. Rather, it contains four child elements, one of which, name, contains two child elements of its own. The employee element appears three times but can appear many more.

The XML tags are descriptive of the data they contain. They are playing the same role as the schema does in a database. The tags don't say what to do with the data; they just say what the data is.

### Using XSL to Process the Data

The data in an XML document is processed in another document using XSL. XSL (refer to [www.w3.org/TR/WD-xsl](http://www.w3.org/TR/WD-xsl)) is still in its working group phase at W3C. It provides not only a style sheet language to specify formatting semantics, but also a language for transforming XML documents. XSL does a lot more than apply styles. An XSL processor can evaluate, arrange, rearrange, and process information. The data in an XML document can be easily added to and modified.

In my example, the XSL document (shown in Figure 3) is called in Figure 2, Section H and processes the data in Figure 2 to display the result shown in Figure 1. The first two lines of the document contain processing instructions (Figure 3, Section A). You can see style sheets used throughout the document by the STYLE="value" expression.

The <xsl></xsl> start and stop tags define the root element of the XSL document as shown in Section B of

Figure 3. The <xsl:for-each select...> instruction in Section B says that, for every employee that is found in the personnel element, do everything below until the closing tag </xsl:for-each>. It processes the instruction in the cells of the table and orders the rows by last name. The <xsl:value-of select instruction in Section C returns the content of the element. This instruction is processed in all five cells so that the information for each employee is entered under the headings Last Name, First Name, Social Security Number, Birth Date, and Position in Figure 1. This process continues until the data for every employee is entered into a row in the table.

XML describes the data and organizes it into a structured format so that it can be delivered in a consistent way. XSL is the processing power that turns this raw data into information. It provides a language for transforming XML documents and presents an XML vocabulary for specifying formatting semantics. XSL allows XML data to be manipulated and displayed in many different ways, generating different virtual XML documents in response to user queries. Adobe, Microsoft, Arbortext, Inso, Bitstream, Enigma, IBM, Lotus, Netscape, RivCom, and Sun Microsystems have taken part in the development of XSL. Remember that XSL is presently a W3C proposal and is still being revised.

### Looking Ahead

At the Future of HTML Workshop held in San Jose, California, in May of 1998, the W3C decided that the next generation of HTML would be cast as a suite of XML tags. The W3C has set up a new HTML Working Group that is expected to last until 2000. Microsoft, Netscape, Adobe,

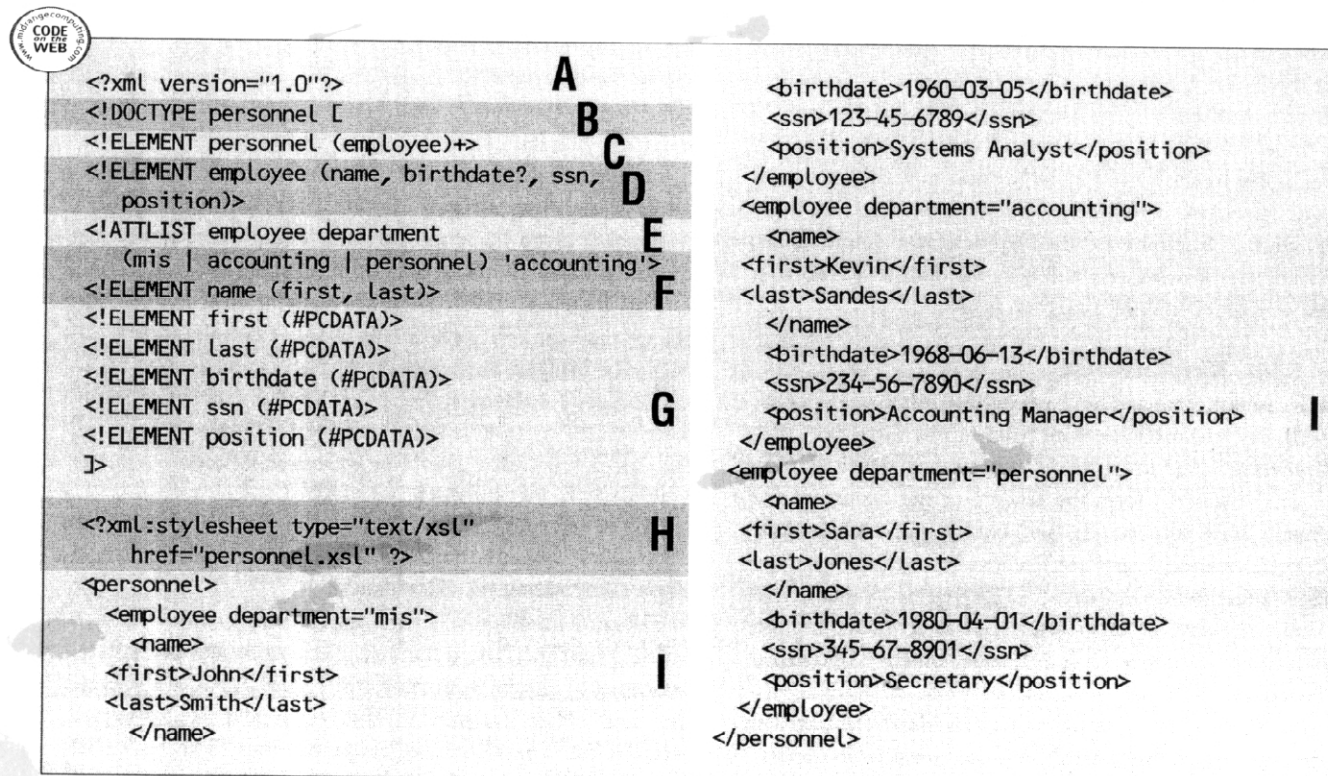


Figure 2: This XML document contains a DTD and markup for the data contained in three personnel records.