

AN INTRODUCTION TO XML

Mark E. Donaldson

What is XML?

XML stands for eXtensible Markup Language. Like HTML, XML derives from the granddaddy of all markup languages: SGML (Standard Generalized Markup Language). SGML is a metalanguage, or a system for defining markup languages such as HTML. XML is also a metalanguage, a subset of SGML designed for use on the Web. As with SGML, you can use XML to define different markup languages for specific uses, particularly for data representation.

HTML has limitations that make it less than perfect in advanced Web applications:

- It is a presentation technology: It combines the data with the display of the data, which makes separating the two difficult.
- It has a fixed tag set; you cannot extend HTML with your own application-specific tags.
- It is "flat": You cannot specify a hierarchy of data that provides details such as containment and importance.
- It does not allow easy transmission of data to the client for further processing. Instead, HTML is continually generated by the server, and the client is only a display engine.
- It provides only one "view" of your data. If you want to provide different views, you have to regenerate the data and the complete HTML page on the server for display on clients. If the data was separated from the display, you could download the data to clients and then send various views of the data as needed.
- It is not easily machine-readable or human-readable. HTML is not very consistent. Some tags have matching start and end tags (like `<body>` and `</body>`) while others have start tags but no end tags (like `<p>` and ``). HTML parsers have to handle this somewhat random formatting.

HTML and XML are standards created by the World Wide Web Consortium, or the W3C. The W3C's members realized that for the Web to continue to grow, there must be a way to separate the data from the display of Web pages--thus XML was born. Why not just use SGML? Without going into the gory details, suffice it to say that SGML is extremely complex (the specification is over 500 pages); the XML standard is far simpler (a paltry 26 pages).

Although strictly speaking, XML is a data-markup specification, the term has come to include a variety of related technologies, such as XML DOM, XSL, XLL, XML namespaces, and VML. In this article, we'll look at XML, XSL (eXtensible Style Language), and XML DOM (Document Object Model), and we'll introduce the other standards.

Anatomy of an XML Document

The XML 1.0 standard can be found at www.w3.org/TR/REC-xml. Be warned, though: It is not the best place to start learning XML but it is a good resource to check out when you finish this article.

Before looking at the first example, let's get some XML nomenclature down. As in HTML, an XML start tag is denoted by less-than and greater-than marks like this: `<tag>`. An end tag is the same, but with a forward slash after the less-than sign: `</tag>`. A tag pair with its content, such as `<movie>Gone with the Wind</movie>`, is called an element. Everything between a start and end tag is called an

AN INTRODUCTION TO XML

Mark E. Donaldson

element's content. In the previous example, Gone with the Wind is the content of the movie element. XML start tags can also have attribute lists, such as `<tag attribute1="100" attribute2="200">`.

XML documents are considered well formed if they follow the rules of the standard (more on this later). The most basic rules you will encounter are that every element with a start tag must have a corresponding end tag, and the end tag of a nested element must appear before the end tag of the element containing it. For example, this is not legal:

```
<auction_item>  
<description>  
</auction_item>  
</description>
```

But this is:

```
<auction_item>  
<description>  
</description>  
</auction_item>
```

The W3C's standard for XML has the complete list of rules your documents must follow. A well-formed XML document must contain at least one element. An element called the root, which is not within the contents of any other element, must also appear in the document. Every XML document will be a hierarchical list of elements because of this. The hierarchy (child-parent-sibling relationships) is defined by nesting elements inside of other tags.

To illustrate this, let's look at an example. Figure 1 contains some sample airline flight data in XML format. You can see that XML is relatively readable. The meanings of the tags and their contents are pretty self-explanatory. Imagine this data displayed in an HTML table. It would be almost impossible to pull the data apart from the display. But with the data in XML format, we can easily share this information with a travel agent, store it in a database, or format it for display.

Now let's dissect this example further. The first line is the XML standard header:

```
<?xml version="1.0" ?>
```

This header tells the processor (the IE5 XML parser and browser in this case) that the file is an XML file and also specifies which XML version (there's only 1.0 so far) the file conforms to.

The next line of the document is a comment:

```
<!--An example of airline flights in XML-->
```

XML uses the same comment scheme as HTML, so this is probably already familiar to you.

Unlike HTML, all start tags in XML documents must have matching end tags, so the start tag at the top of the hierarchy, `<flight_schedule>`, must have a matching end tag, `</flight_schedule>`, at the end of the document.

