XSL-FO Input

Basic Help for Using XML, XSLT, and XSL-FO

$Revision: 490012 $

Table of contents

1. Overview

FOP uses XSL-FO as input. It is the responsibility of the user to make sure that the XSL-FO submitted to FOP is correct. The tutorial items presented here are not comprehensive, but are of the FAQ variety. Another good FAQ is Dave Pawson's XSL FAQ.

2. XML Issues

2.1. Special Characters

When entering special (non-ASCII) characters in XML, the general rule is to use the applicable Unicode character instead of trying to use a character entity as you would with HTML. Remember that HTML is an SGML document type. SGML has a limited character set, which requires it to use character entities to represent special characters. One of the improvements of XML over SGML (and thus HTML) is native support for Unicode. Basic XML has only a handful of character entities, primarily because it doesn't really need more.

Entities such as `&uuml;` (u with an umlaut), which work in HTML, will be flagged as undefined entities unless you define them yourself in your DTD. Use the corresponding Unicode character instead. A list of predefined HTML entities and their Unicode codepoints can be found at Character entity references in HTML 4.

One common example is ` `, used to obtain a non-breaking space in HTML. In XML, use   instead.

For other non-ASCII characters, such as the Euro symbol, checkbox, etc., see the Unicode Reference By Name document that is found at the Unicode Consortium site.

After finding the correct Unicode codepoint to represent the character, use XML Character References to put the character into your source XML, XSLT or FO. See the non-breaking-space comments above for an example of the syntax using decimal notation. The following hexadecimal example will result in a Euro sign:

```
&#x20AC;
```

Getting your XML correctly encoded is only part of the job. If you want the character to display or print correctly (and you probably do), then the selected font must contain the necessary glyph. Because of differences between font encoding methods, and limitations in some font technologies, this can be a troublesome issue, especially for symbol characters. The FOP example file Base-14 Font Character Mapping is a very useful resource in sorting these issues out for the Base-14 fonts. For other fonts, use font editing sofware or operating system utilities

(such as the Character Map in most Windows platforms) to determine what characters the font supports.

An alternative to encoding the character and making it available through a font is to use an embedded graphic to represent the character: GIF, PNG, SVG, etc.

2.2. Entity Characters

The handful of basic XML character entities that do exist are the ampersand, apostrophe, less-than, greater-than, and single-quote characters. These are needed to distinguish markup tags from content, and to distinguish character entities from content. To avoid parser complaints about illegal characters and entities in your input, ensure that ampersands in text and attributes are written as &, "<" is written as <, and ">" as >. It is not necessary everywhere, but it is wise to do so anyway, just to be sure.

Most XML parsers will provide a line number and sometimes a column number for offending characters.

Review the XML Specification or a good tutorial for details of the XML file format.

2.3. Encoding Issues

If the parser complains about illegal bytes or characters in the input, or there are unexpected characters in the output, this is usually is the result of a character encoding problem. See the XSL FAQ for additional information. Many software packages that produce XML, including XSLT processors, use UTF-8 encoding as a default. If you view their output with something not aware of the encoding, like Notepad for Win95/98/ME/NT, funny characters are displayed. A Å is a giveaway.

3. XSLT Issues

3.1. Current Date and Time

XSL-FO does not currently have a function for retrieving the current date and time. However, in some cases, XSLT can be used to place the current date and time into the XSL-FO document as it is generated.

One possibility is to use the exslt date and time extension.

Another possibility is to use java or javascript (or perhaps some other language). Here is an

example, using java, that works with Xalan. First, create the appropriate namespace:

```
<xsl:stylesheet version="1.0"
  ...
  xmlns:java="http://xml.apache.org/xslt/java" exclude-result-prefixes="java"
  ...
```

Next, use the java language to retrieve and format the current date and time. Here is an example template:

```
<xsl:template match="TodaysDate">
    <xsl:value-of select="java:format(java:java.text.SimpleDateFormat.new
('MMMM d, yyyy, h:mm:ss a (zz)'), java:java.util.Date.new())"/>
  </xsl:template>
```

