abc

Contents

Book	Author	Price
how_to_XML_XSLFO	Juhi	110

110

Cover

- Output title, date and author to a cover page. which are children of the head element in the cover page. but not output the abstract.
- The width of the block that contains the title is 130mm, the height is 30mm, and the block must be centered, using gray for the background color, dark gray for the border color. The title is placed 25mm down from the margin top, and make a 122mm height space between the title and the author to be written next. Use the font size 24pt, font style sans-serif. the text must be centered inside the block.
- The width of the block that contains the date is 160mm and the block must be centered, using no background color, no border. Use the font size 14pt, font style serif. Make a 5mm height space between the date and the author.。
- The width of the block that contains the author is 160mm and the block must be centered, using no background color, no border. Use the font size 14pt, font style serif. When an image of logotype is specified to the author, place it preceding the author.

The cover is created by the templates that process the head.

The layout specification of a title portion is arranged by the portion of name="cover.title" in xsl:attribute-set.

Layout specification of title, author, date.

```
<!-- cover -->
<xsl:attribute-set name="cover.title" >
  <xsl:attribute name="space-before">25mm</xsl:attribute>
  <xsl:attribute name="space-before.conditionality">retain</xsl:attribute>
  <xsl:attribute name="space-after">122mm</xsl:attribute>
  <xsl:attribute name="font-size">24pt</xsl:attribute>
  <xsl:attribute name="font-family">"sans-serif"</xsl:attribute>
  <xsl:attribute name="text-align">center</xsl:attribute>
  <xsl:attribute name="text-align-last">center</xsl:attribute>
  <xsl:attribute name="start-indent">18mm</xsl:attribute>
  <xsl:attribute name="width">130mm</xsl:attribute>
  <xsl:attribute name="height">30mm</xsl:attribute>
  <xsl:attribute name="background-color">#EEEEEE</xsl:attribute>
  <xsl:attribute name="border-style">outset</xsl:attribute>
  <xsl:attribute name="border-color">#888888</xsl:attribute>
  <xsl:attribute name="padding-top">5pt</xsl:attribute>
  <xsl:attribute name="padding-bottom">5pt</xsl:attribute>
</xsl:attribute-set>

<xsl:attribute-set name="cover.date" >
  <xsl:attribute name="space-after">5mm</xsl:attribute>
  <xsl:attribute name="font-size">14pt</xsl:attribute>
  <xsl:attribute name="font-family">"serif"</xsl:attribute>
  <xsl:attribute name="text-align">center</xsl:attribute>
  <xsl:attribute name="text-align-last">center</xsl:attribute>
  <xsl:attribute name="width">160mm</xsl:attribute>
</xsl:attribute-set>

<xsl:attribute-set name="cover.author" >
  <xsl:attribute name="font-size">14pt</xsl:attribute>
  <xsl:attribute name="font-family">"serif"</xsl:attribute>
  <xsl:attribute name="text-align">center</xsl:attribute>
  <xsl:attribute name="text-align-last">center</xsl:attribute>
```

```
  <xsl:attribute name="width">160mm</xsl:attribute>
</xsl:attribute-set>
```

Followings shows the point which should be aware of:

- In order to layout the title, we use fo:block-container.
- The Property space-before ="25 mm" is specified to the fo:block-container. This title becomes the first block of the region body. However, space-before in the first block of this region body will be discarded by default, and a title will be arranged at the upper end of this region body. By considering as space-before.conditionality="retain", a space can be obtained compulsorily.

If the logo attribute is specified to the author, it is rendered as the image. This is processed by the author.logo.img template. The pos attribute specify to put the image on the left or top of the author. The example of an author name with a picture may be shown at the cover of this document.

