# XSL Formatting Objects

Lecture on

## XSL Formatting Objects

„From XML to PDF“

Walter Kriha



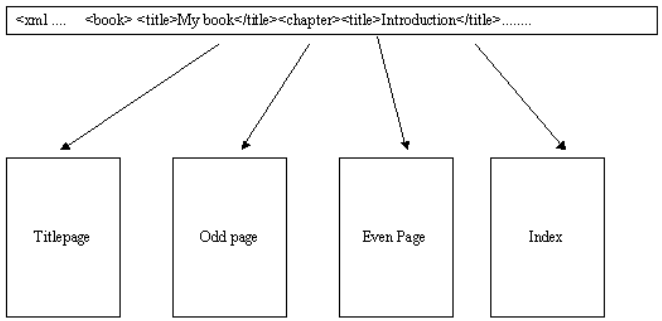
## Goals

- Learn the basic concepts of a page description language
- Learn the core elements of XSL-FO
- Understand a simple hand-coded xsl-fo example
- Learn the processing pipeline from xml to pdf
- Start generating fo files with XSLT

XSL-FO is not so much different from e.g. PDF format. All page description languages need certain constructs e.g. to define blocks, alignments, fonts etc.



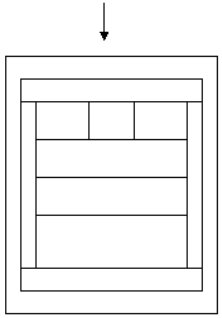
## Putting Text on Pages




All pages may be the same or each one can have a different layout. Double-sided printing changes the layout compared to single-sided printing because odd pages have different margins. What we need is a way to describe areas of pages and fill them with content from our text. In xsl fo we create those areas by defining FORMATTING OBJECTS on pages. The fo processor will then take our text and fill it into those formatting objects.

# Putting Text on a Page

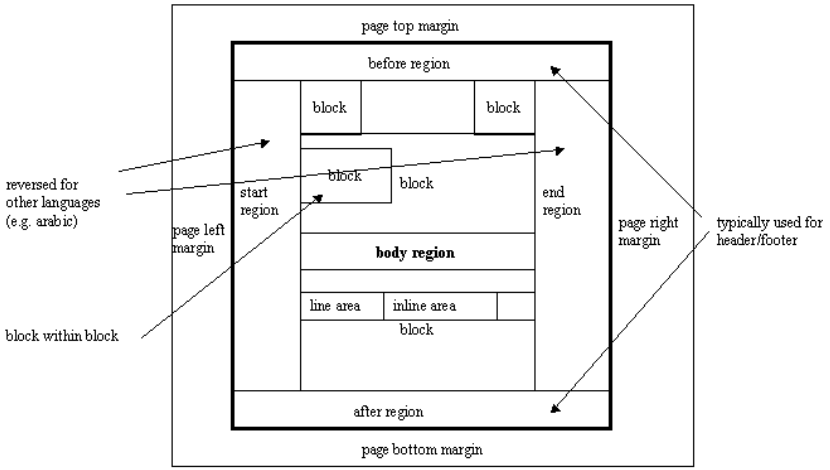
<xml .... <book> <title>My book</title><chapter><title>Introduction</title>.....



Formatting objects devide a page into rectangular areas which either stay empty (like margins) or get filled with static content (header/footer) or dynamic content (rest of text)



# Defining a page using formatting objects




The main structuring elements are shown: page, regions, blocks (with embedded blocks) and line areas within blocks (inline elements). A block always creates a break. Please note that the body region encloses the other regions. Therefore body margins must be equal or larger than the other region margins to not overlap with them.