4. XSL-FO Issues

4.1. Vertical Centering

To vertically center an image, table, or other item, use display-align="center". See display-align Compliance for implementation status. Here is a small, self-contained document centering an image on a page:

```
<?xml version="1.0"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="content"
        page-width="210mm" page-height="297mm">
      <fo:region-body/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="content">
    <fo:flow flow-name="xsl-region-body">
      <fo:table table-layout="fixed" width="100%">
        <fo:table-column column-width="proportional-column-width(1)"/>
        <fo:table-body>
          <fo:table-row height="297mm">
            <fo:table-cell display-align="center">
              <fo:block text-align="center">
                <fo:external-graphic src="fop.jpg"/>
              </fo:block>
            </fo:table-cell>
          </fo:table-row>
        </fo:table-body>
      </fo:table>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

4.2. Horizontal Centering (Tables)

To center a table horizontally, one possibility is to add one column on the left and one on the right which pad the table so that the visible part is centered:

```
<?xml version="1.0"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="content"
      page-width="210mm" page-height="297mm">
      <fo:region-body/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="content">
    <fo:flow flow-name="xsl-region-body">
      <fo:table table-layout="fixed" width="100%">
        <fo:table-column column-width="proportional-column-width(1)"/>
        <fo:table-column column-width="100mm"/>
        <fo:table-column column-width="proportional-column-width(1)"/>
        <fo:table-body>
          <fo:table-row>
            <fo:table-cell column-number="2">
              <fo:block>foo</fo:block>
            </fo:table-cell>
          </fo:table-row>
        </fo:table-body>
      </fo:table>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

If your table is more complicated, or if defining borders on individual cells becomes too much work, use the code above and nest your table within the middle cell.

4.3. Right-Aligning (Tables)

To right-align a table, you can use the same approach as above for centering tables. Just remove the last table-column element which causes all the left-over space not used by the columns with a fixed column-width to be assigned to the first column which effectively right-aligns the table.

4.4. Recto/Verso Static Content Differences

One frequent request is that static content be different between recto pages (right-side or odd-numbered pages typically) and verso pages(left-side or even-numbered pages typically). For example, you may wish to place the page number on the "outer" side of each page. There are examples in the FO distribution and in the XSL FAQ FO section.

First, define a page master with alternating pages masters for odd and even pages. Then specify appropriate regions in these page masters, giving them different names. Use these names to put

different static content in these regions. Here is a self-contained demonstration:

```
<?xml version="1.0"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="even"
      page-height="297mm" page-width="210mm"
      margin-top="20mm" margin-bottom="20mm"
      margin-left="25mm" margin-right="25mm">
      <fo:region-body margin-bottom="20mm"/>
      <fo:region-after region-name="footer-even" extent="20mm"/>
    </fo:simple-page-master>
    <fo:simple-page-master master-name="odd"
      page-height="297mm" page-width="210mm"
      margin-top="20mm" margin-bottom="20mm"
      margin-left="25mm" margin-right="25mm">
      <fo:region-body margin-bottom="20mm"/>
      <fo:region-after region-name="footer-odd" extent="20mm"/>
    </fo:simple-page-master>
    <fo:page-sequence-master master-name="document">
      <fo:repeatable-page-master-alternatives>
        <fo:conditional-page-master-reference odd-or-even="even"
          master-reference="even"/>
        <fo:conditional-page-master-reference odd-or-even="odd"
          master-reference="odd"/>
      </fo:repeatable-page-master-alternatives>
    </fo:page-sequence-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="document">
    <fo:static-content flow-name="footer-even">
      <fo:block text-align="start"><fo:page-number/></fo:block>
    </fo:static-content>
    <fo:static-content flow-name="footer-odd">
      <fo:block text-align-last="end"><fo:page-number/></fo:block>
    </fo:static-content>
    <fo:flow flow-name="xsl-region-body">
      <fo:block/>
      <fo:block break-before="page"/>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

4.5. Making the First Page Special

To get a special header on the first page, one possibility is to insert it into the flow instead of the static content. Alternatively, use a page master referring to different page masters for the first page and the rest. Here is a code sample:

```
<?xml version="1.0"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="first"
```

