FOP's Design and Implementation

Simon Pepping

FOP team, Apache Software Foundation

Renaud Richardet

FOP contributor, Apache Software Foundation

FOP's Design and Implementation

by Simon Pepping and Renaud Richardet

FOP is a Formatting Object Processor of the Apache project. It aims to provide a standards compliant implementation of XSL-FO. It understands Unicode, has bidirectional writing capabilities, and implements a wide range of rendering formats.

FOP is a work in progress. Its code is under continuing development. This documentation describes the state of the code at the time the documentation was written. At the time you read this documentation the code may be different. Note also that different parts of the documentation were written or revised at different times.

Revision History

Revision 0.x	2003, 2004
Various early, partial versions	
Revision 1.0	01 August 2004
Committed to the FOP code repository	
Revision 1.0.x	2005
Various updates, see the FOP code repository	

Table of Contents

Chapter 1
Overview

A FOP process has three stages:

1. building the FO tree,
2. building the Area tree,
    a. The getNextBreakPoss call tree
    b. The addAreas call tree
    c. Finishing the page
3. rendering the pages.

These stages are preceded by two other stages:

- 0. *preparation:* the objects for the FOP process are constructed; this phase may be done by FOP (apps package), or by the calling application;
- X. *parsing:* this phase is done by a SAX parser; FOP's own preparation stage uses a JAXP SAXParserFactory to call an available SAX parser.

Stage 0. The preparation stage occurs before the other stages. When it is completed it starts the parsing stage by calling the parser's `parse` method.

The parsing stage and stages 1, 2, 3 run concurrently. Each stage calls into the following stage at appropriate points in its process. It is feasible that stages 2 and 3 will run in separate threads.

Stage X. The parser now takes control. It parses the FO document or walks over the DOM tree. When it encounters XML events such as the start or end of the document, the start or end of an XML element, or character data, it makes a call back into the FO tree builder.

Stage 1. The FO tree builder now takes control to create or finish the FO node for which the XML event was fired. When it is done, it returns control to the parser.

The end events of a few XML elements invoke further actions of the FO tree builder. When a page-sequence FO node is finished, the FO tree builder notifies its tree listeners (of which there usually is only one, the Area tree builder) of this event. Each listener in turn takes control to process the page sequence.

Stage 2. The Area tree builder (which is the tree listener) now takes control to lay out the page sequence and construct the Area tree for it. This stage is divided in three substages.

Stage 2a. The (pseudo)tree of possible break points is created. Each node in the FO tree creates a Layout Manager of the appropriate type. The resulting tree of Layout Managers is recursed. Each Layout Manager asks each of its child Layout Managers in turn to return a possible break point, until all child Layout Managers are finished. This process is started by the Page Layout Manager, which is tied to the page-sequence FO node that was just completed in stage 1. Each time when its current child Layout Manager returns a possible break point, a complete (pseudo)tree of possible break points for a page has been collected. The next substage takes control.

Stage 2b. Now the area tree is created. The (pseudo)tree of possible break points and their Layout Managers is recursed. Each possible break point may create an area. It then calls the possible break points of the child LM that fall in this area, to create and return their area, and adds those areas to its own area. This process is started by the Page Layout Manager after the previous substage has finished. When its possible break point returns its

area, the area tree for the flow of the page is complete.

Stage 2c. The Page Layout Manager adds the static areas and hands the page to the Area tree builder, which adds it to the root area. The Area tree builder now inspects the set of complete pages, and fills in forward references to the page just finished. Pages which are now complete including the forward references contained in them, are handed over to the renderer, which now takes control. When the renderer returns control, the Page Layout Manager starts again in stage 2a to lay out the next page in the page sequence.

When all pages of this page sequence are done, this stage finishes, and the Area tree builder returns control to the FO tree builder.

Stage 3. The renderer now takes control to render the finished pages. When it is done with those pages, it returns control to the Area tree builder.

This process model is FOP's default process model. It is completely configurable, through the objects constructed in the preparation stage. Stage 1 is configured by the content handler that is registered with the parser. Stage 2 is configured by the listeners that are registered with the FO tree builder. The layout process in stage 2 is also configured by the layout strategy that is registered with the Area tree builder. [It might be more appropriate to say that stage 2 is controlled by the tree control object. The actual Area tree builder is assigned by the layout strategy.] Stage 3 is configured by the selected renderer or output format.

Chapter 2

Phase 0: Preparation

This chapter describes the structure of FOP as it was in the first quarter of 2004. In the second quarter of that year the top level structure was strongly refactored. The new situation is not described here.

2.1. The preparation process

FOP's processing model is the SAX parsing model:

- Construct a suitable content handler object of type `org.xml.sax.ContentHandler`,
- create a parser of type `org.xml.sax.XMLReader`,
- register the content handler with the parser,
- call the parser's parse method on the input source.

From there on the parser takes control. For every XML event it passes control to the appropriate content handler methods. When these methods return, parsing continues until the next XML event is encountered. Once the parser has taken control, its content handler is the only entry point into FOP's data structures and methods.

The preparatory phase of FOP concerns itself with the construction of a suitable content handler object, whose methods allow FOP's process to handle the FO elements reported by the parser and build the required output document.

An application may choose to deal with the whole preparatory phase itself, and then call the parser's `parse` method.

The input source may be an FO file, which may be fed to the `parse` method as a string or as an `org.xml.sax.InputSource` object. Alternatively, a DOM document object may be used as input, e.g. the output of an XSLT processor. In that case:

- the parser should be of type `org.apache.fop.tools.DocumentReader`,
- the input source should be of type `org.apache.fop.tools.DocumentInputSource`, created with the DOM document as the argument of the constructor.

The classes `Fop` and `Driver` contain methods which applications may use as more convenient entry points into FOP.

The method `Fop.main` may be called with the input and output file names as arguments. This is mainly useful for starting Fop from the command line.

The class `Driver` contains a method `getContentHandler`, which can be used to create a suitable content handler. It also contains three `render` methods which are convenient entry points for applications.

These 4 methods may be invoked on a driver object which already has the following members: `userAgent`, `renderer`, `log`, `stream`. In addition, the driver object may have the following members: a `TreeBuilder` object having a member `userAgent`, and a `Document` object, which may have a member `layoutStrategy`. Using one's own `TreeBuilder` and `Document` objects allows one to customize FOP's behaviour in a major way.

The `render` methods invoke `getContentHandler` to create a suitable content handler. They register it as the content handler of the parser. They register the member `currentDocument` as a tree listener to the mem-

ber `foInputHandler`.

A suitable `org.xml.sax.ContentHandler` object for FOP processing is an object of type `org.apache.fop.fo.FOTreeBuilder` and has the following structure:

```
                  treeBuilder
                      |
       +--------------+------------------+
       |              |                  |
foInputHandler     userAgent       foTreeControl
       |                                 |
foTreeControl                        driver
foTreeListeners:                    usedFonts
   [foTreeControl]                   triplets
                                      fonts
                                  layoutStrategy -- foTreeControl
                                    areaTree -- atModel, foTreeControl
                                     atModel -- renderer
                                  foInputHandler
```

The `driver` and `renderer` objects are complex objects:

```
        driver             renderer
          |                    |
      treeBuilder        foTreeControl
        renderer           userAgent
     foInputHandler         fontList
        ostream             ostream
       userAgent
     foTreeControl
```