# Main Page Formatting Objects

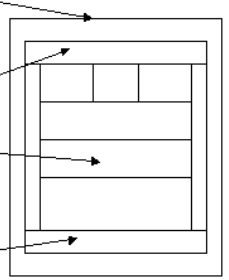
- fo:region-before (for header)
- fo:region-after (for footer)
- fo:region-start, fo:region-end
- fo:body (main text area)
- fo:leader (e.g. to make a rule)
- fo:block
- fo:inline

These elements define the layout of a page. Of course they can be qualified with a large number of attributes, e.g. defining border width, language, alignment etc. The key question now is how do we define different page layouts? We need a templating mechanism. This is easily constructed using xml syntax. XSL fo defines a „Page master“ element which takes a name and has the page formatting objects as children.



# Simple Page Master

```
<fo:simple-page-master master-name="body-first"
                       page-width="{$page.width}"
                       page-height="{$page.height}"
                       margin-top="{$page.margin.top}"
                       margin-bottom="{$page.margin.bottom}"
                       margin-left="{$page.margin.inner}"
                       margin-right="{$page.margin.outer}">
  <fo:region-body margin-bottom="{$body.margin.bottom}"
                  margin-top="{$body.margin.top}"
                  column-count="{$column.count.body}">
  </fo:region-body>
  <fo:region-before region-name="xsl-region-before-first"
                    extent="{$region.before.extent}"
                    display-align="before"/>
  <fo:region-after region-name="xsl-region-after-first"
                   extent="{$region.after.extent}"
                   display-align="after"/>
</fo:simple-page-master>
```



The simple page master currently defines rectangular areas of a page. Later fo versions could also define circular areas. Region-start and region-end are not used in this case. Please note the name „body-first“. Other fo elements will use this name to refer to specific page masters if they need a page template definition. Page masters can be compared to powerpoint layout masters. Example taken from Norman Walsh's docbook-xsl tool.

# Properties as Parameters

configuration file:

```
<xsl:param name=„page.height" select=„297mm"/>
```

fo simple page master:

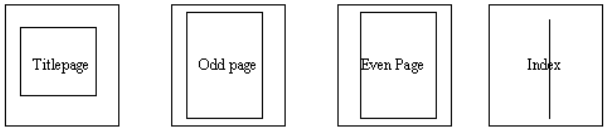
```
page-height="{$page.height}"
```

This is a clever way to make stylesheets configurable. Different configuration files can then define different page values, e.g. for A4 or letter pages. You can use this mechanism with any xsl stylesheets, not just those producing fo files.



# Different Page Layouts

```
<fo:layout-master-set>
        <fo:simple-page-master master-name=„Titlepage„>
                <fo:region....
        <fo:simple-page-master master-name=„Oddpage„>
                <fo:region....
        <fo:simple-page-master master-name=„Evenpage„>
                <fo:region....
```




If a document needs different page layouts, the simple page masters are collected within a fo:layout-master-set. This set will also contain page sequence masters (see later). Again, note the names of the page layouts by which they will be referenced.



# Layout Properties

- margins (page, regions, body)
- border (style, width, right, left, spacing, separation, color)
- background (color, image, position, repeat)
- padding (after, before, left, right, end, start)
- page-break (before, after, inside)
- font (family, size, stretch, style, variant, weight)

This is only a sample of all available properties. Many properties have the same name and meaning as specified in CSS2, e.g. font properties.



# Output Producing Elements

- fo:static-content (zero or more per region)
- fo:flow (max. one per region)
- fo:page-sequence (parent of fo:flow and fo:static-content). It references a simple-page-master element as a page template.

Text from the xml document becomes output only within static-content or flow elements. Static content means content that appears on several pages (e.g. a page number, a header or footer). Regular text is produced within fo:flow an put on a page as long there is enough room on this page. If the page is full and a simple page master template is used, the fo processor will instantiate a new page from the page master template and continue filling in text. The process ends when all text has been put on pages.

