XML to PDF using XSL-FO

Author: Darshan Singh (Managing Editor, PerfectXML.com)
Last Updated: March 30, 2003

In this article, you'll learn about XSL-FO and an example to convert XML into a PDF document. This article assumes that you have some familiarity with XML and XSLT.

Contents

Introduction

It is a fact that XSLT has become an important technology for almost all XML developers. I think, some of the reasons behind XSLT's success include ease of learning, great tools/vendors support, and its ability of allow easily turning XML into HTML or to some other XML (or text) forms. In addition to this, some other features that made XSLT popular includes: the modularity features (create re-usable templates), conditional processing features, ability to write XSLT extensions (script-, .NET-, or Java-based), sorting, looping, numbering, ability to pass parameters and work with variables, and so on. Today, XSLT is widely used to *transform* XML-based data/content into HTML to be displayed inside the Web browsers. In summary, XSLT offers the perfect solution to transform XML into HTML or any other text format for that matter. However, if the requirement is to generate production-quality print documents from the XML content, just XSLT is not sufficient.

In day-to-day life, we often make use of printed material – textbooks, manuals, contracts, catalogs, newspaper, magazines, and brochures. Producing high-quality print documents requires ability to do pagination, generating table of contents and index, headers and footers, margins, multi-column output, odd/even page masters, footnotes, endnotes, floats, fonts, leading, word spacing, highlighting and so on.

If the content is stored in XML format, and you need to compose high-quality print and online pages, that has above mentioned features (such as pagination), you can make use of another W3C specification, known as, **XSL Formatting Objects** or **XSL-FO** (also known as XSLFO or just XSL).

XSL-FO: Past and Present

XSL Working Group at W3C was created around January 1998 and on 18th August, 1998 this group **published** the first working draft of the Extensible Stylesheet Language (XSL), a XML-based language to express stylesheets.

XSL was created with two main goals: first, to allow transforming XML documents, and second, to specify the formatting semantics to produce formatted output on a display, on paper, or in speech, or onto other media.

After producing six subsequent XSL working drafts, the specification turned to candidate recommendation status in November 2000. By this time, the original XSL specification was divided into three specifications:

XSLT (or XSL Transformation), **XPath** (XML Path Language), and **XSL-FO**. The popularity of XSLT and XPath resulted in fast progression of these two specifications, both receiving the W3C Recommendation status in November 1999. W3C is working on version 2.0 of XSLT and XPath specifications – already four working drafts have been produced and sometime soon they'll receive the candidate recommendation status.

XSL-FO on the other hand progressed slowly and finally on 15 October 2001, W3C decided to turn XSL-FO to **Recommendation** status. Since then, various vendors have announced the support for XSL-FO, and it is being used more and more for XML-based printing solutions.

XSL-FO 1.0 (http://www.w3.org/TR/xsl/) specification is probably the longest (about 400 pages) document on the W3C site. Most of the space in the XSL specification is occupied by the large number of formatting objects (for both printed documents as well as multimedia documents) that the spec defines.

Generating PDF from XML

Today, the most common application of XSL-FO is to produce Adobe PDF documents from the source XML files. Here is the two step process that is used to turn XML into PDF:

Step 1. XML to XSL-FO using XSLT:

The first step is to *transform* XML into XSL-FO format document. XSL-FO, like XSLT, makes use of XML syntax. The obvious choice here is to use XSLT to transform XML into XSL-FO. However, if you want, you can use some other method, such as DOM- or SAX-based processing the XML document, and generating XSL-FO document. In this article, we'll use XSLT to transform XML into XSL-FO. XSLT transformation engine, such as MSXML, Xalan, Saxon, XT, etc. can be used to apply the XSLT stylesheet on the XML document.

XSL-FO document describes the details of the presentation, such as physical size of the page, pagination details, margins, fonts, font sizes, colors, and so on. These characteristics are expressed using XSL **formatting objects** (for example: `fo:page-sequence`, `fo:block`, `fo:footnote`, `fo:float`, `fo:region-body`, and so on) and **formatting properties** (for example: `background-attachment`, `background-color`, `font-family`, `text-depth`, and so on). If you are know CSS (Cascading Style Sheets), the XSL formatting properties should sound familiar, however it is important to realize that XSL-FO provides a more sophisticated layout model than the CSS, and that XSL-FO is specifically designed to be used for generating fine-grained presentational layout files.