2.2. A detailed overview of the objects

```
 treeBuilder = {
    fobjTable: instance of java.util.HashMap(id=589)
    namespaces: instance of java.util.HashSet(id=590)
    currentFObj: null
    rootFObj: null
    foInputHandler: instance of org.apache.fop.fo.FOTreeHandler(id=591)
    userAgent: instance of org.apache.fop.apps.FOUserAgent(id=592)
    foTreeControl: instance of org.apache.fop.apps.Document(id=593)
    class$org$apache$fop$fo$ElementMapping: instance of java.lang.Class(reflected class=or
}
```

```
 treeBuilder.foInputHandler = {
    collectStatistics: true
    MEM_PROFILE_WITH_GC: false
    runtime: instance of java.lang.Runtime(id=595)
    pageCount: 0
    initialMemory: 0
    startTime: 0
    foTreeListeners: instance of java.util.HashSet(id=596)
    org.apache.fop.fo.FOInputHandler.foTreeControl: instance of org.apache.fop.apps.Docume
    org.apache.avalon.framework.logger.AbstractLogEnabled.m_logger: instance of org.apache
}
```

```
 treeBuilder.foTreeControl = "org.apache.fop.apps.Document@125844f"
 treeBuilder.foInputHandler.foTreeListeners = "[org.apache.fop.apps.Document@125844f]"
```

```
 treeBuilder.userAgent = {
    log: instance of org.apache.avalon.framework.logger.ConsoleLogger(id=597)
    defaults: instance of java.util.HashMap(id=601)
    handlers: instance of java.util.HashMap(id=602)
    baseURL: ""
    pdfEncryptionParams: null
    px2mm: 0.35277778
}


 treeBuilder.foTreeControl = {
    driver: instance of org.apache.fop.apps.Driver(id=587)
    usedFonts: instance of java.util.HashMap(id=604)
    triplets: instance of java.util.HashMap(id=605)
    fonts: instance of java.util.HashMap(id=606)
    layoutStrategy: instance of org.apache.fop.layoutmgr.LayoutManagerLS(id=607)
    areaTree: instance of org.apache.fop.area.AreaTree(id=608)
    atModel: instance of org.apache.fop.area.RenderPagesModel(id=609)
    bookmarks: null
    idReferences: instance of java.util.HashSet(id=610)
    foInputHandler: instance of org.apache.fop.fo.FOTreeHandler(id=591)
}


 treeBuilder.foTreeControl.driver = {
    NOT_SET: 0
    RENDER_PDF: 1
    RENDER_AWT: 2
    RENDER_MIF: 3
    RENDER_XML: 4
    RENDER_PRINT: 5
    RENDER_PCL: 6
    RENDER_PS: 7
    RENDER_TXT: 8
    RENDER_SVG: 9
    RENDER_RTF: 10
    treeBuilder: instance of org.apache.fop.fo.FOTreeBuilder(id=588)
    rendererType: 1
    renderer: instance of org.apache.fop.render.pdf.PDFRenderer(id=599)
    foInputHandler: instance of org.apache.fop.fo.FOTreeHandler(id=591)
    source: null
    stream: instance of java.io.BufferedOutputStream(id=600)
    reader: null
    log: instance of org.apache.avalon.framework.logger.ConsoleLogger(id=597)
    userAgent: instance of org.apache.fop.apps.FOUserAgent(id=592)
    currentDocument: instance of org.apache.fop.apps.Document(id=593)
}


 treeBuilder.foTreeControl.areaTree = {
    model: instance of org.apache.fop.area.RenderPagesModel(id=609)
    atControl: instance of org.apache.fop.apps.Document(id=593)
    idLocations: instance of java.util.HashMap(id=615)
    resolve: instance of java.util.HashMap(id=616)
    treeExtensions: instance of java.util.ArrayList(id=617)
}


 treeBuilder.foTreeControl.atModel = {
    renderer: instance of org.apache.fop.render.pdf.PDFRenderer(id=599)
    prepared: instance of java.util.ArrayList(id=618)
    pendingExt: instance of java.util.ArrayList(id=619)
    endDocExt: instance of java.util.ArrayList(id=620)
    org.apache.fop.area.StorePagesModel.pageSequence: null
```

```
    org.apache.fop.area.StorePagesModel.titles: instance of java.util.ArrayList(id=621)
    org.apache.fop.area.StorePagesModel.currSequence: null
    org.apache.fop.area.StorePagesModel.extensions: instance of java.util.ArrayList(id=622
}


 treeBuilder.foTreeControl.atModel.renderer = {
    MIME_TYPE: "application/pdf"
    pdfDoc: instance of org.apache.fop.pdf.PDFDocument(id=624)
    pages: null
    pageReferences: instance of java.util.HashMap(id=625)
    pvReferences: instance of java.util.HashMap(id=626)
    ostream: instance of java.io.BufferedOutputStream(id=600)
    pdfResources: null
    currentStream: null
    currentContext: null
    currentPage: null
    currentState: null
    currentFontName: ""
    currentFontSize: 0
    pageHeight: 0
    filterMap: null
    textOpen: false
    prevWordY: 0
    prevWordX: 0
    prevWordWidth: 0
    wordAreaPDF: instance of java.lang.StringBuffer(id=627)
    BPMarginOffset: 0
    IPMarginOffset: 0
    org.apache.fop.render.PrintRenderer.fontInfo: instance of org.apache.fop.apps.Document
    org.apache.fop.render.PrintRenderer.fontList: null
    org.apache.fop.render.AbstractRenderer.userAgent: instance of org.apache.fop.apps.FOUs
    org.apache.fop.render.AbstractRenderer.producer: "FOP 1.0dev"
    org.apache.fop.render.AbstractRenderer.creator: null
    org.apache.fop.render.AbstractRenderer.creationDate: null
    org.apache.fop.render.AbstractRenderer.options: instance of java.util.HashMap(id=629)
    org.apache.fop.render.AbstractRenderer.currentBPPosition: 0
    org.apache.fop.render.AbstractRenderer.currentIPPosition: 0
    org.apache.fop.render.AbstractRenderer.currentBlockIPPosition: 0
    org.apache.fop.render.AbstractRenderer.containingBPPosition: 0
    org.apache.fop.render.AbstractRenderer.containingIPPosition: 0
    org.apache.avalon.framework.logger.AbstractLogEnabled.m_logger: instance of org.apache
}


 treeBuilder.foTreeControl.layoutStrategy = {
    name: "layoutmgr"
    addLMVisitor: null
    org.apache.fop.layout.LayoutStrategy.name: "undefined"
    org.apache.fop.layout.LayoutStrategy.document: instance of org.apache.fop.apps.Documen
}


 treeBuilder.foTreeControl.atModel.renderer.ostream = {
    buf: instance of byte[512] (id=632)
    count: 15
    java.io.FilterOutputStream.out: instance of java.io.FileOutputStream(id=633)
}
```

For the members `fontList`, `fonts`, `usedFonts` and `triplets` of `treeBuilder.foTreeControl`, see under Fonts.

2.3. A detailed overview of the entry methods

Already created (e.g. in `Fop.main`): an object of type `Driver` with the members `userAgent`, `renderer`, `log`, `stream`.

To create `userAgent` one may use `Driver.getUserAgent`: if `driver` does not have `userAgent`, create a new `UserAgent`.

To create `renderer` one may use one of three methods:

- `setRenderer(int renderer)`
- `setRenderer(String rendererClassName)`
- `setRenderer(Renderer renderer)`

All three methods set the FOP version on the `renderer`, and register `userAgent` with it, which is obtained using `Driver.getUserAgent`.

`render(InputHandler inputHandler)`:

- creates `XMLReader parser`, `InputSource source`;
- calls `render(XMLReader parser, InputSource source)`.

`render(org.w3c.dom.Document document)`:

- creates `DocumentReader reader`, `DocumentInputSource source`;
- calls `render(XMLReader parser, InputSource source)`.

`render(XMLReader parser, InputSource source)`:

- creates content handler by calling `getContentHandler()`.
- registers the content handler with the parser.
- Adds `currentDocument` as a tree listener to `foInputHandler`.
- calls `parser.parse(source)`.

`getContentHandler()`:

- if driver does not have a `treeBuilder`, call `initialize()`: create a new `TreeBuilder`, set the `UserAgent` on it.
- if driver does not have a `currentDocument`, create a new `Document`.
- create a new `FOTreeHandler foInputHandler` using `currentDocument` as an argument (`currentDocument` is member `foTreeControl` in `foInputHandler`).
- create a new `AreaTree` using `currentDocument` as an argument, and register it with `currentDocument`.
- create a new `RenderPagesModel` using `renderer` as an argument, and register it with `currentDocument` and with `currentDocument.areaTree`.
- register `currentDocument` with the `renderer` (`currentDocument` is member `fontInfo` in `renderer`); setup `fontList` in `currentDocument`.

- start the `renderer` with the `outputstream`.
- register `foInputHandler` with `currentDocument`.
- if `currentDocument` does not have a `layoutStrategy`, create a new `LayoutStrategyLS` for it with `currentDocument` as an argument.
- register `userAgent`, `foInputHandler` and `currentDocument` with `treeBuilder` (`currentDocument` is member `foTreeControl` in `treeBuilder`).
- return `treeBuilder`.

Chapter 3

Phase 1: Building the FO tree

3.1. Creating the FO nodes

FOP's first task is building a suitable data structure from the XML input, which is an XML file with formatting objects or a result tree with formatting objects from an XSLT transformation. One could call this FOP's data binding. The data structure is an FO tree, i.e., a tree of FONode objects. The structure of the FO tree exactly parallels the structure of the XML file or the corresponding DOM tree, but instead of XML nodes its nodes are objects of type `org.apache.fop.fo.FONode`. The FO tree is built on the basis of SAX parser events. The parser is responsible for parsing the XML document; it calls FOP's callbacks when SAX events occur.

FOP's callbacks are implemented by the `FOTreeBuilder treebuilder` object, which is a SAX content handler. It was constructed in the preparation phase, and registered with the parser as the content handler. It has meaningful implementations of the methods `startDocument`, `endDocument`, `startElement`, `endElement`, and `characters`.

`treebuilder` delegates its `startDocument` and `endDocument` methods to its `FOTreeHandler` object `foInputHandler`. `FOTreeHandler` is a subclas of `FOInputHandler`.

```
 treebuilder.foInputHandler = {
    runtime: instance of java.lang.Runtime(id=635)
    pageCount: 0
    initialMemory: 0
    startTime: 0
    foTreeListeners: instance of java.util.HashSet(id=636)
    org.apache.fop.fo.FOInputHandler.foTreeControl: instance of org.apache.fop.apps.
    org.apache.avalon.framework.logger.AbstractLogEnabled.m_logger: instance of org.
}
```