AN INTRODUCTION TO XML

Mark E. Donaldson

The rest of the XML document is a series of elements delimited by start and end tags. The tags in this example describe a flight schedule with two flights. Each flight contains the following hierarchical data:

- Origination
 - City
 - Airport
- Destination
 - City
 - Airport
- Connection
 - City
 - State
 - Airport
- Airline
- Price
- Aircraft
 - Manufacturer
 - Model
- Departure_time
- Arrival_time

Well-formed vs. Valid XML Documents

There are two ways to code an XML document correctly: It can either be well formed or valid. A document is considered well formed if it adheres to rules laid out in the XML standard. It is considered valid if it adheres to a document template for the data or to a schema.

Validity is important for exchanging data using XML. What if I have an invoice XML document and want to exchange it with my business partners? Being able to tell my partners what format I require and what format my documents will adhere to is important.

There are currently two ways for specifying schemas for XML documents: the Document Type Definition (DTD) and the XML Schema. DTDs are part of the XML 1.0 standard, so they are currently the most prevalent. The problem with DTDs is that they use an unintuitive syntax. Another shortcoming is that they don't let you specify the type of various elements

The XML Schema specification is an alternative being considered by the W3C that was proposed by Microsoft and other companies in the XML industry. XML Schemas use an XML-like syntax to describe documents. In addition, XML Schemas provide the ability to describe the types of elements in a document (through XML-Data). In our opinion, Schemas are easier for non-SGML pros to read, understand, and create, compared with DTDs.

To give you an idea of the differences between DTDs and XML Schemas, take a look at the files included in our download, which show DTDs for our flight data versus the XML Schema. Which do you understand quickest, the DTD or the XML Schema?

XML gives you the ability to easily define your own document types and tags, but if everyone comes up with their own common document types, nobody will be able to exchange similar documents. There are two proposed "common document" standards groups, and companies are quickly choosing sides.

AN INTRODUCTION TO XML

Mark E. Donaldson

- BizTalk. This Microsoft-led group hopes to create a foundation for common business documents such as invoices and purchase orders.
- XML.org. This site is basically an anti-Microsoft effort that seeks to use DTDs to define basic business documents.

It's too early to tell which standard will win at this stage of the game. Most likely there will be two or three popular document formats, and conversion utilities will help people migrate documents from one format to another.

XML Document Object Model

Once you have an XML document, what can you do with it? In the next section, we'll look at how to display an XML document by converting it to HTML, but the more common scenario is that the XML will be read or manipulated using the XML Document Object Model (DOM). When an XML processor parses an XML document, the processor stores the document in an in-memory tree. The DOM is a programmatic interface into that tree, which lets you read, add, delete, and edit nodes of the tree.

Figure 2 shows our sample file as it would be stored in an in-memory tree. Every DOM tree starts with a document object where all data is stored.

In IE5, the XML DOM is accessible via any programming or script language. For example, the following JavaScript will change the airline in the first flight to American Airlines:

```
var myDocument
  = new ActiveXObject("microsoft.xmlDOM");
myDocument.load("flights.xml");
myDocument.documentElement.childNodes.item(0).childNodes.item(3).text
  = "American Airlines";
```

The first line of this script creates an empty instance of a DOM object. The next line loads our data file into the DOM using the document.load() method. Finally, the last line uses a string of commands to navigate to the first flight's airline node and to change the text to "American Airlines." It does this through the childNodes attribute, which provides access to the list of child nodes using a numerical index. The DOM provides many other ways to access nodes by name and other criteria.

This example is pretty trivial, but it gives you a good first exposure to what DOM programming is like. If you are going to do any work with XML, you'll need to understand the DOM and learn how to manipulate and traverse document object trees. To help you learn more about the DOM, a sample HTML file that loads an XML file and then invokes DOM script commands has been provided as part of the download for this article.

Displaying XML

For displaying XML data, you have two choices: apply a Cascading Style Sheet (CSS) to generate HTML styles based on the data, or use another XML technology that will soon be a standard, the eXtensible Style Language.

AN INTRODUCTION TO XML

Mark E. Donaldson

The choice depends on the complexity of the data display. CSS is appropriate if you want basic output. For example, you could generate a simple table and have flights show up in gray and airlines in blue with a Helvetica font. For more complex scenarios, XSL provides more power. If you want users to be able to sort a table of flights by clicking on a header and you also want to provide several different ways to display your data, XSL is the superior choice. Also, XSL uses an XML-like syntax for describing how to convert the XML to HTML, so it may not be as foreign to you as CSS.