Templates that transform the head elements

```
<xsl:template match="doc/head">
  <fo:page-sequence master-reference="PageMaster-Cover">
    <fo:flow flow-name="xsl-region-body">
     <fo:block-container xsl:use-attribute-sets="cover.title">
       <xsl:apply-templates select="/doc/head/title"/>
     </fo:block-container>
      <fo:block-container xsl:use-attribute-sets="cover.date">
        <xsl:apply-templates select="/doc/head/date"/>
      </fo:block-container>
      <fo:block-container xsl:use-attribute-sets="cover.author">
        <xsl:apply-templates select="/doc/head/author"/>
      </fo:block-container>
    </fo:flow>
  </fo:page-sequence>
</xsl:template>

<xsl:template match="doc/head/title">
  <fo:block start-indent="0mm">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>

<xsl:template match="doc/head/date">
  <fo:block>
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>

<xsl:template match="doc/head/author">
  <fo:block>
    <xsl:if test="@logo">
      <xsl:call-template name="author.logo.img"/>
    </xsl:if>
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>

<xsl:template name="author.logo.img">
  <xsl:choose>
    <xsl:when test="@pos='side'">
      <fo:inline space-end="1em">
        <fo:external-graphic src="{@logo}">
          <xsl:if test="@width and @height">
```

```
            <xsl:attribute name="content-width">
              <xsl:value-of select="@width"/>
            </xsl:attribute>
            <xsl:attribute name="content-height">
              <xsl:value-of select="@height"/>
            </xsl:attribute>
          </xsl:if>
        </fo:external-graphic>
      </fo:inline>
    </xsl:when>
    <xsl:otherwise>
      <fo:block space-after="1em">
        <fo:external-graphic src="{@logo}">
          <xsl:if test="@width and @height">
            <xsl:attribute name="content-width">
              <xsl:value-of select="@width"/>
            </xsl:attribute>
            <xsl:attribute name="content-height">
              <xsl:value-of select="@height"/>
            </xsl:attribute>
          </xsl:if>
        </fo:external-graphic>
      </fo:block>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

The template structure is very simple. xsl:use-attribute-sets calls a group of properties defined by the xsl:attribute-set element and applies them to fo:block-container that matches to each title, date and author. In each fo:block-container, each template is applied to each element of title, date, and author.

Table of Contents

- A table of contents is positioned next to the cover by feeding a page. The title is "Table of Contents". The background color is gray.
- A table of contents is created by collecting the title elements of of part, chapter, section, subsection, and subsubsection in an XML document.
- The contents of each line consist of the each title in the part, chapter, section, subsection, leaders (rows of dots) and a page number.
- Space before, left indent, font size, font weight are specified in each line according to the nest level of each part, chapter, section, subsection.
- The internal link from each line of a table of contents to the title in the text is set for a PDF output.

Templates for Creating a table of contents

A table of contents is created by the toc template. The toc template is called from the templates that process the root elements doc, by using <xsl:call-template name="toc">

Toc template

```
<xsl:template name="toc">
  <!-- generate fo:page-sequence-->
    <fo:page-sequence master-reference="PageMaster-TOC">
    <!-- generate flow applied to region-body -->
    <fo:flow flow-name="xsl-region-body" >
      <!--generate a block of table of contents-->
      <fo:block xsl:use-attribute-sets="div.toc">
        <!--generate the title "Table of Contents"-->
        <fo:block xsl:use-attribute-sets="h2">Table of Contents</fo:block>
        <!-- select the elements of part, chapter, section, subsection,
                 subsubsection from the whole XML documents-->
        <xsl:for-each select="//part |
                              //chapter |
                              //section |
                              //subsection |
                              //subsubsection">
          <!-- apply template for each element
               to generate each line of the contents.-->
          <xsl:call-template name="toc.line"/>
        </xsl:for-each>
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</xsl:template>
```

The toc template processes a table of contents in the following order.

1. Create a new page-sequence. This page-sequence refer to fo:simple-page-master described in master-reference="PageMaster-TOC" for a new page layout. Because the new page-sequence is created, the page is broken when it is printed.

2. Next, generate fo:flow object in the region-body. Create a block that contains a whole table of contents by applying attribute-set that is a name of div.toc. This attribute-set specifies the background color gray. Then, create a title "Table of Contents".
3. xsl:for-each select="..." is used to create a set of nodes consisting of parts, characters, sections, subsections, subsubsections of the whole document. Then, each node is sent to toc.line template that processes a line of the contents. Lines of a table of contents are alined in the order of appearance of the corresponded node in the XML document tree.

This template is called from the template that processes the doc elements. So, "current node" is the doc element node. xsl:for-each change this current node into each node group specified by the select attribute. Therefore, the current node is one of the five elements, part, chapter, section, subsection, and subsubsection in the toc.line template. When xsl:for-each finished processing, the current node returns to the previous doc element node.xsl:for-each

Templates for Creating Lines of TOC

The toc.line template creates a line of contents

toc.line templates that create each line of contents

