Using XSL-FO to create printable documents

Portable database reporting using XML revisited

Rodolfo Raya

November 01, 2001

Need portable documents that, unlike most XML documents, include representation information? This article introduces XSL-FO (XML Stylesheet Language-Formatting Objects) and explains how it can come to the rescue. To demonstrate the advantage of using XSL-FO, the article includes an example implementation of a database reporting system that uses Java and XML code.

In this article I'll demonstrate why XML Stylesheet Language-Formatting Objects, known as XSL-FO, is the tool to use when you need to work with XML documents tailored for printing. I'll describe the benefits of including formatting information inside XML documents under particular circumstances.

Formatting Objects basics

XML was designed as a portable means to exchange data between different applications; data presentation is often left to the applications themselves. An XML document typically describes data in an orderly manner, with indentation being the only formatting. In this section I take a look at the use of XSL-FO as a specialized XML vocabulary designed to describe document appearance. I also provide some tips on how to use XSL-FO.

A definition of FO and its role in XSL

As you probably already know, XSL stands for Extensible Stylesheet Language. In defining the components of the language, the W3C's Working Group was clear in specifying that XSL consists of two major parts:

• A language for transforming XML documents
• An XML vocabulary for specifying formatting semantics

The first component of XSL is known as XSL Transformations or XSLT. The second part is called XSL-FO or, simply, Formatting Objects (FO).

XSLT is used to make XML documents readable by transforming tagged data into a nice looking document. The document transformation usually is accomplished by applying the rules defined in a style sheet through the use of an XSLT processor. This method requires one XML document
as source, another document with formatting information (a style sheet), and the processor. But what if you want to use XML but you don't want to deal with two documents? The quick answer is: include formatting information in your documents.

You can use FO to include formatting information in your documents. Although it isn't as popular as XSLT, FO certainly is as important. XSL-FO provides a set of tags that can be used to define how a document will appear to the user. With FO, you can define page layout, font style, colors, image rendering, and many other design properties.

If you take the time to read the 400-plus page XSL-FO specification (see Related topics), you may be surprised by the large number of formatting objects defined by the W3C Working Group. FO is not limited to printed documents; it leaves a door open for multimedia documents. If 400 pages is too long for your reading taste, try the excellent 62-page digest by Elliote Rusty Harold (see Related topics).

It's interesting to note that in the XSL specification, the W3C didn't include an official DTD (Document Type Definition) that you could use to validate an FO document. Luckily, an experimental FO DTD has been produced by RenderX that can help with document validation.

At this point, a couple of hard questions may come to mind: When should XSL-FO be used, and why? I'll explore a few possibilities.

**Some uses of FO**

To start, I would divide XML documents into two categories:

- Pure data documents for information exchange between applications only
- Documents that will be read by a human being

Referring to the first category, XML documents that simply transfer data between two different applications usually don't need any formatting information at all. Computer programs don't care about the look and feel of the data they manipulate. Sometimes even the indentation included inside the documents is considered superfluous.

Documents that belong to the second category usually are transformed using a style sheet before they are read. And documents that are prepared using the same style sheet will look alike. But there are times when you don't want to use a style sheet because you want to give different representations to objects of the same type. This is where FO can help: You can write your documents using FO as a specialized XML vocabulary.

To write documents using the format that you need, you can use one of the many word-processing applications that are available. However, remember that each of them stores documents in proprietary format. So I can write a document using Microsoft Word under Windows, open it using StarOffice in Linux, and read the text, but invariably the format is different from the original. If my document could be saved as an XML document using FO, however, I can expect to retain the formatting information regardless of the application I use. The great advantage to using FO is that you need to work with only one document format in any platform or application.
Word processors and XML

By using XML to store documents, word processors can gain in portability and compatibility.

Abiword is a small, fast, open-source word processor that stores text documents in XML format. It is able to export and import text to and from XSL-FO and DocBook. Abiword transforms documents from its own format into subsets of DocBook or XSL-FO and vice versa. By doing so -- and with a little massage from external programs -- one can obtain different output from a single document source.

Another word processor that uses XML to store documents is StarWriter, a component of StarOffice and OpenOffice. Its DTD is available for download at the OpenOffice Web site (see Related topics).

While FO seems to be an excellent format for word processors to use for storing documents, it isn't the only XML option for document storage. DocBook is a DTD that OASIS promotes as a standard for document writing. I've used it a few times and like the way my work is organized. (See Related topics for a link to the OASIS Web site). The readers of the technical stuff I wrote using DocBook never realized that I used XML for the task. They received printed booklets created using the style sheets provided by Norman Walsh to generate FO files that Fop (Apache's FO Processor) converted to regular PDF files.