Step 2. Processing XSL-FO using Formatting Engine (or Formatter):

Once you have the XSL-FO document, XSL rendering engine, such as FOP or XSL Formatter (see **XSL-FO Tools** section below) can be used to convert XSL-FO elements into a PDF, PostScript, RTF or any other such print format.

XSL-FO is not meant to be hand-coded. Generally, XSLT stylesheet will be written to transform XML into XSL-FO. The rendering engine is then used to convert XSL-FO into the required print documents. External resources (such as images and fonts) can be referred in the XSL-FO document. SVG elements can be used inside XSL-FO document to produce vector graphics (such as charts or maps).

Before we look at how the XSL-FO document looks like, and an example of producing PDF from XML, let's review some XSL-FO Tools.

XSL-FO Tools

Commercial as well as open source (free) XSL-FO implementations are evolving rapidly. In this section, you'll learn about some of the popular XSL-FO tools.

- **XSL Formatter** by Antenna House, Inc.
  Commercial product; Runs on Microsoft Windows (Unix/Linux version to be made available soon), consists of an XSL-FO engine and a user interface. XSL Formatter offers programming and GUI-based formatting solution that conforms to XSL V1.0 W3C Recommendation. The XSL Formatter IDE is well designed. I personally use it a lot to immediately see the results by providing it either the XML+XSLT combination (the IDE then uses MSXML or other specified XSLT processor to produce the XSL-FO) or by directly specifying the XSL-FO file. XSL Formatter SDK can be used to add the XSL-FO support in your applications. **Click here** for more information on XSL Formatter.

- **Formatting Objects Processor** or **FOP** by The Apache Software Foundation.
  FOP is an open source, Java application, XSL-FO rendering engine that can produce various other formats of output in addition to PDF, such as PCL, PS, SVG, XML, etc. The current version 0.20.5 supports the XSL-FO Version 1.0 W3C Recommendation. If you are looking for a free, Java-based (and hence works on many platforms) XSL-FO solution, FOP is your best choice. **Click here** to learn more about FOP.

- **XEP Rendering Engine** by RenderX
  Commercial product, written in Java, COM wrapper provided for Windows application integration, implements XSL-FO 1.0 W3C recommendation and allows converting XML documents into PDF or PostScript. **Click here** to learn more about XEP.

- **Formatting Objects Authoring** or **FOA** by Fabio Giannetti (HP)
  FOA is an open source Java application, designed to be used as a XSL-FO **authoring** tool – a graphical interface designed to aid in creation of XSL-FO files. **Click here** to learn more about FOA. Also see Word HTML 2 Formatting Objects (or WH2FO).

XSL-FO Document Structure

As alluded to previously, XSL-FO document, like XSLT, uses the XML syntax. XSL-FO elements belong to the namespace **http://www.w3.org/1999/XSL/Format** and generally **fo:** namespace prefix is used for the XSL-FO namespace URI. Note that XSL 1.0 turned to recommendation status in October 2001; the 1999 in the namespace URI indicates the year in which the URI was allocated by the W3C. It does not indicate the version of XSL being used.

XSL-FO document beings with a node named **fo:root**. This node contains a single **fo:layout-master-set**, an optional **fo:declarations**, and a sequence of one or more **fo:page-sequences**. In brief, the `fo:layout-master-set` tag defines page layouts (the geometry and sequencing of the pages) and the children of the `fo:page-sequences`, which are called flows (contained in `fo:flow` and `fo:static-content` tags), includes the actual content to be formatted.

```
fo:root := (layout-master-set, declarations?, page-sequence+)

fo:layout-master-set := (simple-page-master | page-sequence-master)+

fo:declarations := (color-profile)+

fo:page-sequence := (title?, static-content*, flow)
```