```
<!-- global parameter and variable used when creating the table of contents. -->
<xsl:param name="toc-level-default" select="3"/>
<!-- The template that creates the table of contents -->
<xsl:variable name="toc-level-max">
  <xsl:choose>
    <xsl:when test="not (doc/@toclevel)">
      <xsl:value-of select="$toc-level-default"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="number(doc/@toclevel)"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>

<xsl:template name="toc.line">
  <!-- Count the nest level of current node,
    set the value to "level" local variable. -->
  <xsl:variable name="level" select="count(ancestor-or-self::part |
                                        ancestor-or-self::chapter |
                                        ancestor-or-self::section |
                                        ancestor-or-self::subsection |
                                        ancestor-or-self::subsubsection )"/>
  <!-- Test if the nest level can be a target. -->
  <xsl:if test="$level &lt;= $toc-level-max">
    <!-- Create fo:block for each line of toc. -->
    <fo:block  text-align-last="justify">
      <!-- Widen the margin left in proportion to a nest level.-->
      <xsl:attribute name="margin-left">
        <xsl:value-of select="$level - 1"/>
        <xsl:text>em</xsl:text>
      </xsl:attribute>

      <!-- space-before becomes larger in proportion
           that the nest level becomes upper.-->
      <xsl:attribute name="space-before">
          <xsl:choose>
          <xsl:when test="$level=1">4pt</xsl:when>
          <xsl:when test="$level=2">3pt</xsl:when>
          <xsl:when test="$level=3">3pt</xsl:when>
```

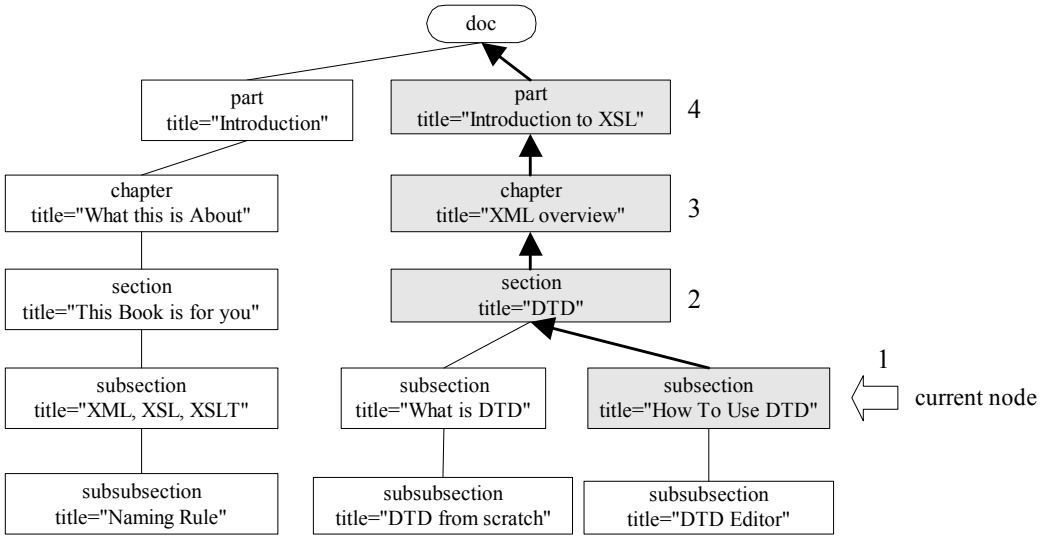
```
          <xsl:otherwise>1pt</xsl:otherwise>
        </xsl:choose>
      </xsl:attribute>
      <!-- font-size is processed in the same way-->
      <xsl:attribute name="font-size">
        <xsl:choose>
          <xsl:when test="$level=1">1em</xsl:when>
          <xsl:otherwise>0.9em</xsl:otherwise>
        </xsl:choose>
      </xsl:attribute>
      <!-- font-weight is also processed in the same way -->
      <xsl:attribute name="font-weight">
        <xsl:value-of select="800 - $level * 100"/>
      </xsl:attribute>
      <!-- Below is the data of the table of contents -->
      <xsl:value-of select="title"/>