Now let's take our flight example and convert it to a displayable format using XSL--Figure 3 shows how. The first interesting line is a standard heading for XSL (you can learn more about its exact meaning in the XSL working documents at the W3C).

```
<xsl:stylesheet xmlns:xsl
  ="http://www.w3.org/TR/WD-xsl">
```

The next line tells the XSL processor to select every record in the XML document for processing.

```
<xsl:template match="/">
```

The XSL file contains HTML tags that are rendered just as they appear in the XSL file. These are the HTML elements that start the table and define its header. Next we have an XSL for-each statement that iterates through each flight element and processes it by destination order:

```
<xsl:for-each select="flight_schedule/
flight" order-by="destination">
```

In the for-each loop, the HTML tags that produce the elements of the table are generated with XSL value-of select statements, which pull the desired values out of the flight element and convert them to strings to be placed in the HTML table:

```
<TR>
<TD><xsl:value-of select="destination"/></TD>
<TD><xsl:value-of select="airline"/></TD>
<TD><xsl:value-of select="movie"/></TD>
<TD><xsl:value-of select="meal"/></TD>
</TR>
```

Figure 4 shows the results of these XSL commands when applied to the airline flight data file. Note that IE5 will automatically apply XSL commands to an XML file if you place this processing instruction after the XML header:

```
<?xml-stylesheet type="text/xsl"
href="flight1.xsl"?>
```

XSL is a very sophisticated style language. You can learn more about it by reviewing the proposed standard at www.w3.org or reading Microsoft's excellent online user's guide at msdn.microsoft.com/xml.

AN INTRODUCTION TO XML

Mark E. Donaldson

Using XSL for Multiple Views

One application of XML/XSL is the implementation of multiple views. Because the data and the display are separated, instead of coding a complete new display for your data in HTML and having to reenter all of the data, you can simply change the XSL to produce a new display. For example, if you had users who wanted to sort the flight data by the type of aircraft, you might provide an "aircraft view" using the following code:

```
<xsl:for-each select="flight_schedule/flight" order-by="aircraft">
```

When displayed, this shows the aircraft-view table in Figure 5.

What About Netscape?

Most of the examples so far have focused on XML support in Internet Explorer 5.0, but what about XML support in Netscape Navigator? As you probably have heard, the next Netscape browser, currently called Gecko, will be based on the open-source project called Mozilla.org.

The Mozilla.org site has an in-depth discussion of the XML support in Gecko. Here are some of the highlights:

- Mozilla.org uses the free-software XML parser called expat.
- There will be full XML/CSS support.
- There is no mention of XSL, because it is not yet a standard.
- Full DOM support is expected.
- The Gecko user interface will build upon an XML-based language called XUL, short for XML-based user-interface language. More details are available at www.mozilla.org/rdf/doc/xml.html.

The reality of the situation is that although IE5 has an increasing market share, you probably won't be able to count on XML support in your users' browsers until a Netscape browser that supports XML is released. But that doesn't keep you from using XML on the server today.

XML Support on the Server

Microsoft Internet Information Server 5.0 supports all of the XML features we have examined via its Web server. You can use the Internet Server API, Active Server Pages, or CGI scripts to load XML, using the same parser that Microsoft uses for IE5's XML support.

XML on the server lets you load an XML document into memory or generate one using Active Data Objects (ADO). Once you have an XML document on the server, you can manipulate it using the DOM, exchange the document with a business partner, or convert it to HTML for display on a browser using XSL or CSS.

Another option is that you can detect which browser is being used, and if the user's browser is IE5, you can send the raw XML data to the browser for client-side manipulation. For other browsers, you can do pure server-side manipulation. This strategy will prepare you for a future when XML support is more prevalent in browsers.

Putting it All Together

Here's a summary of the benefits of using XML in your Web-based applications:

Revised January 1, 2009

Page 6 of 7

AN INTRODUCTION TO XML

Mark E. Donaldson

- XML lets you exchange data with business partners.
- XML is a great way to load data and manipulate it on either your server or client tier. This makes it easy to coalesce data from various sources on your server.
- Using XML on the browser, you can eliminate round trips for your data. Imagine if our flight data were 2MB and had to be resent every time a user wanted to change the view.
- With the data stored in the browser using XML, we can very easily change the data's display.
- You can use XSL to implement multiple views of any XML data by simply changing the display description. You don't have to reenter the data for every page. Updates to your data show up automatically in your views of the data. If you have done any object-oriented programming or studied patterns, this is the familiar Model-View-Controller paradigm whose benefits are universally recognized.