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="abc">
        <fo:simple-page-master master-name="pqr">
        ...
    </fo:layout-master-set>
    <fo:declarations>...</fo:declarations>
    <fo:page-sequence master-reference="abc">
        <fo:title>... </fo:title>
        <fo:static-content ... >...</fo:static-content>
        <fo:flow ...>...</fo:flow>
    </fo:page-sequence>
    <fo:page-sequence master-reference="pqr">
        <fo:title>... </fo:title>
        <fo:static-content ... >...</fo:static-content>
        <fo:flow ...>...</fo:flow>
    </fo:page-sequence>
    ...
</fo:root>
```

Figure 1. XSL-FO Document Structure

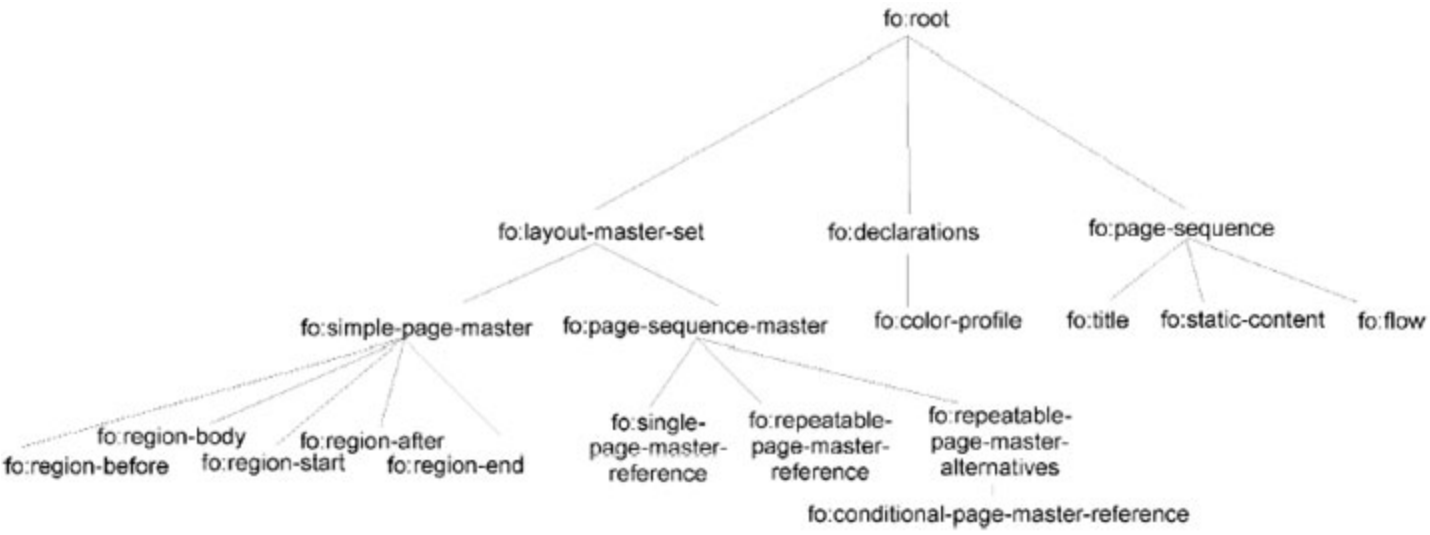

Figure 2. Tree Representation of the Formatting Objects for Pagination (Source: W3C Web site)

Complete details on XSL-FO document structure can be found in the **XSL specification**.

Formatting Objects and Formatting Properties

In simple words, the tags defined by the XSL-FO specification can be termed as Formatting Objects and attributes on those tags as Formatting Properties.

There are about 56 Formatting Objects categorized in eight categories:

1. **Declaration, Pagination, and Layout Formatting Objects**
   Includes tags such as `fo:root`, `fo:page-sequence`, `fo:layout-master-set`, `fo:region-body`, `fo:flow`, and so on. **Click here** for more details.

2. **Block Formatting Objects**
   Includes two tags `fo:block` and `fo:block-container`. **Click here** for more details.

3. **Inline Formatting Objects**
   Includes tags such as `fo:external-graphic`, `fo:leader`, `fo:page-number`, and so on. **Click here** for more details.

4. **Table Formatting Objects**
   Includes tags such as `fo:table`, `fo:table-column`, `fo:table-header`, `fo:table-footer`, `fo:table-body`, `fo:table-row`, `fo:table-cell` , and so on. **Click here** for more details.

5. **List Formatting Objects**
   Includes tags such as `fo:list-block`, `fo:list-item`, `fo:list-item-body`, and `fo:list-item-label`. **Click here** for more details.

6. **Link and Multi Formatting Objects**
   Includes tags such as `fo:basic-link`, `fo:multi-case`, and so on. **Click here** for more details.

7. **Out-of-line Formatting Objects**
   Includes tags `fo:float`, `fo:footnote`, and `fo:footnote-body`. **Click here** for more details.

8. **Other Miscellaneous Formatting Objects**
   Includes tags `fo:wrapper`, `fo:marker`, and `fo:retrieve-marker`. **Click here** for more details.

XSL-FO specification copies and inherits lots of formatting properties from the CSS2 specification. More than 100 properties are defined, and grouped under various categories such as Accessibility Properties, Absolute Position Properties, Border, Padding, and Background Properties, Font Properties, Relative Position Properties, Alignment Properties, Layout-related Properties, and so on. **Click here** for complete details on XSL-FO Formatting **Properties**.

"Hello XSL-FO"

I hope by now, you have a good understanding of what XSL-FO and what it can be used for, and how an XSL-FO document would look like. Before we look at a detailed XML-to-PDF example that makes use of both, XSLT and XSL-FO, let's hand-code a simple "Hello XSL-FO" .fo document and convert that into a PDF document.

You'll need XSL-FO formatter to process the following document to generate the PDF output. Download **XSL Formatter** (evaluation version), **FOP**, or **XEP** (evaluation version), and we'll show you how to run each of this to turn XSL-FO input into PDF output.

c:\hello.fo

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

   <fo:layout-master-set>

      <fo:simple-page-master master-name="all-pages"
            page-width="8.5in" page-height="11in"
```

```
            margin-top="20pt" margin-bottom="20pt"
            margin-left="50pt" margin-right="50pt">

            <fo:region-body region-name="xsl-region-body"
               margin-bottom="25pt" margin-top="10pt"
               border="2.25pt solid gray" padding="16pt"/>

      </fo:simple-page-master>

   </fo:layout-master-set>

   <fo:page-sequence master-reference="all-pages">
       <fo:flow flow-name="xsl-region-body">

          <fo:block text-align="center">
              <fo:external-graphic
                src="http://www.PerfectXML.com/images/PerfectXML_button.jpg"/>
          </fo:block>

          <fo:block text-align="center">
              <fo:leader leader-pattern="dots" leader-length="360pt"/>
          </fo:block>

          <fo:block font-family="sans-serif"
                font-size="17pt" line-height="20pt"
                text-align="center">
                Hello
             <fo:inline font-weight="bold" font-style="italic">XSL-FO</fo:inline>
          </fo:block>

       </fo:flow>
   </fo:page-sequence>

</fo:root>
```