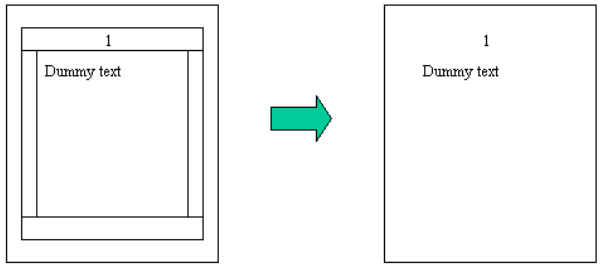
# A mini fo example

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:fo="http://www.w3.org/1999/XSL/Format„ version='1.0'>
<xsl:output method="xml" indent="no"/>
<fo:root language=„en">
   <fo:layout-master-set>
    <fo:simple-page-master master-name=„onepage„><fo:region-before/>fo:region-
          body/></fo:simple-page-master>
   </fo:layout-master-set>
   <fo:page-sequence master-reference=„onepage">
     <fo:static-content flow-name=„xsl-region-before"> <fo:block font-size=„10pt" font-
          family=„serif"> <fo:page-number> </fo:block>
     </fo:static-content>
     <fo:flow flow-name=„xsl-region-body">  <!- points to the body region of page master →
        <fo:block font-size=„20pt" font-family=„serif"> Dummy Text </fo:block>
     </fo:flow>
   </fo:page-sequence>
</fo:root>
```

We have defined one page layout template called „onepage“ and one processing sequence which fills the template with a bit of text. The flow-name attribute points to the specific region of the layout template where the text should go.



# Result




Instead of just giving a static piece of dummy text we could use xsl:apply-templates match=„someElement“ to fill in content from our xml text base.



# block level elements

- fo:block (like a paragraph)
- fo:table
- fo:table-with-caption
- fo:block-list

These elements are close to those in HTML and work similiar. They make it easier to output tables or lists. Every block level element will cause a line break. They need to be embedded in a fo:flow or fo:static-content element to produce output.



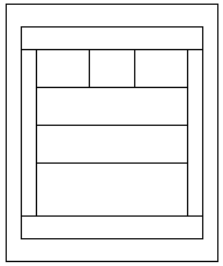
# HTML Tables vs. FO Tables

| HTML Element | XSL FO Element |
|---|---|
| TABLE | fo:table-and-caption |
| no equivalent | fo:table |
| CAPTION | fo:table-caption |
| COL | fo:table-column |
| COLGROUP | no equivalent |
| THEAD | fo:table-header |
| TBODY | fo:table-body |
| TFOOT | fo:table-footer |
| TD | fo:table-cell |
| TR | fo:table-row |

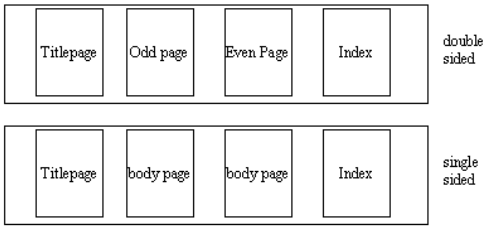
from http://www.ibiblio.org/xml/books/bible2/chapters/ch18.html, elliotte rusty harolds excellent introduction to fo. (see resources)

# From page layout to document layout

fo:simple-page-master define the layout of single pages



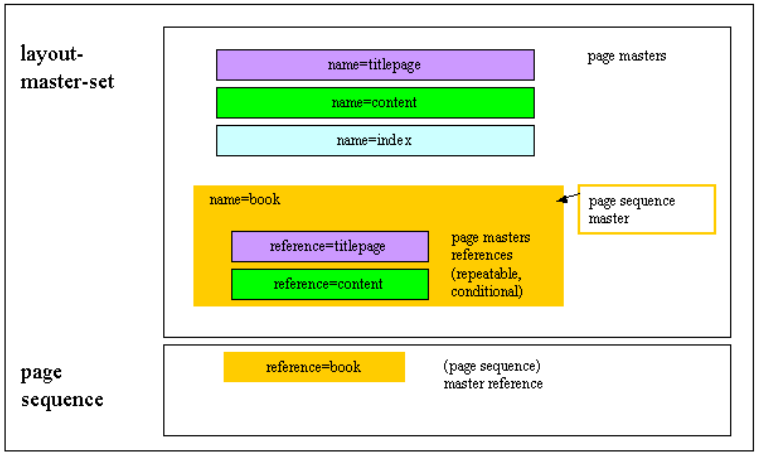
fo:page-sequence-master define the layout of complete documents as a series of different pages




So page-sequence masters are templates for complete documents. They refer via master-references to different page-master templates. This allows us to define a different layout for single-sided or double-sided printing of the same document



# Page Sequence Masters




the book page sequence master includes via reference two different page layouts. For each page layout one can specify if it should be repeated, how many times and under what circumstances. A page sequence needs only refer to a page sequence master to get the proper order of page layouts.



# example

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:fo="http://www.w3.org/1999/XSL/Format„ version='1.0'>
<xsl:output method="xml" indent="no"/>
<fo:root language=„en">
   <fo:layout-master-set>
     <fo:simple-page-master master-name=„title„><fo:region-body/></fo:simple-page-master>
     <fo:simple-page-master master-name=„content„><fo:region-body/></fo:simple-page-master>
   </fo:layout-master-set>
   <fo:page-sequence-master master-name=„article">
          <fo:single-page-master-reference master-reference=„title"/>
          <fo:repeatable-page-master-reference master-reference=„content"/>
   <fo:page-sequence master-reference=„article">
    ........
   </fo:page-sequence>
</fo:root>
```