```
      page-height="297mm" page-width="210mm"
      margin-top="20mm" margin-bottom="20mm"
      margin-left="25mm" margin-right="25mm">
      <fo:region-body margin-bottom="20mm"/>
      <fo:region-after region-name="footer-first" extent="20mm"/>
    </fo:simple-page-master>
    <fo:simple-page-master master-name="rest"
      page-height="297mm" page-width="210mm"
      margin-top="20mm" margin-bottom="20mm"
      margin-left="25mm" margin-right="25mm">
      <fo:region-body margin-bottom="20mm"/>
      <fo:region-after region-name="footer-rest" extent="20mm"/>
    </fo:simple-page-master>
    <fo:page-sequence-master master-name="document">
      <fo:repeatable-page-master-alternatives>
        <fo:conditional-page-master-reference page-position="first"
          master-reference="first"/>
        <fo:conditional-page-master-reference page-position="rest"
          master-reference="rest"/>
      </fo:repeatable-page-master-alternatives>
    </fo:page-sequence-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="document">
    <fo:static-content flow-name="footer-first">
      <fo:block text-align="center">First page.</fo:block>
    </fo:static-content>
    <fo:static-content flow-name="footer-rest">
      <fo:block text-align-last="center">Other page.</fo:block>
    </fo:static-content>
    <fo:flow flow-name="xsl-region-body">
      <fo:block/>
      <fo:block break-before="page"/>
      <fo:block break-before="page"/>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

4.6. Blank Pages

Sometimes it is desirable to insert blank pages within your output, starting, for example, a new chapter on an odd page or an even page. A blank page can be forced by using `break-before="page-even"` or similar properties, or by a force-page-count="end-on-odd" on a page sequence.

To write "This page is intentionally left blank" (or something similar) on an intentionally blank page, first define a conditional page master with a page master specific for blank pages. This allows you to specify static content for blank pages (by definition, a page is blank if no content from a flow is rendered on the page). Omit your normal headers and footers, and use (for example) an extended header to print the "..left blank" statement:

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

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="normal"
      page-height="297mm" page-width="210mm"
      margin-top="20mm" margin-bottom="20mm"
      margin-left="25mm" margin-right="25mm">
      <fo:region-body margin-bottom="20mm"/>
      <fo:region-after region-name="footer-normal" extent="20mm"/>
    </fo:simple-page-master>
    <fo:simple-page-master master-name="blank"
      page-height="297mm" page-width="210mm"
      margin-top="20mm" margin-bottom="20mm"
      margin-left="25mm" margin-right="25mm">
      <fo:region-body/>
      <fo:region-before region-name="header-blank" extent="297mm"/>
    </fo:simple-page-master>
    <fo:page-sequence-master master-name="document">
      <fo:repeatable-page-master-alternatives>
        <fo:conditional-page-master-reference blank-or-not-blank="not-blank"
          master-reference="normal"/>
        <fo:conditional-page-master-reference blank-or-not-blank="blank"
          master-reference="blank"/>
      </fo:repeatable-page-master-alternatives>
    </fo:page-sequence-master>
  </fo:layout-master-set>
  <fo:page-sequence master-reference="document"
force-page-count="end-on-even">
    <fo:static-content flow-name="footer-normal">
      <fo:block text-align="center">Normal footer</fo:block>
    </fo:static-content>
    <fo:static-content flow-name="header-blank">
      <fo:block space-before="100mm" text-align-last="center">
        Intentionally left blank.</fo:block>
    </fo:static-content>
    <fo:flow flow-name="xsl-region-body">
      <fo:block/>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

4.7. Preformatting Content

Sometimes it is desirable to retain linebreaks and hard spaces, and to get preformatted text to pass through without being changed. The XSL-FO specification provides some properties for this: white-space-collapse and linefeed-treatment. In FOP, use white-space-collapse="false" on an enclosing block.

Warning:

Due to a bug in current versions of FOP, setting white-space-collapse="false" will also preserve line breaks in the text. Do not rely on this behavior, as it is non-conformant and will be changed.

4.8. Total Document Pages

It is frequently desirable to know the total number of pages in a document and to use that number within the document. For example, you might wish to show the page number on the first page as being "page 1 of 12". To accomplish this, place an empty block with an id at the end of the flow:

```
<fo:flow ...>
    ...
  <fo:block id="last-page"/>
</fo:flow>
```

Get the number of the last page as follows:

```
 <fo:page-number-citation ref-id="last-page"/>
```

This does not work in certain situations: multiple page sequences, an initial page number other than 1, or forcing a certain page count, thereby producing blank pages at the end.

Warning:

There is no reliable way to get the real total page count with FO mechanisms. You can only get *page numbers*.