      <fo:leader leader-pattern="dots"/>
      <!-- Output fo:page-number-citation.
           Formatter replaces it to the page number. -->
      <fo:page-number-citation ref-id="{generate-id()}"/>
    </fo:block>
  </xsl:if>
</xsl:template>
```

The toc template processes a table of contents in the following order.

1. Count the depth of the nest of the current node (nest level) from the root node, then set the value to a level variable.
2. If a nest level is on or below the level of "toc-level-max", it is processed. If not, it is not processed. The level of "toc-level-max" is specified by the toclevel property of the doc element. If the level is not specified, the value is 3.
3. Create fo:block for each line in the table of the contents.
4. According to the depth from the root node, determine the the property value of indent, font size, font weight.
5. Output title, leader, and page number which is the real data of a line in a table of contents. Enclose the title of a table of contents with fo:basic-link and set the link from the title to the body text. Generated PDF is set as an internal link. (Details about fo:basic-link are explained in the 'Functions for creating PDF' section in this document.)

Counting the Nest Level

The nest level can be counted as follows: count(ancestor-or-self::part | ancestor-or-self::chapter | ancestor-or-self::section | ancestor-or-self::subsection | ancestor-or-self::subsubsection). In other words, count itself which is a child of the doc elemnet or the ancestor nodes. This chart is shown below:



※count(ancestor-or-self::part|ancestor-or-self::chapter|ancestor-or-self::section|ancestor-or-self::subsection) returns the number of the ancestor nodes, including itself, of part, chapter, section, subsection, subsubsection.
For example in case that current node is title="How To Use DTD", count(...) function returns the value 4.

Count the nest level from the root element

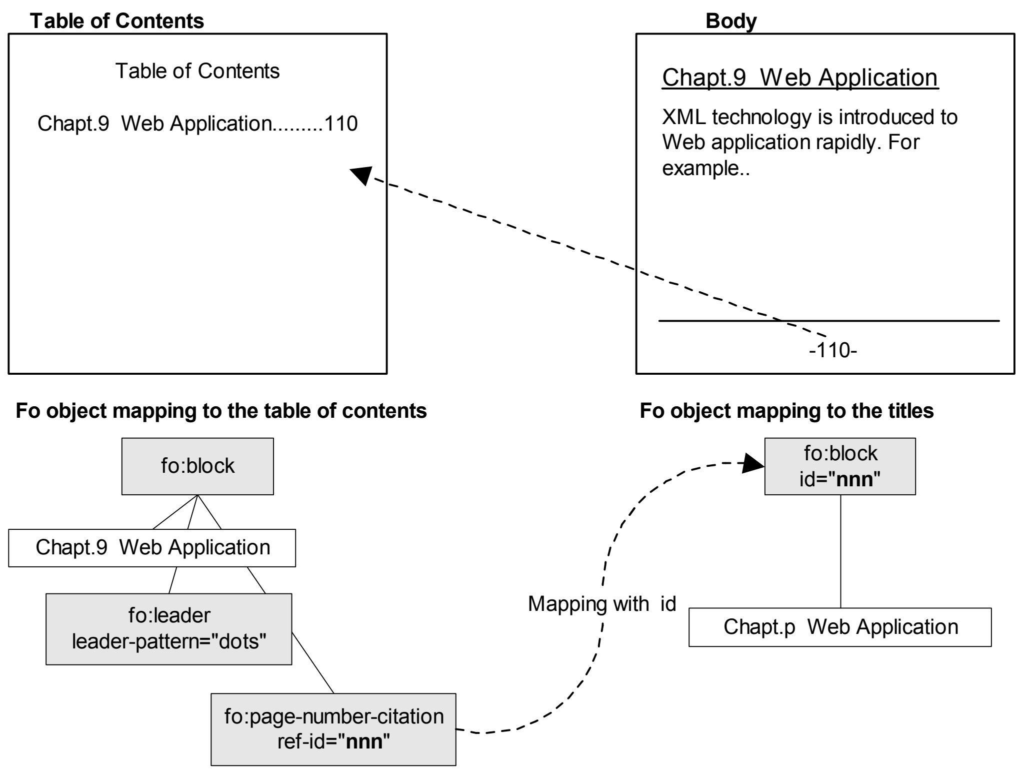
Setting properties according to the nest level

Set fo:block properties according to the nest level of the current node. Note that in this case, the properties are not set according to the element such as part, chapter, section, subsection. By setting properties according to the nest levels, the table of contents can be generated without depending on the elements used, but using the same format. Next table shows the properties set in the stylesheet.