The first important task of `treebuilder` is creating a suitable FO node for each element in the XML document. This is done in its `startElement` method. Its most important tool in this process is its `fobjTable` object. `fobjTable` is a map of maps. For each namespace supported by FOP it contains a map of local XML element name to a Node maker object `fobjMaker`, of type `ElementMapping.Maker`. In addition to the FO namespace it contains makers for FOP's extensions namespace, the SVG namespace and Batik's extensions namespace.

```
 treebuilder.fobjTable = "{
  http://www.w3.org/1999/XSL/Format={
   static-content=org.apache.fop.fo.FOElementMapping$SC@39e5b5,
   table=org.apache.fop.fo.FOElementMapping$Ta@117f31e,
   external-graphic=org.apache.fop.fo.FOElementMapping$EG@15a6029,
   table-column=org.apache.fop.fo.FOElementMapping$TC@5f6303,
   table-and-caption=org.apache.fop.fo.FOElementMapping$TAC@5d9084,
   table-footer=org.apache.fop.fo.FOElementMapping$TB@bad8a8,
   declarations=org.apache.fop.fo.FOElementMapping$Dec@e61fd1,
   wrapper=org.apache.fop.fo.FOElementMapping$W@331059,
   page-sequence=org.apache.fop.fo.FOElementMapping$PS@766a24,
   single-page-master-reference=org.apache.fop.fo.FOElementMapping$SPMR@32784a,
   footnote=org.apache.fop.fo.FOElementMapping$Foot@1774b9b,
   multi-switch=org.apache.fop.fo.FOElementMapping$MS@104c575,
   bidi-override=org.apache.fop.fo.FOElementMapping$BO@3fa5ac,
   layout-master-set=org.apache.fop.fo.FOElementMapping$LMS@95cfbe,
   float=org.apache.fop.fo.FOElementMapping$F@179dce4,
   list-item=org.apache.fop.fo.FOElementMapping$LI@1950198,
   basic-link=org.apache.fop.fo.FOElementMapping$BL@19bb25a,
   multi-property-set=org.apache.fop.fo.FOElementMapping$MPS@da6bf4,
   table-row=org.apache.fop.fo.FOElementMapping$TR@1e58cb8,
```

```
   region-end=org.apache.fop.fo.FOElementMapping$RE@179935d,
   block=org.apache.fop.fo.FOElementMapping$B@b9e45a,
   leader=org.apache.fop.fo.FOElementMapping$L@3ef810,
   table-header=org.apache.fop.fo.FOElementMapping$TB@100363,
   list-item-body=org.apache.fop.fo.FOElementMapping$LIB@14e8cee,
   multi-properties=org.apache.fop.fo.FOElementMapping$MP@67064,
   region-after=org.apache.fop.fo.FOElementMapping$RA@bcda2d,
   multi-case=org.apache.fop.fo.FOElementMapping$MC@97d01f,
   block-container=org.apache.fop.fo.FOElementMapping$BC@e0a386,
   title=org.apache.fop.fo.FOElementMapping$T@feb48,
   retrieve-marker=org.apache.fop.fo.FOElementMapping$RM@11ff436,
   color-profile=org.apache.fop.fo.FOElementMapping$CP@da3a1e,
   character=org.apache.fop.fo.FOElementMapping$Ch@11dba45,
   simple-page-master=org.apache.fop.fo.FOElementMapping$SPM@b03be0,
   page-sequence-master=org.apache.fop.fo.FOElementMapping$PSM@2af081,
   footnote-body=org.apache.fop.fo.FOElementMapping$FB@113a53d,
   marker=org.apache.fop.fo.FOElementMapping$M@c5495e,
   table-body=org.apache.fop.fo.FOElementMapping$TB@53fb57,
   inline=org.apache.fop.fo.FOElementMapping$In@19a32e0,
   table-cell=org.apache.fop.fo.FOElementMapping$TCell@8238f4,
   list-block=org.apache.fop.fo.FOElementMapping$LB@16925b0,
   region-start=org.apache.fop.fo.FOElementMapping$RS@297ffb,
   table-caption=org.apache.fop.fo.FOElementMapping$TCaption@914f6a,
   conditional-page-master-reference=org.apache.fop.fo.FOElementMapping$CPMR@1f4cbee,
   list-item-label=org.apache.fop.fo.FOElementMapping$LIL@787d6a,
   multi-toggle=org.apache.fop.fo.FOElementMapping$MT@71dc3d,
   initial-property-set=org.apache.fop.fo.FOElementMapping$IPS@1326484,
   repeatable-page-master-alternatives=org.apache.fop.fo.FOElementMapping$RPMA@16546ef,
   repeatable-page-master-reference=org.apache.fop.fo.FOElementMapping$RPMR@1428ea,
   flow=org.apache.fop.fo.FOElementMapping$Fl@18a49e0,
   page-number=org.apache.fop.fo.FOElementMapping$PN@1f82982,
   instream-foreign-object=org.apache.fop.fo.FOElementMapping$IFO@16d2633,
   inline-container=org.apache.fop.fo.FOElementMapping$IC@e70e30,
   root=org.apache.fop.fo.FOElementMapping$R@154864a,
   region-before=org.apache.fop.fo.FOElementMapping$RBefore@3c9217,
   region-body=org.apache.fop.fo.FOElementMapping$RB@9b42e6,
   page-number-citation=org.apache.fop.fo.FOElementMapping$PNC@14520eb
  },
  http://xml.apache.org/fop/extensions={
   bookmarks=org.apache.fop.fo.extensions.ExtensionElementMapping$B@1d7fbfb,
   label=org.apache.fop.fo.extensions.ExtensionElementMapping$L@e020c9,
   outline=org.apache.fop.fo.extensions.ExtensionElementMapping$O@888e6c
  },
  http://www.w3.org/2000/svg={
        <default>=org.apache.fop.fo.extensions.svg.SVGElementMapping$SVGMaker@1742700,
   svg=org.apache.fop.fo.extensions.svg.SVGElementMapping$SE@acb158
  },
 http://xml.apache.org/batik/ext={
   <default>=org.apache.fop.fo.extensions.svg.BatikExtensionElementMapping$SVGMaker@1af33d
   batik=org.apache.fop.fo.extensions.svg.BatikExtensionElementMapping$SE@17431b9
  }
 }"
```

The values in this map are objects of subclasses of `ElementMapping.Maker`. `ElementMapping.Maker` is a static nested class of `ElementMapping`. It has no members and a single object method `FONode make(FONode parent)`. The subclasses are static nested classes of `FOElementMapping`. Each subclass has its own implementation of the `make` method, and returns its own subclass of FONode. For example, `FOElementMapping$R` returns a `org.apache.fop.fo.pagination.Root` object.

`treebuilder` delegates its `endElement` method to the node's `end` method, which allows FOP to take appropriate action at the end of each FO element. The only node type whose `end` method takes special action, is `org.apache.fop.fo.pagination.PageSequence`. It hands control to FOP's next phase, building of the area tree.

3.2. Creating the property values

Formatting objects have many attributes by which the user may finetune their behaviour. When the FO tree is built, the attributes must be converted to properties. This conversion process must implement XSLT's sometimes complicated rules of default values, inheritance, shorthand notations etc. This is one of the tasks of the property subsystem, which is described in its own chapter.

Chapter 4

Phase 2: Building the Area tree

4.1. Initiating the layout process

In the `PageSequence.endOfNode()` method, the `AreaTreeHandler` object `foEventHandler`'s method `endPageSequence` is called. This method constructs a `PageSequenceLayoutManager` for the `PageSequence` FO node, which manages all page-related layout.

```
org.apache.fop.fo.pagination.PageSequence.endOfNode():
this.getFOEventHandler().getFOEventHandler().endPageSequence(this):
-> foTreeBuilder.getFOEventHandler().endPageSequence(this)
-> foEventHandler.endPageSequence(this) (type AreaTreeHandler):
```

This method creates a new `PageSequenceLayoutManager` for the `PageSequence` FO node. The pages in this page sequence are completely layed out by the `PageSequenceLayoutManager`, in its `activateLayout` method. The first step in the layout process is getting the page setup from the page masters. Then the FO tree is processed.

```
  [1] org.apache.fop.layoutmgr.PageSequenceLayoutManager.activateLayout (PageLayoutM
  [2] org.apache.fop.fo.area.AreaTreeHandler.endPageSequence (AreaTreeHandler.java:2
  [3] org.apache.fop.fo.pagination.PageSequence.endOfNode (PageSequence.java:202)
  [4] org.apache.fop.fo.FOTreeBuilder.endElement (FOTreeBuilder.java:292)
  ... parser stuff
  [13] org.apache.xerces.parsers.AbstractSAXParser.parse (null)
  [14] org.apache.xalan.transformer.TransformerIdentityImpl.transform (null)
  [15] org.apache.fop.apps.InputHandler.render (InputHandler.java:120)
  [16] org.apache.fop.apps.Fop.main (Fop.java:102)
```

```
main[1] dump pageSLM
 pageSLM = {
    areaTreeHandler= org.apache.fop.area.AreaTreeHandler  (id=79)
    areaTreeModel= org.apache.fop.area.RenderPagesModel  (id=81)
    bFinished= false
    bFirstPage= false
    bInited= false
    childLMiter= org.apache.fop.layoutmgr.LMiter  (id=85)
    childLMs= java.util.ArrayList  (id=89)
    curBody= null
    curChildLM= null
    curFlow= null
    curPage= null
    currentSimplePageMaster= null
    curSpan= null
    curSpanColumns= 0
    flowBPD= 0
    flowIPD= 0
    fobj= org.apache.fop.fo.pagination.PageSequence  (id=12)
    fobjIter= java.util.AbstractList$ListItr  (id=90)
    isFirstPage= true
    markers= null
    pageCount= 1
    pageNumberGenerator= null
    pageNumberString= null
    parentLM= null
    staticContentLMs= java.util.HashMap  (id=93)
}
```

The above calling sequence contains one configuration point. FOP's area tree building process can be modified by registering a different LayoutManagerMaker with the Area tree handler. The LayoutManagerMaker controls the creation of Layout Managers.

Warning

TO BE EXPANDED

4.2. Creating the page and body areas

4.2.1. Overview

Create the layout (`activateLayout`)

- First create a new Page Viewport (`makeNewPage`).
  - First finish the current page (`finishPage`).
  - Then create the new page viewport (`createPage`).
    - First get the page master (`getSimplePageMasterToUse`, `pageSequence.getPageSequenceMaster` or `pageSequence.getSimplePageMaster`).
    - Then get the body (`currentSimplePageMaster.getRegion`, from `currentSimplePageMaster`'s `regions` map).
    - Then create the page viewport (`createPageAreas(currentSimplePageMaster)`).
      - From the properties of the page master create the page reference rectangle, a new page, a new FODimension object, and a CTM object.
      - For each region in the page master (in our example we only have a body):
        - make a region viewport (`makeRegionViewport`), which involves calculating the position of the region on the page, using the FODimension and CTM objects.
        - make the reference area (`makeRegionBodyReferenceArea`, `makeRegionReferenceArea`).