As per the XSL-FO specification, the c:\hello.fo sample document starts with the element named `root` and uses the namespace **http://www.w3.org/1999/XSL/Format**.

The `layout-master-set` tag is used to define page layout attributes. The above sample contains just one `simple-page-master`, as we want all (just one page in this example) to have the same layout. However, let's say if you wanted the very first page to look different, and then odd and even pages to have different layouts, in such cases, you'll have to then define in total three `simple-page-master` tags.

The layout of the page and the actual content on that page is linked via the **master-name** attribute on the `simple-page-master` and the **master-reference** attribute on the `page-sequence` tag.

Notice that we have used various measuring units in the property values, such as *pt* and *in* in the above sample FO document. The measuring units supported by XSL-FO include: cm (for centimeters), mm (for millimeters), in (for inches), pt (for points; 1/72 of an inch), pc (for picas; 1/6 of an inch), px (for pixels), and em (for the width of uppercase M).

XSL-FO specification allows dividing the page into up to five regions: **region-body** (main area in the center of the page), **region-before** (header), **region-after** (footer), **region-start** (left-sidebar), and **region-end** (right-sidebar). The above sample XSL-FO document defines the attributes for just region-body using the `fo:region-body` tag. So far, we have just described the layout of the page - the actual content of the page is enclosed in the `fo:flow` tag within `fo:page-sequence`. The `fo:flow` tag in the above example contains three `fo:block` children nodes. The `fo:block` is the most basic formatting element, which can be thought of as synonymous to <p> tag in HTML.

The first block contains a reference to external image, second block prints a dotted line, and third line prints text "Hello XSL-FO" (with XSL-FO printed in bold and italics).