Before I go on, I should note that many people believe that including format information inside an XML document inherently goes against the basics of XML. In a sense they are right: XML was devised as a data exchange mechanism free from interfering formats. But, as mentioned above, FO belongs to another technology called XSL that coincidentally uses the XML format. I see no evil in the mix. Consider XSL-FO a specialized vocabulary for layout description -- nothing more, nothing less.

FO by example

In this section I show you how to write a simple FO document. If you plan to master FO, you should learn on your own how to use the 56 different objects that comprise XSL-FO. (Try the link to the Chapter 18 of the XML Bible in Related topics.)

How to write an FO document

In the first article, my goal was to obtain printable reports. I achieved that by using Fop to generate PDF documents that could be read on screen or printed, while keeping the format specified in an XML document.

Figure 1 shows a modified version of the original reporting system diagram.
Where you see `output.fo`, an intermediate XML document had to be translated to FO using a style sheet before it was sent to Fop. Actually, I cheated because when I explained the header of the intermediate document, the following code writes the header of a document that uses FO with a minor change: an FO document always starts with `fo:root` as its root element. The real header should be:

```xml
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
```

An FO file is made of:

1. An XML header and name space declaration
2. Page layout information
3. Page header and footer content
4. Text content
5. Closing tags -- never forget them!

I know myself and I’m sure that the task of writing the XSL style sheet to transform the intermediate document would be a never-ending story. The plans for MXReports include lots of future updates and that means lots of additional changes to the programs and the style sheet. I’ll try to deal with the program only by relying on Fop to transform my intermediate FO documents.

Now that you know how the header should look, let’s work on how to describe the page layout. In the sample program that I included in my first article, I showed you how to retrieve from report.xml the following data:

- `pageHeight`
• pageWidth
• headerHeight
• footerHeight

The example in Listing 1 uses those variables in the following code snippet to tell Fop the page layout wanted for the document.

When you execute Listing 1, the output looks like Listing 2.

**Listing 1. Describing the desired page layout for the FO processor**

```java
try {
    output.write("<?xml version="1.0" encoding="utf-8"?>\n");
    output.write("<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">\n");
    output.write(" <fo:layout-master-set>\n");
    output.write("  <fo:simple-page-master master-name="sample" 
    page-height="" + pageHeight + "mm"\n";
    output.write("    page-width="" + pageWidth + "mm"\n";
    output.write("  margin-left="25mm"\n";
    output.write("  margin-right="25mm"\n";
    output.write("  margin-top="" + headerHeight + "mm"\n";
    output.write("  margin-bottom="" + footerHeight + "mm"\n";
    output.write("  <fo:region-body margin-top="0mm"\n";
    output.write("      margin-bottom="0mm" />\n");
    output.write("  <fo:region-before extent="3cm" />\n");
    output.write("  <fo:region-after extent="1.5cm" />\n");
    output.write("  <fo:simple-page-master>\n");
    output.write("  </fo:layout-master-set>\n");
    output.write("  <fo:page-sequence master-name="sample">\n");
    output.write("    <fo:flow flow-name="xsl-region-body">\n";
}
```  

**Listing 2. The output from running Listing 1**

The text in Listing 2 defines that I want a standard A4 page size (210mm x 297mm) with the specified margins using the data extracted from the XML report definition. A page has five zones, or regions: region-body, region-before, region-after, region-start, and region-end. Figure 2 shows the placement of each region in a page.
Figure 2. Page layout and region placements

Inside the zip file that accompanies this article (see Related topics) you will find mxreports.java, which includes all of the sample code. Download the sample code and take a look at the methods writeHeader() and writeDetails().

In writeHeader() you will see how to describe page layout and how to define header and footer content as static objects.

The method writeDetails() defines the content of the report as a table in an HTML-like style. I used a table in this application because the detail band contains a fixed number of elements that should be printed once for each row in the dataset. In other word, the report consists of a table with as many rows as records in the dataset and as many columns as elements defined in the detail band. But keep in mind that a regular FO file usually is made of blocks and inline objects.

The sample in Listing 3 illustrates how to define a table with two columns using FO.

Listing 3. Using FO to define a table with two columns

```xml
<fo:table>
  <fo:table-column column-width = "90mm" />
  <fo:table-column column-width = "90mm" />
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell>
```
Note: Specify column width explicitly if you plan to use Fop as described in this article.

If you’ve ever written a table using HTML, it will be easy for you to understand the basics of a table object. Take a look at Listing 3.