At this point the page viewport and its region viewports have been laid out.

- Then create the body's main reference area (`createBodyMainReferenceArea`).
- Then create a Span (`createSpan`).
- And get the flowIPD (curFlow.`getIPD`()).

At this point the body has a single span area with a single flow area without children.

4.2.2. Detailed view

The call stack when creating the region viewports:

```
[1] org.apache.fop.layoutmgr.PageLayoutManager.makeRegionViewport (PageLayoutManager.java
[2] org.apache.fop.layoutmgr.PageLayoutManager.createPageAreas (PageLayoutManager.java:8
[3] org.apache.fop.layoutmgr.PageLayoutManager.createPage (PageLayoutManager.java:748)
[4] org.apache.fop.layoutmgr.PageLayoutManager.makeNewPage (PageLayoutManager.java:467)
[5] org.apache.fop.layoutmgr.PageLayoutManager.activateLayout (PageLayoutManager.java:22
```

At the end of `createPageAreas` the following properties of the page have been established:

The page reference rectangle:

```
pageRefRect = {
   x: 56692
   y: 56692
```

```
    width: 481891
    height: 728505
    serialVersionUID: -4345857070255674764
}
```

The page reference area:

```
 page = {
    regionBefore: null
    regionStart: null
    regionBody: instance of org.apache.fop.area.RegionViewport(id=1279)
    regionEnd: null
    regionAfter: null
    unresolved: null
}
```

```
 page.regionBody = {
    region: instance of org.apache.fop.area.BodyRegion(id=1280)
    viewArea: instance of java.awt.Rectangle(id=1281)
    clip: false
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 0
    org.apache.fop.area.Area.props: null
}
```

```
 page.regionBody.region = {
    beforeFloat: null
    mainReference: null
    footnote: null
    columnGap: 18000
    columnCount: 1
    refIPD: 0
    org.apache.fop.area.RegionReference.regionClass: 2
    org.apache.fop.area.RegionReference.ctm: instance of org.apache.fop.area.CTM(id=
    org.apache.fop.area.RegionReference.blocks: instance of java.util.ArrayList(id=1
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 0
    org.apache.fop.area.Area.props: null
}
```

```
 page.regionBody.viewArea = {
    x: 56692
    y: 56692
    width: 481891
    height: 728505
    serialVersionUID: -4345857070255674764
}
```

The `PageViewport` is returned:

```
 curPage = {
    page: instance of org.apache.fop.area.Page(id=1261)
    viewArea: instance of java.awt.Rectangle(id=1289)
    clip: false
    pageNumber: null
    idReferences: null
    unresolved: null
    pendingResolved: null
    markerFirstStart: null
```

```
    markerLastStart: null
    markerFirstAny: null
    markerLastEnd: null
    markerLastAny: null
}
```

When `makeNewPage` returns, the Page LayoutManager has a Page Viewport and a Body Region object. The layout dimensions have been calculated:

```
 this = {
    pageNumberGenerator: instance of org.apache.fop.fo.pagination.PageNumberGenerator(id=1
    pageCount: 1
    pageNumberString: "1"
    isFirstPage: false
    bFirstPage: false
    curPage: instance of org.apache.fop.area.PageViewport(id=1288)
    curBody: instance of org.apache.fop.area.BodyRegion(id=1280)
    curSpan: null
    curSpanColumns: 0
    curFlow: null
    flowBPD: 728505
    flowIPD: 0
    areaTree: instance of org.apache.fop.area.AreaTree(id=1005)
    pageSequence: instance of org.apache.fop.fo.pagination.PageSequence(id=1006)
    currentSimplePageMaster: instance of org.apache.fop.fo.pagination.SimplePageMaster(id=
    staticContentLMs: instance of java.util.HashMap(id=1008)
    lmls: instance of org.apache.fop.layoutmgr.LayoutManagerLS(id=1009)
    org.apache.fop.layoutmgr.AbstractLayoutManager.userAgent: instance of org.apache.fop.a
    org.apache.fop.layoutmgr.AbstractLayoutManager.parentLM: null
    org.apache.fop.layoutmgr.AbstractLayoutManager.fobj: instance of org.apache.fop.fo.pag
    org.apache.fop.layoutmgr.AbstractLayoutManager.foID: null
    org.apache.fop.layoutmgr.AbstractLayoutManager.markers: null
    org.apache.fop.layoutmgr.AbstractLayoutManager.bFinished: false
    org.apache.fop.layoutmgr.AbstractLayoutManager.curChildLM: null
    org.apache.fop.layoutmgr.AbstractLayoutManager.childLMiter: instance of org.apache.fop
    org.apache.fop.layoutmgr.AbstractLayoutManager.bInited: false
}
```

The method `createBodyMainReferenceArea()` adds a `MainReferenceArea` to the body region:

```
 curBody = {
    beforeFloat: null
    mainReference: instance of org.apache.fop.area.MainReference(id=1293)
    footnote: null
    columnGap: 18000
    columnCount: 1
    refIPD: 0
    org.apache.fop.area.RegionReference.regionClass: 2
    org.apache.fop.area.RegionReference.ctm: instance of org.apache.fop.area.CTM(id=1282)
    org.apache.fop.area.RegionReference.blocks: instance of java.util.ArrayList(id=1283)
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 0
    org.apache.fop.area.Area.props: null
}


 curBody.mainReference = {
    spanAreas: instance of java.util.ArrayList(id=1294)
    columnGap: 0
    width: 0
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 0
    org.apache.fop.area.Area.props: null
}
```

```
 curBody.mainReference.spanAreas = "[]"
```

After `createSpan(1)`:

```
 curBody.mainReference.spanAreas = "[org.apache.fop.area.Span@1581e80]"


 curBody.mainReference.spanAreas.get(0) = {
    flowAreas: instance of java.util.ArrayList(id=1299)
    height: 0
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 481891
    org.apache.fop.area.Area.props: null
}
```

Chapter 5

Phase 2a: The `getNextBreakPoss` call tree

5.1. Overview

- Create a new layout context for the children.
- Then process the flow: loop until the flow is exhausted (isFinished()):
  - Get the next possible breakpoint (getNextBreakPoss).
    - Loop until the list of child layout managers is exhausted:
      - Get a child layout manager (AbstractLayoutManager.getChildLM). The current child layout manager is returned until it is finished. Then the layout manager for the next child is returned.
      - Create a new layout context for the children.
      - If the child layout manager is not finished, get the next possible breakpoint (getNextBreakPoss).
      - If a breakpoint is returned, break the loop and return the breakpoint.
  - This finishes a page.
  - Get the next possible breakpoint (getNextBreakPoss) (*continued*)
    - Loop until the list of child layout managers is exhausted: (*continued*)
      - Else if no breakpoint is returned, do the next cycle with the next child layout manager.
    - Mark the layout manager as finished (the list of child layout managers is exhausted).

At this point a complete (pseudo)tree of possible break points for a page has been collected.

5.2. How do layout managers get layout managers for the child FO nodes?

Child layout managers are created and retrieved in the method `AbstractLayoutManager.getChildLM`.

The layout manager gets the layout manager for its next child from its `LMIter` object `childLMIter`. This `LMIter` object contains an iterator over the children of the layout manager's FO node. It behaves itself as an iterator over the list of layout managers for the children. It constructs those layout managers when needed, in its `preLoadNext` method, which calls the convenience method `preLoadList`. Using the iterator `fobjIter`, `preLoadList` iterates through the children of the current FO-Object and preloads the corresponding LMs (`LayoutManagerMaker.makeLayoutManagers()`). Thus `LMIter` has a layout manager for its next child. It returns this layout manager in the call to its `next()` method, when the current layout manager invokes its `getChildLM` method.

`BlockLayoutManager.ProxyLMiter` has its own subclass of `LMIter`, which implements a different method of creating child layout managers. See the next section.

The method that creates the LMs, `LayoutManagerMaker.makeLayoutManagers()`, is implemented by FOP's implementation of the `LayoutManagerMaker` interface, `LayoutManagerMapping`. A different LM system can be hooked into FOP by inserting a different implementation of the `LayoutManagerMaker` interface. (See `FOUserAgent.setLayoutManagerMakerOverride`.)

Also note that `AbstractLayoutManager.getChildLM` itself does not behave as an iterator. The current child layout manager is returned until it is finished. One can safely make multiple calls to `getChildLM`. If the current child layout manager is unfinished and does nothing in between the calls, it remains unfinished, and is returned at every call. If the current child layout manager is finished, the next layout manager is loaded, and, because it is unfinished, returned at every call. If this is the last child layout manager and it is finished, then null is returned because in `LMiter.preLoadNext baseIter.hasNext()` returns false. The latter case is used in `BlockLayoutManager.getNextBreakPoss`.