Property	Nest level				
	1	2	3	4	5
margin-left	0em	1em	2em	3em	4em
space-before	5pt	3pt	1pt	1pt	1pt
font-size	1em	0.9em	0.9em	0.9em	0.9em
font-weight	700	600	500	400	300

Getting page numbers

You have to create page numbers that show the page where each part, chapter, section, subsection appear. **Page numbers cannot be fixed until the Formatter finishes formatting the XSL-FO instance.** In order to solve this problem, XSL-FO provides the function fo:page-number-citation. XSL Formatter replaces fo:page-number-citation with the page number in the end of the processing. The ref-id property specifies which page number is replaced. XSL Formatter finds the formatting object that has the same value in the id property as that specified by ref-id. Then, the page number that the formatting object belongs to is given. Therefore fo:block generated from the part, chapter, section, subsection elements must have the id properties. This mechanism is shown below.



Mechanism of fo:page-number-citation

In the template, the generate-id()function is used as the value of the ref-id property. XSLT processor generates the unique characters using generate-id() to distinguish the current node.

fo:leader

Use fo:leader between the title of the contents and the page numbers. fo:leader is a special object for generating inline area. In the example, leader-pattern="dots" is specified. It plays the role to fill the space between the title and the page number.

It is important that text-align-last="justify" in fo:block specifies to justify the entire line. So, the titles are left-justified and the page numbers are right-justified, then the leader pattern fill the space between them.

fo:leader property can specify various patterns as shown below. fo:leader properties are written in the left side.

leader-pattern="dots".. 99
leader-pattern="use-content" (content="*")*** 99
leader-pattern="rule" rule-style="dotted".. 99
leader-pattern="rule" rule-style="dashed"- 99
leader-pattern="rule" rule-style="solid"____ 99
leader-pattern="rule" rule-style="double"____ 99
leader-pattern="rule" rule-style="groove"____ 99
leader-pattern="rule" rule-style="ridge"____ 99

The following pattern can also be specified.