Table 1. Comparing equivalent FO and HTML table tags

<table>
<thead>
<tr>
<th>FO tag</th>
<th>HTML tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>fo:table-body</td>
<td>table</td>
</tr>
<tr>
<td>fo:table-row</td>
<td>tr</td>
</tr>
<tr>
<td>fo:table-cell</td>
<td>td</td>
</tr>
</tbody>
</table>

How to process FO files and make them useful

By now you should at least have an understanding of what an FO file is and how to generate one. What you might also want to know is how to make that file readable and tag free.

For the moment, there are only a few alternatives available to making XSL-FO files useful. The best options for tranforming FO files into readable format are:

- Portable Document Format (PDF)
- Rich Text Format (RTF)
- Web pages (HTML)

You can generate PDF files using Fop, a free open-source tool developed as an Apache XML project, or by using XEP, a commercial product from RenderX (see Related topics).

Browsing Fop-related newsgroups, I found StandaloneConverter, an experimental standalone FO-to-RTF converter that was developed by Bertrand Delacretaz. I was able to compile and run the program, and you will find an RTF version of currency.fo in the zip file. The RTF file generated by this tool doesn't keep the margins that are specified in currency.fo, but it provides an excellent means to deploy a document that can be edited.

As stated earlier in this article, an FO file is a regular XML file, and you can use any XSLT tool, such as Xalan, to convert from FO to HTML.
FO present and future

With XML's increasing popularity, I believe that within a couple of years XML will be established as the standard format for storing and distributing documents. Many software companies are updating their products to support XML. Most of them use proprietary XML formats to store documents, but some of these companies have started publishing the DTDs they use. DTD publishing is a direct invitation to use XSLT to achieve compatibility among heterogeneous systems.

Adobe's Portable Document Format (PDF) standard has worldwide acceptance, and using Fop to produce PDF documents is becoming very popular. Fop works with XSL-FO documents only. As more people use it, Fop will likely generate an increase in the use of FO as a specialized vocabulary for presentation description.

I don't think XSL-FO will ever be used as an established standard for document writing, but the generalized use of other XML vocabularies (like DocBook) and the high availability of Java XSLT tools will increase the use of FO as an auxiliary technology.

Conclusion

In this article I described XSL-FO, the specialized XML vocabulary that can be used to create XML documents with formatting information included. I touched on word processors and why it is important that XML be used to store documents. I detailed a method for generating an FO file using Java. Finally, I looked at options for transforming FO file.

This article is not intended as a thorough XSL-FO tutorial. I've used only a few formatting objects in the sample code, and I showed only a small portion of their properties. In the Resources section, you will find pointers to good tutorials on FO. It doesn't take too much to learn the basics of XSL-FO. In fact, I spent more time searching the Web for information than in developing an understanding of the basics of the subject!
## Downloadable resources

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x-xslfo/xmlreports.zip</td>
<td></td>
</tr>
</tbody>
</table>
Related topics

- I invite you to make use of the article's zip file, which includes a program that will help you learn how to generate a PDF file using Fop.
- Throughout this article there are lots of references to the Apache XML Project's FO Processor, Fop. Find out more about Fop, download the code, and access related resources at the Apache Fop home page. Apache devotes a page to Fop resources. Several references to XSL-FO are on the Fop page, but beware that the link to the XML Bible points to an outdated version of the book.
- Immerse yourself in Formatting Objects by jumping into the W3C's XSL-FO specification, which runs to more than 400 pages.
- Chapter 18 of the XML Bible (in the second edition) by Elliotte Rusty Harold, provides an excellent introduction to XSL-FO. Get the book, read it online, or at least print out Chapter 18 as an FO reference.
- Download Abiword, an open-source word processor that uses XML, from the Abiword page.
- StarWriter, a component of StarOffice and OpenOffice, is another word processor that uses XML to store documents. Download its DTD from the OpenOffice Web site.
- OASIS promotes DocBook, a DTD, as a standard for document writing.
- RenderX sells XED, a commercial tool capable of transforming XSL-FO files into PDF format.
- Norman Walsh created XSL and DSSSL style sheets for DocBook that you could use as templates for your own XSL-FO transformations. Find them on the sourceforge DocBook page.
- The StandaloneConverter program, a Java tool that converts FO to RTF has evolved into an open-source project, jfor, which is hosted at sourceforge. Find out more about jfor at the jfor home page.
- Try out XSLFast, a product I found in jCatalog that the suppliers call "the world's first graphical editor for XSL-FO documents."
- By the way, I wrote this article in XML format using VIM 6.0 (the latest version of an old editor), parsed the text calling Xerces with a one-line batch file, and generated HTML with another one-line batch file using Xalan.