Stack trace: Creating a new layout manager for a child, in LMiter.preLoadNext, in AbstractLayoutManager.getChildLM:

```
[1] org.apache.fop.layoutmgr.LayoutManagerMapping.makeLayoutManagers(org.apache.fop.fo.F
[2] org.apache.fop.layoutmgr.PageSequenceLayoutManager(org.apache.fop.layoutmgr.Abstract
[3] org.apache.fop.layoutmgr.PageSequenceLayoutManager(org.apache.fop.layoutmgr.Abstract
[4] org.apache.fop.layoutmgr.LMiter.hasNext() line: 39
[5] org.apache.fop.layoutmgr.PageSequenceLayoutManager(org.apache.fop.layoutmgr.Abstract
[6] org.apache.fop.layoutmgr.PageSequenceLayoutManager.getNextBreakPoss(org.apache.fop.l
[7] org.apache.fop.layoutmgr.PageSequenceLayoutManager.activateLayout() line: 232
```

5.3. Block layout managers and their child layout managers

Block LMs are different in their treatment of their child LMs. For this purpose BlockLayoutManager defines a nested class ProxyLMiter, which is a subclass of LMiter.

This proxy is the basic LMiter over the children of the block. If the proxy produces a child LM that does not generate inline areas, the child LM is added to the list of child LMs as normal. But if the childLM generates an inline area, a new LineLayoutManager object is created (BlockLayoutManager.ProxyLMiter.createLineManager). This LM asks the proxy to produce more child LMs. As long as these child LMs generate inline areas, they are collected by the LineLayoutManager object. Finally, the LineLayoutManager object creates its LMiter object as the ListIterator over the list of collected child LMs.

5.4. About getNextBreakPoss and the list of child layout managers

Note that the breakpoint may come from a deeply nested child. Each layout manager keeps a reference to its current child layout manager. The whole list is descended again (getChildLM) at the next call to getNextBreakPoss.

Warning

TO BE IMPROVED

Stack of layout managers:

```
PageSequence  PageLayoutManager

Flow          FlowLayoutManager

Block         BlockLayoutManager

Block         LineLayoutManager

FOText        TextLayoutManager
```

For BlockLayoutManager and LineLayoutManager Block is the same, but their childLMIter are different: BlockLayoutManager$BlockLMiter vs AbstractList$ListItr

```
[1] org.apache.fop.layoutmgr.TextLayoutManager.getNextBreakPoss (TextLayoutManager.java:
[2] org.apache.fop.layoutmgr.LineLayoutManager.getNextBreakPoss (LineLayoutManager.java:
[3] org.apache.fop.layoutmgr.BlockLayoutManager.getNextBreakPoss (BlockLayoutManager.jav
[4] org.apache.fop.layoutmgr.FlowLayoutManager.getNextBreakPoss (FlowLayoutManager.java:
[5] org.apache.fop.layoutmgr.PageLayoutManager.getNextBreakPoss (PageLayoutManager.java:
```

```
  [6] org.apache.fop.layoutmgr.PageLayoutManager.activateLayout (PageLayoutManager.j
```

A `TextLayoutManager`:

```
 this = {
    vecAreaInfo: instance of java.util.ArrayList(id=1062)
    chars: instance of char[13] (id=1064)
    textInfo: instance of org.apache.fop.fo.TextInfo(id=1065)
    iAreaStart: 0
    iNextStart: 0
    ipdTotal: null
    spaceCharIPD: 4448
    hyphIPD: 5328
    halfWS: instance of org.apache.fop.traits.SpaceVal(id=1066)
    iNbSpacesPending: 0
    org.apache.fop.layoutmgr.AbstractLayoutManager.userAgent: instance of org.apache
    org.apache.fop.layoutmgr.AbstractLayoutManager.parentLM: instance of org.apache.
    org.apache.fop.layoutmgr.AbstractLayoutManager.fobj: instance of org.apache.fop.
    org.apache.fop.layoutmgr.AbstractLayoutManager.foID: null
    org.apache.fop.layoutmgr.AbstractLayoutManager.markers: null
    org.apache.fop.layoutmgr.AbstractLayoutManager.bFinished: false
    org.apache.fop.layoutmgr.AbstractLayoutManager.curChildLM: null
    org.apache.fop.layoutmgr.AbstractLayoutManager.childLMiter: instance of org.apac
    org.apache.fop.layoutmgr.AbstractLayoutManager.bInited: true
}
```

Text in `fo:text` is handled by a `TextLayoutManager`. Two routines add the text and calculate the next possible break.

5.5. `LineLayoutManager.getNextBreakPoss`

5.5.1. Prepare for the main loop

- Create a new empty list of possible line endings, `vecPossEnd`.
- Retrieve the `ipd` `availIPD` from the layout context.
- Create a new layout context (inline layout context) for the child layout managers, based on the layout context for this layout manager.
- Clear the map of previous `ipd`s.
- Record the length of `vecInlineBreaks`, which we can use to find the last breakposs of the previous line.
- Set `prevBP` to `null`; `prevBP` contains the last confirmed breakposs of this line.

5.5.2. The main loop over the list of child layout managers

Loop until the list of child layout managers is exhausted:

- Get a child layout manager (`AbstractLayoutManager.getChildLM`). The current child layout manager is returned until it is finished. Then the layout manager for the next child is returned.
- Record the last breakposs.
- Record whether the breakposs we are going to find is the first breakposs of this line.
- Record whether it is the first breakposs of this child layout manager.
- Initialize the inline layout context (note that it is not a new layout context, the same inline layout context is used by all child layout managers) (method `InlineStackingLayout.initChildLC`):
  - Record whether this is a new area; it is a new area if this is the start of a new line or of a new child LM.
  - If this is the start of a new line
    - record whether this is the first area of this child LM,
    - set the leading space as passed by argument.
  - Else if this starts a new child LM

        - record that this is the first area,
        - set the leading space from the previous BP.
    - Else set the leading space to `null`.
- Record on the inline layout context whether leading space is supppressed; it is suppressed if this is the start of a new line, but not the start of this child LM, and the previous line was not ended by a forced break.
- Retrieve the next breakposs from the current child LM (`getNextBreakPoss` method of child LM). If it is not `null`:
    - Calculate the `ipd` up to the previous BP (method `InlineStackingLayout.updatePrevIPD`):
        - Take an empty `ipd` size.
        - If this starts a new line:
            - if it has a leading fence, add leading space (?),
            - list the `ipd` for the LM of this BP in the map of previous `ipd`s.
        - Else
            - retrieve the `ipd` for the LM of this BP in the map of previous `ipd`s,
            - if that is `null` (first BP of this child LM)
                - retrieve the `ipd` for the LM of the previous BP in the map of previous `ipd`s,
                - add the leading space of this BP,
                - add the pending space-end (stacking size) of the previous BP,
                - list the `ipd` for the LM of this BP in the map of previous `ipd`s.
    - Add to the `ipd` the pending space-end (stacking size) of this BP.
    - Record whether this BP could end the line:
        - if a break may occur after this BP, record true;
        - else if this BP is suppressible at a line break, return false;
        - else, return whether this is the last child LM and it is finished, or the next area could start a new line.
    - If this BP could end the line, add trailing space.
    - If this BP exceeds the line length (`bpDim.min > availIPD.max`),
        - if the text should be justified or if this is the first BP of this line,
            - if we are in a hyphenation try, break the loop; we have exhausted our options and one of the previous BPs should end the line (`_exit of loop_`);
            - if this BP could not end the line, add it to the list of inline breaks, and continue with the next iteration;
            - prepare to hyphenate: get the hyphenation context for the text between the last and this BP (method `getHyphenContext`):
                - add this BP to the list of inline breaks; even though this is not a good BP, we add it to the list, so that we can retrieve the text between the last and this BP;
                - iterate back to the previous BP in this list;
                - loop over the following BPs in this list:
                    - retrieve the text between the preceding and this BP.
                - remove this BP again from the list of inline breaks.
                - create a hyphenation object for the retrieved text, taking the language, country and other hyphenation properties into account.
                - create a hyphenation context object from it, and return that.
            - store the hyphenation context with the inline layout context.
            - Record on the inline layout context that we are in a hyphenation try.
            - reset the child LMs to the previous BP or to the start of the line.
        - Else (if text should not be justified and if this is not the first BP of this line) break the loop; one of the previous BPs should end the line (`_exit of loop_`);
    - Else (if this BP does not exceed the line length):
        - add this BP to the list of inline breaks,
        - if this BP could end the line,
            - record it as the last confirmed BP: set prevBP to this BP.
            - if this BP is a forced line break, break the loop; this BP (or one of the previous BPs?) should end the line (`_exit of loop_`).
            - if this BP may fill the line length (`bpDim.max >= availIPD.min`), add it to the list of possible line endings, `vecPossEnd`, with a cost which is equal to the difference of the optimal values of the content length and the line length (`Math.abs(availIPD.opt - bpDim.opt)`).
- If we are in a hypenation try, and the hyphenation context has no more hyphenation points, break the loop; this or one of the previous BPs should end the line (`_exit of loop_`).

5.5.3. After the main loop

There are five exit points of the main loop:

1. The last BP in the hyphenation try has exceeded the line length.
2. The last BP has exceeded the line length, and we cannot get a hyphenation context.
3. The last BP has exceeded the line length, and we do not hyphenate.
4. The last BP has not exceeded the line length but forces a line break.
5. We have run out of hyphenation points (and the last BP has not exceeded the line length).
6. Natural end of the while loop: we are through the list of child layout managers.

If the last BP has exceeded the line length, it is not in the list of inline breaks, and prevBP points to the last good break; otherwise it is in the list of inline breaks, and prevBP points to it.