Assuming that you have above sample XSL-FO text saved as c:\hello.fo, here is how you can get it ready for printing:

- **Using XEP**
  XEP includes a batch file that you can use (and pass it -format and -out parameters) or you can use the following command line instructions to process a .fo document and generate a PDF file (assuming that XEP is installed under C:\XEP):

```
set CP="C:\XEP\lib\xep33_trial.jar;C:\XEP\lib\cryptix32.jar;C:\XEP\lib\cryptix32-
pgp.jar;C:\XEP\lib\saxon.jar;C:\XEP\lib\xt.jar"

java -classpath %CP% -Dcom.renderx.xep.ROOT="C:\XEP" com.renderx.xep.XSLDriver
    -format c:\hello.fo -out c:\hello.pdf
```

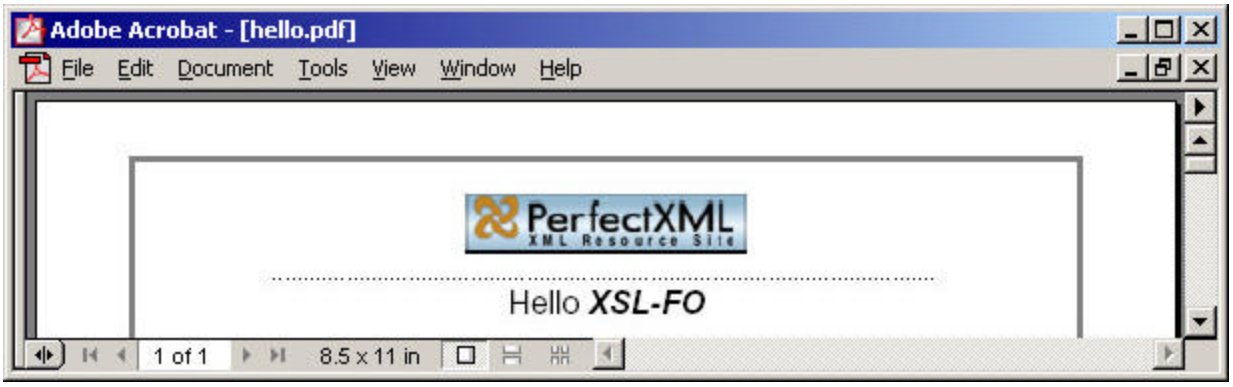

Figure 3. Hello.PDF - generated using XEP Rendering Engine by RenderX

- **Using XSL Formatter**
  Start the XSL Formatter IDE, open c:\hello.fo document and click on the Run Formatter toolbar button:

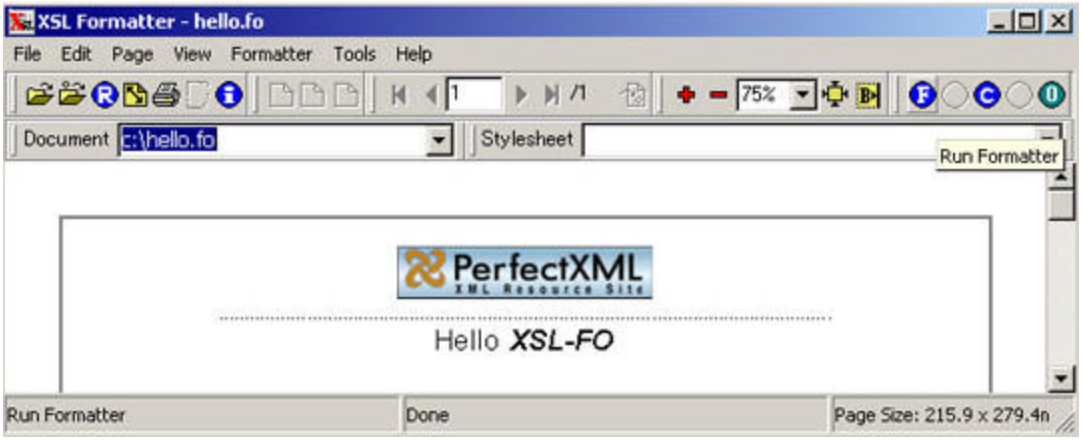

Figure 4. Hello.fo - as formatted by XSL Formatter (Antenna House, Inc.)