```
<fo:leader leader-pattern="use-content">+</fo:leader>
```

Arbitrary specification of a pattern ++ 99

Example of the generated contents

Shown below is an example of toc created by taking the steps described.

Generated table of contents

```
<fo:block text-align-last="justify" margin-left="0em" space-before="5pt" font-
size="1em" font-weight="700">
  Preface
  <fo:leader leader-pattern="dots"/>
  <fo:page-number-citation ref-id="IDA4AIOB"/>
</fo:block>
```

See the table of contents in this report for an output example.

Body

- Process all the descendants or self of the body elements in the source XML document.
- Each page layout of the body consists of a page header, a page footer, and a body region. The contents of a page header and a page footer are arranged in a symmetrical position by odd-numbered and even-numbered pages.
- When a page has a footnote, the line which divides body region with footnote region is drawn.

Templates for Processing a Body

The body in the XML source document is contained to the descendants or self of the body element. Shown below are the templates that process the body element.

Templates that process the body element

```
<!-- process of body element -->
<xsl:template match="body">
<!-- start page number is 1 -->
  <fo:page-sequence master-reference="PageMaster" initial-page-number="1">
    <!-- left page header -->
    <fo:static-content flow-name="Left-header">
          <!-- snip-->
    </fo:static-content>
    <!-- right page header -->
    <fo:static-content flow-name="Right-header">
          <!-- snip-->
    </fo:static-content>
    <!-- left page footer -->
    <fo:static-content flow-name="Left-footer">
          <!-- snip -->
    </fo:static-content>
    <!-- right page footer -->
    <fo:static-content flow-name="Right-footer">
          <!-- snip -->
    </fo:static-content>

    <!-- leader for footnote-separator -->
    <fo:static-content flow-name="xsl-footnote-separator">
      <fo:block>
        <fo:leader leader-pattern="rule" rule-thickness="0.5pt" leader-length="33%"/>
      </fo:block>
    </fo:static-content>

    <!-- body -->
    <fo:flow flow-name="xsl-region-body">
      <fo:block>
        <xsl:apply-templates/>
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</xsl:template>
```

This template pocesses the body elements as shown below.

1. Generate a fo:page-sequence based on the new "PageMaster", the page layout right after the table of contents is changed.
2. Set header/footer regions based on the page layout. Place the title of the document in the header, page numbers in the footer.
3. Create a border region between the body and the footnote by using leaders.
4. Create flow objects in the body region.
5. xsl:apply-templates processes the descendants or self of the body element.

A page header and a page footer are described in fo:static-content. In the body page, in order to change the page layout on right and left pages, four fo:static-content should be prepared which are used for the right and left pages of the footer, and for the right and left pages of a header. Moreover, the border between the text and a footnote is created using also fo:static-content.

Four of fo:static-content are mapped to the region of a page as follows. The layout for right and left pages are defined in fo:simple-page-master of the body, which is prepared by the "Body - Change Page Layout on Right and Left pages" section in this document, then the header region and footer regions of each page are named, respectively. On the other hand, flow-name is specified to fo:static-content and the contents of fo:static-content are poured into the region whose name maps to theregion-name .

Page	Name of region	Name of static-content
Right page heade	fo:region-before region-name="Right-header"	fo:static-content flow-name="Right-header"
Right page footer	fo:region-after region-name="Right-footer"	fo:static-content flow-name="Right-footer"
Left page header	fo:region-before region-name="Left-header"	fo:static-content flow-name="Left-header"
Left page footer	fo:region-after region-name="Left-footer"	fo:static-content flow-name="Left-footer"

The border between a footnote and the text is created by fo:static-content with flow-name called xsl-footnote-separator. The fo:leader object is used for drawing a line. One third of the width of the body region is secured as a solid line.

The content of the body is outputted as a child of fo:flow.

Page Number Setting

It is possible to set up the initial value of a page number using the initial-age-number property belonging to fo:page-sequence. In SD2FO-DOC.xsl, initial-page-number="1" is set to fo:page-sequence of the body, 1 page is made to begin from the body in SD2FO-DOC.xsl.