- If we are through the list of child LMs, mark this LM as finished.
- If no BP was produced, return `null`. (This should concur with being finished?)
- If prevBP is `null`, there is not last good break; set it to this BP, even though it fails some criteria:
  - it has exceeded the line length in the hyphenation try or we cannot get a hyphenation context,
  - or it cannot end the line but it is the last BP of the last child layout manager.
- If this BP is not a forced break, and there are several possible line breaks, select the best one; make `prevBP` point to it. (Could this produce the wrong result if the BP has exceeded the line length and at the same time is a forced line break? Should `prevBP` be tested for being a forced break instead?)
- If the last BP is not the actual breakpoint `prevBP` (`bp != prevBP`) and the material after `prevBP` is not suppressible at the end of a line, back up to `prevBP` for a proper start of the next line.
- If the text should be justified and the breakpoint is a forced line break (here `prevBP` is tested) or this is the last line of the layout manager, set text alignment to left-aligned.
- Make a line break and return the associated breakposs.

5.6. `LineLayoutManager.makeLineBreak`

Arguments are:

- index in `vecInlineBreaks` of line breaking BP of last line,
- target line length,
- type of text alignment.

Calculate line dimensions. The Line LayoutManager contains the parameters `lineHeight`, `lead` and `follow` as members, which it received from its Block LayoutManager parent at construction. The Blo ckLayoutManager contains these parameters as members as well, and has received them from a `TextInfo` object in the method set `BlockTextInfo`. The `TextInfo` object has a reference to the `Font` object. `lead` is the font ascender, `follow` is the font descender.

The leading is the difference between `lineHeight` and font height = ascender + descender. The leading is split in two halves, one for above, the other for below the line. The variable `lineLead` is calculated as the distance from the baseline to the top of the line, the variable `maxtb` is calculated as the distance from the baseline to the bottom of the line. The variable `middlefollow` set equal to `maxtb`. These parameters correspond to the members `lead`, `total` and `follow` of the breakposs.

Warning

Find out the exact meaning of these.

- Loop over the breakposs in `vecInlineBreaks`:
  - adjust `lineLead`, `maxtb` and `middlefollow` if the corresponding dimension of the BP is larger;
  - add the `ipd`s; a BP does not just hold the `ipd` of the area since the previous BP, but the cumulative `ipd`

for all the areas contributed by its layout manager; therefore care is taken to add only the ipd of the last BP of each LM; more precisely, the ipd of the last BP of the previous LM is added when the next LM is encountered;
- Add the ipd of the last BP.
- Resolve the trailing space of the last BP.
- Adjust middlefollow if it is smaller than maxtb - lineHeight.
- Calculate the stretch or shrink factor ipdAdjust for the stretch- and shrinkable elements in this line:
  - if content optimum is larger than line length optimum,
    - if content minimum is smaller than line length optimum,
      - calculate ipdAdjust between 0 and -1, real line length = line length optimum,
    - else (content minimum is larger than line length optimum)
      - ipdAdjust = -1, real line length = content minimum,
  - else (content optimum is smaller than line length optimum),
    - if content maximum is larger than line length optimum,
      - calculate ipdAdjust between 0 and 1, real line length = line length optimum,
    - else (content maximum is smaller than line length optimum),
      - ipdAdjust = 1, real line length = content maximum.
- Calculate the stretch or shrink factor dAdjust for white space and the indent:
  - justify: calculate dAdjust,
  - center or end: calculate the required indent.
- Create a new LineBreakPosition based on this LM, the last BP in vecInlineBreaks and the calculated dimensions; use lineLead for the baseline position and lineLead + middlefollow for the lineHeight.
- Create a BreakPoss from this LineBreakPosition object.
- Mark the BP as last if this LM is finished.
- Set the stacking size of the BP (bpd) equal to lineLead + middlefollow.

5.7. Line LayoutManager, a sequence of breakposs

The text is: "waterstaatsingenieur ministersportefeuille aandachtstrekker. Vernederlandste vakliteratuur" etc. It consists of a few long Dutch words, which can be found in the hyphenation exceptions in nl.xml. The column width is 8cm. This text and width have been chosen so that they force many hyphenations.

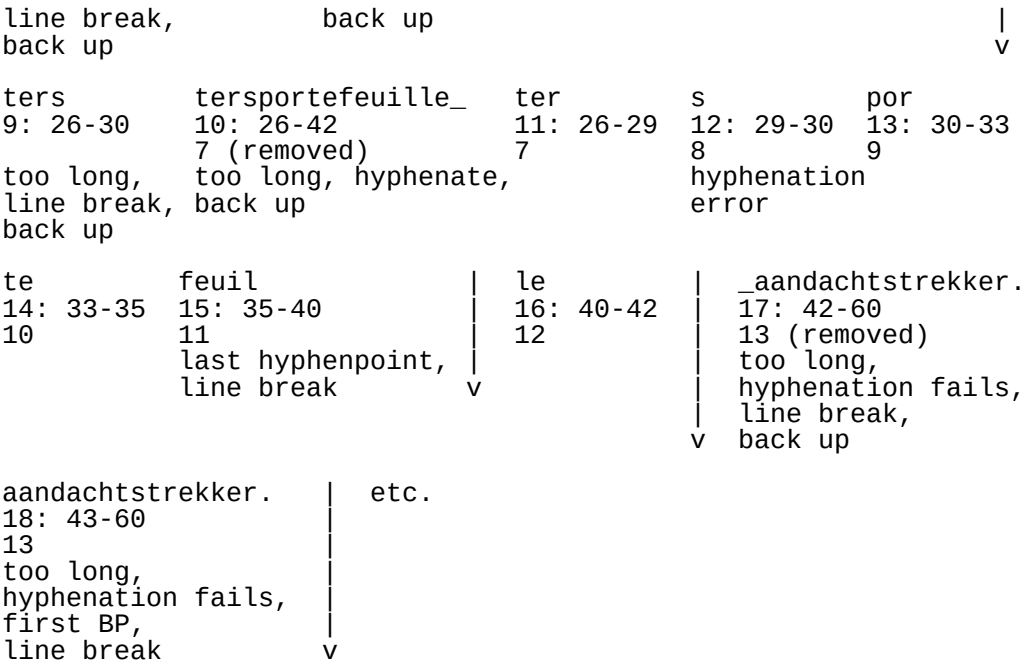
This text is contained in a single FOText node, and is dealt with by a single Text LayoutManager, which is a child layout manager for the Line LayoutManager. The Text LayoutManager maintains a list of area information objects, vecAreaInfo, and the Line LayoutManager maintains a list of breakposs, vecInlineBreaks. Each breakposs refers to an area, in its position.iLeafPos member.

During the process of finding suitable line breaks, breakposs are generated until one is found that exceeds the line length. Then the Line LayoutManager backs up to the previous BP, either to start a hyphenation try or to create the line break. When it creates a line break, the last breakposs, which exceeds the line length, is not added to the list of inline breaks. When it starts a hyphenation try, the breakposs is added to that list, and removed again after the text for hyphenation has been obtained. Each time, the corresponding area info is left in the list of area information objects. As a consequence the list vecAreaInfo is longer than the list vecInlineBreaks, and some areas have no corresponding breakposs.

```
wa-ter-staats-in-ge-nieur mi-nis-ters-por-te-feuil-le aandachtstrekker.
0  2   5      1  1  1    2   2   2  2 3   3  3     4 4                 6
             1  3  5    0   3   6  9 0   3  5     0 2                 0

text:         waterstaatsingenieur  wa      ter     staats   in       |
AreaInfo:     0: 0-20 (removed)     0: 0-2  1: 2-5  2: 5-11  3: 11-13 |
InlineBreaks: 0 (removed)           0       1       2        3        |
              too long, hyphenate,                                     |
              back up                                                 V

gen       genieur   _ministersportefeuille _mi      nis       |
4: 13-15  5: 13-20  6: 20-42               7: 20-23 8: 23-26  |
          4         5 (removed)            5        6         |
too long,           too long, hyphenate                      |
```



A few remarkable points:

1. The first AreaInfo object is removed from the list during backing up. This is because the previous BP, which is `null`, does not belong to this child layout manager, and therefore the whole child LM is reset.

2. The last BP, no. 18, exceeds the line length, but because it is the first BP of this line, it is accepted for the line break.

3. BP 7 at position 29 falls at a point that is not a good hyphenation point. This is probably caused by the fact that for this line only the partial word is subjected to hyphenation. On the previous line the complete word was subjected to hyphenation, and there was no BP at position 29.

4. For the word "aandachtstrekker." hyphenation fails (`hyph == null` is returned). This may be due to the period not being separated from the word. When the period is removed, the word is properly broken.

5.8. `TextLayoutManager.getNextBreakPoss`

- If this is the first call to this Text LayoutManager, initialize `ipdTotal` and record that this breakposs is the first in `iFlags`.
- If leading spaces must be suppressed, suppress all leading space characters `U+20`. Return if this finished the text.
- For the remaining leading space characters,
  - If this is a space `U+20` or a non-breaking space `U+A0`,
    - count it;
    - if this is the first character and this is the first breakposs,
      - if the context has leading spaces, add it (or `halfWS`?);
      - else add it (or `halfWS`?) to the pending space, and add the pending space to the `space` ipd.
    - add the space width to the word `ipd`.
    - set the pending space to `halfWS`.
    - if this is a non-breaking space `U+A0`, register it in `bSawNonSuppressible`.
  - Else (other space characters),
    - register it in `bSawNonSuppressible`.
    - add the pending space to the space `ipd`, and clear the pending space.
    - add the character width to the word `ipd`.