- **Using FOP**
  FOP also includes a batch file (fop.bat) that you can use. This batch file sets the `LIBDIR` and `LOCALCLASSPATH` environment variables, followed by the instruction:

```
java -cp %LOCALCLASSPATH% org.apache.fop.apps.Fop %1 %2 %3 %4 %5 %6 %7 %8
```

Using fop.bat batch file:

```
fop c:\hello.fo c:\hello.pdf
```

Running FOP directly: (assuming classpath is set correctly)

```
java org.apache.fop.apps.Fop c:\hello.fo c:\hello.pdf
```

Refer to individual tool documentation for more details on other command-line parameters and features.

XML + XSLT => XSL-FO => Formatter => PDF

Let's sum up this article with an example. In this section:

- First, we'll present the source XML document.
- Then, the XSLT stylesheet used for transforming the above source XML document into XSL-FO document.
- Finally, we'll use the XSL Formatter IDE to look at the generated print document.

Source XML Document (promote.xml)

The XML document simply contains some text divided into two paragraphs (tags).

```
<?xml version="1.0" encoding="UTF-8"?>
<Media>
   <para>
      Imagine more than 6000 customers going to a store and seeing your ad
signboards!
      That's what happens on PerfectXML. About 6000 unique visitors come to
PerfectXML.com
      daily to get answers to their XML/Web services questions, to learn something
new,
      to get the latest news, to read book chapters, to find out about
      XML/Web services tools and organizations, and more!
   </para>

   <para>
      Promote yourself or your company by contributing content to the PerfectXML
community site.
   </para>
</Media>
```

XSLT to Transform XML to XSL-FO (promote.xsl)

The following stylesheet is written to transform the above XML document into XSL-FO document. The stylesheet uses msxsl:script extension to define a function used for getting the current system date. The stylesheet contains two templates - the root (/) matching template contains the XSL-FO `xsl:fo` element and hence defines the beginning of the XSL-FO document. Next, page layout attributes are defined, followed by `fo:page-sequence` tag that is responsible for actual page content. The `fo:flow` tag contains `xsl:apply-templates` that leads to calling template matching para nodes. For each such node, we print first letter differently than the rest of the text, followed by a dotted blue line.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xsl:stylesheet
     version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns:fo="http://www.w3.org/1999/XSL/Format"
     xmlns:msxsl="urn:schemas-microsoft-com:xslt"
     xmlns:scriptExt="urn:PXML-Utilities">

<xsl:output method="xml" version="1.0" encoding="UTF-8" />

<msxsl:script language="JScript" implements-prefix="scriptExt">
     function todaysDate()
     {
        var DateObj = new Date();
        return DateObj.getMonth()+1 + "/" + DateObj.getDate() + "/" +
DateObj.getFullYear();
     }
</msxsl:script>

<xsl:template match="/">
  <fo:root>
    <fo:layout-master-set>
      <fo:simple-page-master master-name="allPages">
        <fo:region-body border="medium ridge silver"
            margin="0.2in" padding="0.1in"/>
      </fo:simple-page-master>
    </fo:layout-master-set>

    <fo:page-sequence master-reference="allPages">
       <fo:flow flow-name="xsl-region-body">
          <xsl:apply-templates />
          <fo:block color="gray"
             font-family="monospace" text-align="right" font-size="10pt" line-
height="12pt">
               Printed on: <xsl:value-of select="scriptExt:todaysDate()" />
          </fo:block>
       </fo:flow>
    </fo:page-sequence>
  </fo:root>
</xsl:template>

<xsl:template match="para">
          <fo:block
             font-family="Tahoma" text-align="left" font-size="10pt" >
               <fo:inline font-family="Tahoma" text-align="left" font-size="20pt"
                    line-height="22pt">
                  <xsl:value-of select="substring(normalize-space(.), 1, 1)" />
               </fo:inline>

               <xsl:value-of select="substring(normalize-space(.), 2)" />
          </fo:block>
          <fo:block text-align="left">
              <fo:leader color="blue" leader-pattern="dots" leader-
length="550pt"/>
          </fo:block>
</xsl:template>