the sequence master defines one page of layout „title“ and the rest of the xml text gets copied into instances of „content“ pages. Page masters can get referenced conditionally, e.g. if an odd page or an even page needs to be created.



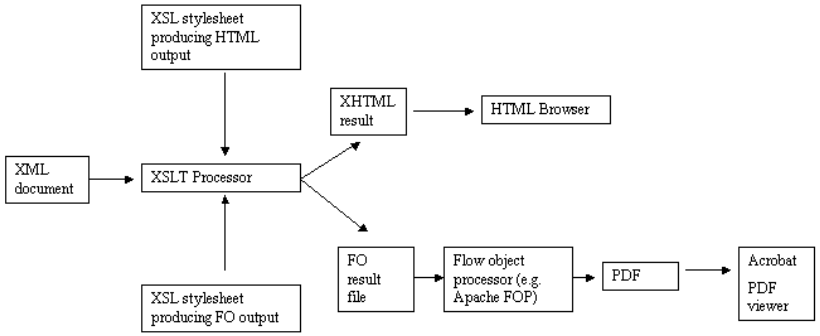
# Graphics

Supported formats are gif, jpeg and png. This depends of course on the fo engine used.

```
<fo:flow font-family="Helvetica" flow-name="xsl-region-body">
<fo:block>
<fo:external-graphic src="pictures/Folie39.PNG" width="auto"
height="auto" content-width="auto" content-height="auto"/>
</fo:block>
</fo:flow>
```

An interesting option is the inclusion of SVG documents. SVG is vector graphics defined using XML. Non-xml graphics is only referenced via the „src“ attribute and not embedded.

# The XSL FO processing pipeline




FO files are NOT hand-written. They are produced pretty much like our html files by using an XSLT stylesheet processor. The result is not html but FO which is simply an XML file containing formatting objects and embedded document content. The FO file is a complete description of the document layout with content. This file gets interpreted by an FO processor which can generate PDF or SVG from it. External graphic references are resolved by the FO processor.



# Resources

• Elliotte Rusty Harold, The XML Bible, Chapter 18. Very good book, online version available from www.ibiblio.org/xml/books/bible2/chapters/ch18.html . I found this introduction to XSL-FO very good.

•Dave Pawson, XSL-FO. Making XML look good in print. The author also maintains the XSL-FAQ. Very good introduction, easy to understand.

• Dave Pawson, a gentle introduction to xsl-fo. www.dpawson.co.uk/xsl/sect3/xsl-fo.html . Read this first if you don‘t want to start with the book right away.

• J. David Eisenberg, Using XSL Formatting Objects. Get it from www.xml.com . It contains nice graphics explaining how margins and borders work in FO.

If you really want to work with FO you will probably also need the XSL book from Michael Kay because most FO gets generated, not hand-coded.