- If this finishes the text,
  - register whether there were any nonsuppressible spaces (bSawNonSuppressible) in iFlags.
  - pass all the info to makeBreakPoss and _return_ its breakposs.
- Else,
  - add pending space.
- If hypenation is on, get the size of the next syllable:
  - get the size of the next syllable,
    - if successful, add the flags BreakPoss.CAN_BREAK_AFTER and BreakPoss.CAN_BREAK_AFTER in iFlags,
  - add the syllable length to the word ipd.
- Else look for a legal line-break: breakable white-space and certain characters such as '-' which can serve as word breaks; don't look for hyphenation points here though,
  - for all following characters:
    - If this is a newline character U+0A, or if textInfo.bWrap and this is a breakable space, or if this is one of the linebreak characters ("-/") and it is the first character or the preceding character is a letter or a digit,
      - add the flag BreakPoss.CAN_BREAK_AFTER to iFlags,
      - if this is not a space U+20,
        - move the counter to the next character,
        - if this is a newline U+0A, add the flag BreakPoss.FORCE to iFlags,
        - else add the character width to the word ipd.
      - if the rest of the text consists of spaces U+20, register that the rest is suppressible at a line break (BreakPoss.REST_ARE_SUPPRESS_AT_LB) in iFlags,
      - pass all the info to makeBreakPoss and _return_ its breakposs.
    - Else add the character width to the word ipd,
  - and continue with the cycle for the next character.
- At the end of the text, pass all the info to makeBreakPoss and _return_ its breakposs.

5.9. TextLayoutManager.makeBreakPoss

- Make word ipd into a MinOptMax object.
- Add space ipd to it.
- Add total ipd from previous texts to it.
- Create an AreaInfo object for this text fragment and add it to the vector of AreaInfo objects.
- Create a breakposs for this text fragment.
- Set the total ipd to the current ipd.
- If the flags contain BreakPoss.HYPHENATED, set the stacking size to the ipd plus the width of the hyphen character,
- Else set the stacking size to the ipd.
- Set the non-stacking size to the line height, from the text info.
- Register the descender and ascender with the breakposs object; this is currently commented out.
- If this is the end of the text,
  - add BreakPoss.ISLAST to the flags,
  - declare the LM finished.
- Register the flags with the breakposs object
- Register the pending space or the absence thereof with the breakposs object.
- Register the leading space or the absence thereof with the breakposs object.
- Return the breakposs object.

Chapter 6

Phase 2b: The addAreas call tree

6.1. Overview

This section presents a verbose overview of the `addAreas` call tree. The following section presents the Layout Managers in more detail.

- FlowLM receives from its parent LM an iterator with a single pagebreak. The first belongs to BlockLM1, the second belongs to BlockLM2. FlowLM itself holds 2 child BPs. The flow consists of two blocks. FlowLM sets up an iterator with its 2 BPs.

- BlockLM1 receives from its parent LM the iterator with those 2 BPs, of which only the first one belongs to it. Its leaf position is 13. BlockLM itself holds 14 child BPs, which all belong to a single LineLM. The block consists of 14 lines. BlockLM sets up an iterator corresponding to the first BP, containing the child BPs 0–13.

- LineLM receives from its parent LM an iterator with those 14 BPs. The leaf positions are 3, 6, 11, 12, 13, 16, 19, 21, 23, 24, 25, 26, 27, 28. LineLM itself holds 29 child BPs, which all belong to a single TextLM. LineLM maintains the position of the next BP in `vecInlineBreaks`, `iStartPos`. Initially it is set to 0. For each of its 14 BPs in the iterator, LineLM sets up an iterator with the child BPs in `vecInlineBreaks` from `iStartPos` up to an including the index `iLeafPos` to which the iterator BP points. Then it `updates` iStartPos to point to the next child BP. The iterators contain the child BP ranges: 0–3, 4–6, 7–11, 12, 13, 14–16, 17–19, 20–21, 22–23, 24, 25, 26, 27, 28.

```
while (parentIter.hasNext()) {
    LineBreakPosition lbp = (LineBreakPosition) parentIter.next();
        ...
    PositionIterator inlinePosIter =
      new BreakPossPosIter(vecInlineBreaks, iStartPos,
                           lbp.getLeafPos() + 1);
    iStartPos = lbp.getLeafPos() + 1;
        ...
    while ((childLM = inlinePosIter.getNextChildLM()) != null) {
        childLM.addAreas(inlinePosIter, lc);
                ...
        }
        ...
}
```

- TextLM receives from its parent LM an iterator with the BPs 0–3. The leaf positions are 0, 1, 2, 3. It has itself 47 items in `vecAreaInfo`. It iterates over the 4 corresponding AIs, records the start of the first one, counts the word spaces, and records the end of the last one. This line contains the characters from 0 up to 13 and has no word spaces.

- TextLM receives from its parent LM an iterator with the BPs 4–6. The leaf positions are 5, 7, 8. It iterates over the three corresponding AIs. This line contains the characters from 13 up to 26 and has one word space. Note that the AIs 4 and 6 remain unused because they do not have a corresponding BP. These AIs represent areas that were too long, and over which the LM backed up.

- TextLM receives from its parent LM an iterator with the BPs 7–11. The leaf positions are 11, 12, 13, 14, 15. It iterates over the five corresponding AIs. This line contains the characters from 26 up to 40 and has no word spaces. Note that the AIs 9 and 10 remain unused because they do not have a corresponding BP.

- TextLM receives from its parent LM an iterator with the single BP 12. The leaf position is 16. This line contains the characters from 40 up to 42 and has no word spaces.

- TextLM receives from its parent LM an iterator with the single BP 13. The leaf position is 18. This line con-

tains the characters from 43 up to 60 and has no word spaces. Note that the AI 17 remains unused because it does not have a corresponding BP. Note also that character 42 has been dropped, because it would be a leading space.

- etc. until all 14 line areas are done. LineLM returns.
- The second BP in the iterator from FlowLM belongs to BlockLM2. The loop `while (parentIter.hasNext())` ends because the LM of the next object is different from the current LM (`BreakPossPosIter.checkNext()`), and BlockLM1 returns. FlowLM's loop `while ((childLM = breakPosIter.getNextChildLM()) != null)` then passes the iterator to BlockLM2.
- BlockLM2 receives from its parent LM the iterator with those 2 BPs. The cursor is at one, because BlockLM1 has used the first object. Only the second BP belongs to BlockLM2. Its leaf position is 0. BlockLM itself holds 1 child BP, belonging to a LineLM. The block consists of a single line. BlockLM sets up an iterator corresponding to the second BP, containing a single child BP.
- LineLM receives from its parent LM an iterator with that BP. The leaf position is 1. LineLM itself holds 2 child BPs, one belonging to a TextLM, the other to `AddLMVisitor$2`. LineLM sets up an iterator corresponding to the BP, containing its two child BPs.
- TextLM receives from its parent LM an iterator with the BPs 0 and 1, of which only the first belongs to it. Its leaf position is 0. It iterates over the corresponding AI. This text area contains the characters from 0 up to 1, i.e. `" "`, and has one word space. This converted to a space area.
- `AddLMVisitor$2` receives from its parent LM an iterator with the BPs 0 and 1. The cursor is at one, because TextLM has used the first object. Only the second BP belongs to `AddLMVisitor$2`. Its leaf position is 0.
- This completes the line. LineLM returns.
- BlockLM2 returns.
- FlowLM returns.

6.2. Detailed overviews

6.2.1. PageLM

```
 bbp = {
    breakps: instance of org.apache.fop.layoutmgr.BreakPoss(id=1167)
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 0
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 bbp.breakps.position = {
    iLeafPos: 1
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 list = "[org.apache.fop.layoutmgr.BreakPoss@1aa2c23]"

 list.get(0).position = {
    iLeafPos: 1
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 list.get(0).position.layoutManager = "org.apache.fop.layoutmgr.FlowLayoutManager@6963d0"
```

6.2.2. FlowLM

```
 this = "org.apache.fop.layoutmgr.FlowLayoutManager@6963d0"

 lfp = {
    iLeafPos: 1
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 iStartPos = 0

 blockBreaks = "[org.apache.fop.layoutmgr.BreakPoss@111bfbc, org.apache.fop.layoutmg

 blockBreaks.get(iStartPos).position = {
    iLeafPos: 13
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 blockBreaks.get(iStartPos).position.layoutManager = "org.apache.fop.layoutmgr.Block

 blockBreaks.get(iStartPos+1).position = {
    iLeafPos: 0
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 blockBreaks.get(iStartPos+1).position.layoutManager = "org.apache.fop.layoutmgr.Bloc
```