</xsl:stylesheet>
```

Formatting using XSL Formatter

After saving the above XML and XSLT stylesheet, start XSL Formatter, type the name and location of the XML document in the Document select box, name and location of XSLT file in the Stylesheet combo box. As the above stylesheet makes use of MSXML supported script extension, make sure XSL Formatter is using MSXML (Formatter | Formatting Options Shift+Ctrl+O) | XSLT Processor tab) and then click on Run Formatter toolbar button or press F5, you should see the output similar to one shown in the following screenshot:

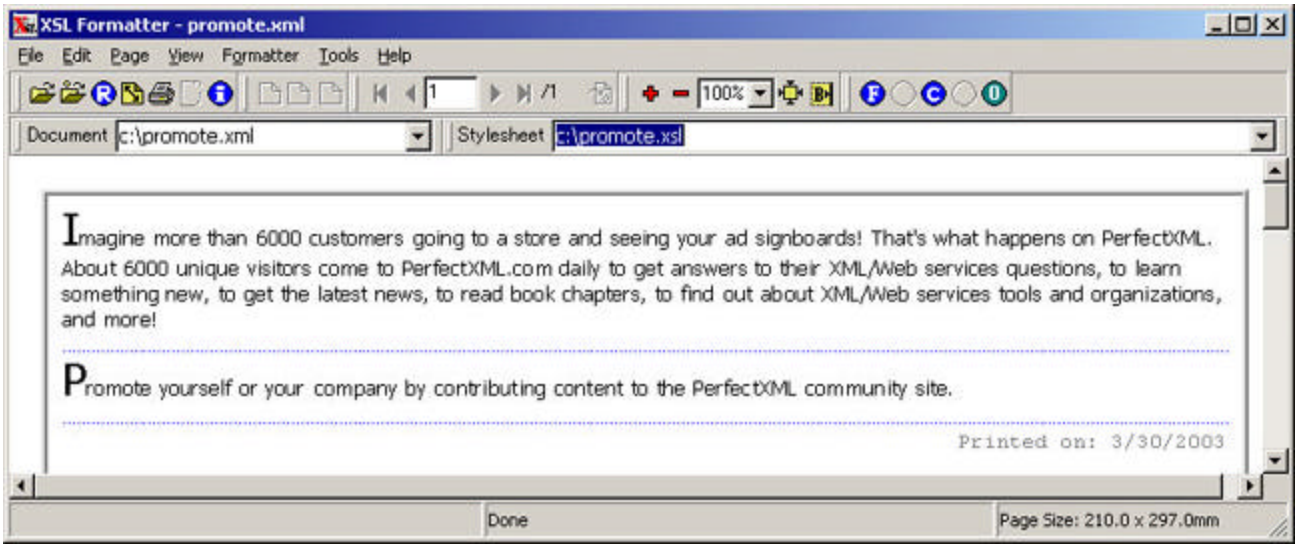

Figure 5. XSL Formatter applies the XSLT stylesheet and shows the formatted XSL-FO output

Next Steps: Things that you can try out

Before concluding this article, I am going to list few things that you can try out or research on your own:

- **Try:** Remove the MSXML-specific text from the above XSLT stylesheet, including the date printing logic, and then try processing the XML+XSLT using XEP and FOP, and generate a PDF document.
- **Try:** Experiment the Antenna House XSL Formatter COM Object (AXFOSVR.dll, AXFOSVR.XFOObj.2) - in a Visual Basic application, in an ASP page, and in a .NET application (using COM interop).
- **Research:** Find out about the limitations of XSL-FO.
- **Try:** Use JAXP and `org.apache.fop` package within a Java Servlet based application to convert XML into PDF.

Resources

- **XSL Info on W3C Web site**
  www.w3.org/Style/XSL/
- **XSL 1.0 Specification**
  www.w3.org/TR/xsl
- **XSL FAQ**
  www.antennahouse.com/XSL20/XSLQA.HTM

- **XSL-FO Tutorial**
  www.renderx.com/tutorial.html

- **What Is XSL-FO and When Should I Use It?**
  www.seyboldreports.com/TSR/free/0217/techwatch.html

- **Formatting Object Processor Web Service**
  www.capescience.com/webservices/fop/

- **XSL-FO Q&A**
  www.dpawson.co.uk/xsl/sect3/

- **Chapter 18 of the XML Bible, Second Edition : XSL Formatting Objects**
  www.ibiblio.org/xml/books/bible2/chapters/ch18.html

- **XSL-FO Discussions**
  groups.yahoo.com/group/XSL-FO

- **XSL-List -- Open Forum on XSL**
  www.mulberrytech.com/xsl/xsl-list/

- **XSL-FO Tools Listing**
  www.PerfectXML.com/Soft.asp?cat=17