The FOP library provides a method to get the total page count after an FO document has been rendered. One possibility is to implement your own wrapper to do a dummy rendering, inquire the total page count and then perform the real rendering, passing the total page count to the XSLT processor to splice it into the generated FO. For example:

```
import org.apache.fop.apps.*;
import org.xml.sax.*;
import java.io.*;
import javax.xml.transform.*;
import javax.xml.transform.sax.*;
import javax.xml.transform.stream.*;

class rendtest {

  public static void main(String args[]) {
    try {
      Driver driver=new Driver();
      driver.setOutputStream(new FileOutputStream(args[2]));
      driver.setRenderer(Driver.RENDER_PDF);
      Transformer transformer=TransformerFactory.newInstance()
        .newTransformer(new StreamSource(new File(args[1])));
      transformer.setParameter("page-count","#");
      transformer.transform(new StreamSource(new File(args[0])),
        new SAXResult(driver.getContentHandler()));
      String pageCount=Integer.toString(driver.getResults().getPageCount());
      driver=new Driver();
      driver.setOutputStream(new FileOutputStream(args[2]));
      driver.setRenderer(Driver.RENDER_PDF);
      transformer=TransformerFactory.newInstance()
```

```
        .newTransformer(new StreamSource(new File(args[1])));
      transformer.setParameter("page-count",pageCount);
      transformer.transform(new StreamSource(new File(args[0])),
        new SAXResult(driver.getContentHandler()));
    }
    catch( Exception e) {
      e.printStackTrace();
    }
  }
}
```

Declare and use the parameter "page-count" in your XSLT.

Warning:

It is possible to run into a convergence problem with this solution. Replacing the "#" placeholder in the first run with the actual page count in the second run, may change the total number of pages in the document.

4.9. Aligning Regions

Although it may seem counterintuitive, the regions on a page may overlap. Defining a certain body region does not automatically constrain other regions. Instead, this has to be done explicitly. Sometimes for creative reasons it may be desirable to have the regions overlap. Otherwise, you will want to set them up so that the header does not overlap body content or the body extend into the footer.

Assuming you wish to keep the regions separate, if you have a header region with an extent of 20mm, you should define a margin for the body region of at least 20mm too. Otherwise the header content may overwrite some stuff in the body region. This applies similarly to the extent of the after region and the bottom margin of the body region.

4.10. Drawing Lines

It is frequently desirable to draw lines in a document, as separators, side bars or folding marks. There are several possibilities:

- Horizontal lines can be drawn using fo:leader.
- Use a solid border on a suitable fo:block. This will work for horizontal and vertical lines only.
- Insert a graphic. GIF, PNG SVG, whatever.

4.11. Validating XSL-FO

RenderX has provided an Unofficial DTD for FO Documents, which may be helpful in

validating general FO issues.

FOP also maintains an Unofficial FOP Schema in the FOP CVS Repository. This document can be used either to validate against the FO standard, or against the actual FOP implementation. See the notes near the beginning of the document for instructions on how to use it.

4.12. Producing landscape pages

Pages in landscape format can easily be produced by exchanging the page-height and page-width values of a simple-page-master element.

```
<fo:layout-master-set>
  <fo:simple-page-master master-name="A4-portrait" page-height="29.7cm"
page-width="21cm" [..]>
    <fo:region-body/>
  </fo:simple-page-master>
  <fo:simple-page-master master-name="A4-landscape" page-height="21cm"
page-width="29.7cm" [..]>
    <fo:region-body/>
  </fo:simple-page-master>
</fo:layout-master-set>
```

4.13. External Resources

Resources needed by an XSL-FO file that are external to it (graphics, for example), are defined in the XSL-FO standard as being of type "uri-specification". This is defined in the standard at Section 5.11 Property Datatypes, which includes a link to the URI standard itself. Refer to the XSL-FO and URI standards themselves for more detailed instructions.

URIs may be either absolute or relative to a base URI. (See FOP: Configuration for information on setting the base URI for a FOP session). Here is an example referencing an external-graphic that is relative to the base URI:

```
<fo:external-graphic src="url('images/logo.jpg')"/>
```

Here is an example referencing an external-graphic that is an absolute URI on a local filesystem:

```
fo:external-graphic src="url('file:d:///images/logo.jpg')"/>
```