6.2.3. BlockLM1

```
 this = "org.apache.fop.layoutmgr.BlockLayoutManager@19e09a4"

 lfp = {
    iLeafPos: 13
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 iStartPos = 0

 childBreaks = "[
org.apache.fop.layoutmgr.BreakPoss@2b249,
org.apache.fop.layoutmgr.BreakPoss@106daba,
org.apache.fop.layoutmgr.BreakPoss@1021f34,
org.apache.fop.layoutmgr.BreakPoss@4eb043,
org.apache.fop.layoutmgr.BreakPoss@163956,
org.apache.fop.layoutmgr.BreakPoss@10e434d,
org.apache.fop.layoutmgr.BreakPoss@16477d9,
org.apache.fop.layoutmgr.BreakPoss@f864fe,
org.apache.fop.layoutmgr.BreakPoss@1ae9aaa,
org.apache.fop.layoutmgr.BreakPoss@2c17f7,
org.apache.fop.layoutmgr.BreakPoss@d9896e,
org.apache.fop.layoutmgr.BreakPoss@1cda59b,
org.apache.fop.layoutmgr.BreakPoss@33788d,
org.apache.fop.layoutmgr.BreakPoss@12fb0af
]"

 childBreaks.get(0).position = {
    dAdjust: 0.0
    ipdAdjust: 1.0
    startIndent: 0
    lineHeight: 19200
    baseline: 17000
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 3
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 childBreaks.get(0).position.layoutManager = "org.apache.fop.layoutmgr.LineLayoutMan
```

```
 childBreaks.get(1).position = {
    dAdjust: 0.0
    ipdAdjust: 1.0
    startIndent: 0
    lineHeight: 19200
    baseline: 17000
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 6
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 childBreaks.get(1).position.layoutManager = "org.apache.fop.layoutmgr.LineLayoutManager@c

 childBreaks.get(2).position = {
    dAdjust: 0.0
    ipdAdjust: 1.0
    startIndent: 0
    lineHeight: 19200
    baseline: 17000
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 11
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 childBreaks.get(2).position.layoutManager = "org.apache.fop.layoutmgr.LineLayoutManager@c

 childBreaks.get(3).position = {
    dAdjust: 7.0
    ipdAdjust: 1.0
    startIndent: 0
    lineHeight: 19200
    baseline: 17000
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 12
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 childBreaks.get(3).position.layoutManager = "org.apache.fop.layoutmgr.LineLayoutManager@c

 childBreaks.get(4).position = {
    dAdjust: 0.0
    ipdAdjust: -1.0
    startIndent: 0
    lineHeight: 19200
    baseline: 17000
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 13
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 childBreaks.get(4).position.layoutManager = "org.apache.fop.layoutmgr.LineLayoutManager@c

 childBreaks.get(5).position = {
    dAdjust: 0.0
    ipdAdjust: 1.0
    startIndent: 0
    lineHeight: 19200
    baseline: 17000
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 16
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 childBreaks.get(5).position.layoutManager = "org.apache.fop.layoutmgr.LineLayoutManager@c

 childBreaks.get(6).position = {
    dAdjust: 0.0
    ipdAdjust: 1.0
    startIndent: 0
    lineHeight: 19200
    baseline: 17000
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 19
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}
```

```
 childBreaks.get(6).position.layoutManager = "org.apache.fop.layoutmgr.LineLayoutMana

 childBreaks.get(7).position = {
    dAdjust: 0.0
    ipdAdjust: 1.0
    startIndent: 0
    lineHeight: 19200
    baseline: 17000
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 21
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 childBreaks.get(7).position.layoutManager = "org.apache.fop.layoutmgr.LineLayoutMana

 childBreaks.get(8).position = {
    dAdjust: 0.0
    ipdAdjust: 1.0
    startIndent: 0
    lineHeight: 19200
    baseline: 17000
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 23
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 childBreaks.get(8).position.layoutManager = "org.apache.fop.layoutmgr.LineLayoutMana

 childBreaks.get(9).position = {
    dAdjust: 0.0
    ipdAdjust: 1.0
    startIndent: 0
    lineHeight: 19200
    baseline: 17000
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 24
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 childBreaks.get(9).position.layoutManager = "org.apache.fop.layoutmgr.LineLayoutMana

 childBreaks.get(10).position = {
    dAdjust: 0.0
    ipdAdjust: 1.0
    startIndent: 0
    lineHeight: 19200
    baseline: 17000
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 25
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 childBreaks.get(10).position.layoutManager = "org.apache.fop.layoutmgr.LineLayoutMa

 childBreaks.get(11).position = {
    dAdjust: 0.0
    ipdAdjust: 1.0
    startIndent: 0
    lineHeight: 19200
    baseline: 17000
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 26
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 childBreaks.get(11).position.layoutManager = "org.apache.fop.layoutmgr.LineLayoutMa

 childBreaks.get(12).position = {
    dAdjust: 0.0
    ipdAdjust: 1.0
    startIndent: 0
    lineHeight: 19200
    baseline: 17000
```

```
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 27
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 childBreaks.get(12).position.layoutManager = "org.apache.fop.layoutmgr.LineLayoutManager@

 childBreaks.get(13).position = {
    dAdjust: 0.0
    ipdAdjust: 1.0
    startIndent: 0
    lineHeight: 19200
    baseline: 17000
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 28
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 childBreaks.get(13).position.layoutManager = "org.apache.fop.layoutmgr.LineLayoutManager@
```

6.2.4. LineLM

```
 this = "org.apache.fop.layoutmgr.LineLayoutManager@c06258"

 lbp = {
    dAdjust: 0.0
    ipdAdjust: 1.0
    startIndent: 0
    lineHeight: 19200
    baseline: 17000
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 3
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 iStartPos = 0

 vecInlineBreaks = "[
org.apache.fop.layoutmgr.BreakPoss@17e4dee,
org.apache.fop.layoutmgr.BreakPoss@12e7c6a,
org.apache.fop.layoutmgr.BreakPoss@ea5461,
org.apache.fop.layoutmgr.BreakPoss@49cf9f,
org.apache.fop.layoutmgr.BreakPoss@1de0b5e,
org.apache.fop.layoutmgr.BreakPoss@bc5596,
org.apache.fop.layoutmgr.BreakPoss@970c0e,
org.apache.fop.layoutmgr.BreakPoss@987197,
org.apache.fop.layoutmgr.BreakPoss@497904,
org.apache.fop.layoutmgr.BreakPoss@1a7f9dc,
org.apache.fop.layoutmgr.BreakPoss@104e28b,
org.apache.fop.layoutmgr.BreakPoss@1b54362,
org.apache.fop.layoutmgr.BreakPoss@15b0e2c,
org.apache.fop.layoutmgr.BreakPoss@ff9053,
org.apache.fop.layoutmgr.BreakPoss@5c7734,
org.apache.fop.layoutmgr.BreakPoss@96212a,
org.apache.fop.layoutmgr.BreakPoss@5b675e,
org.apache.fop.layoutmgr.BreakPoss@df83e5,
org.apache.fop.layoutmgr.BreakPoss@4c6320,
org.apache.fop.layoutmgr.BreakPoss@ffd135,
org.apache.fop.layoutmgr.BreakPoss@1000bcf,
org.apache.fop.layoutmgr.BreakPoss@754fc,
org.apache.fop.layoutmgr.BreakPoss@15c998a,
org.apache.fop.layoutmgr.BreakPoss@6458a6,
org.apache.fop.layoutmgr.BreakPoss@1f82ab4,
org.apache.fop.layoutmgr.BreakPoss@1bb9696,
org.apache.fop.layoutmgr.BreakPoss@9b6220,
org.apache.fop.layoutmgr.BreakPoss@1474e45,
org.apache.fop.layoutmgr.BreakPoss@63a721
]"
```

```
 vecInlineBreaks.get(0).position = {
    iLeafPos: 0
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 vecInlineBreaks.get(0).position.layoutManager = "org.apache.fop.layoutmgr.TextLayou

 vecInlineBreaks.get(1).position = {
    iLeafPos: 1
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 vecInlineBreaks.get(1).position.layoutManager = "org.apache.fop.layoutmgr.TextLayou

 vecInlineBreaks.get(2).position = {
    iLeafPos: 2
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 vecInlineBreaks.get(2).position.layoutManager = "org.apache.fop.layoutmgr.TextLayou

 vecInlineBreaks.get(3).position = {
    iLeafPos: 3
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 vecInlineBreaks.get(3).position.layoutManager = "org.apache.fop.layoutmgr.TextLayou

 vecInlineBreaks.get(4).position = {
    iLeafPos: 5
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 vecInlineBreaks.get(4).position.layoutManager = "org.apache.fop.layoutmgr.TextLayou

 vecInlineBreaks.get(5).position = {
    iLeafPos: 7
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 vecInlineBreaks.get(5).position.layoutManager = "org.apache.fop.layoutmgr.TextLayou

 vecInlineBreaks.get(6).position = {
    iLeafPos: 8
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 vecInlineBreaks.get(6).position.layoutManager = "org.apache.fop.layoutmgr.TextLayou

 vecInlineBreaks.get(7).position = {
    iLeafPos: 11
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 vecInlineBreaks.get(7).position.layoutManager = "org.apache.fop.layoutmgr.TextLayou

 vecInlineBreaks.get(8).position = {
    iLeafPos: 12
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 vecInlineBreaks.get(8).position.layoutManager = "org.apache.fop.layoutmgr.TextLayou

 vecInlineBreaks.get(9).position = {
    iLeafPos: 13
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 vecInlineBreaks.get(9).position.layoutManager = "org.apache.fop.layoutmgr.TextLayou
```

```
 vecInlineBreaks.get(10).position = {
    iLeafPos: 14
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 vecInlineBreaks.get(10).position.layoutManager = "org.apache.fop.layoutmgr.TextLayoutMana

 vecInlineBreaks.get(11).position = {
    iLeafPos: 15
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 vecInlineBreaks.get(11).position.layoutManager = "org.apache.fop.layoutmgr.TextLayoutMana

 vecInlineBreaks.get(12).position = {
    iLeafPos: 16
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 vecInlineBreaks.get(12).position.layoutManager = "org.apache.fop.layoutmgr.TextLayoutMana

 vecInlineBreaks.get(13).position = {
    iLeafPos: 18
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 vecInlineBreaks.get(13).position.layoutManager = "org.apache.fop.layoutmgr.TextLayoutMana

 vecInlineBreaks.get(14).position = {
    iLeafPos: 20
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 vecInlineBreaks.get(14).position.layoutManager = "org.apache.fop.layoutmgr.TextLayoutMana

 vecInlineBreaks.get(15).position = {
    iLeafPos: 21
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 vecInlineBreaks.get(15).position.layoutManager = "org.apache.fop.layoutmgr.TextLayoutMana

 vecInlineBreaks.get(16).position = {
    iLeafPos: 22
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 vecInlineBreaks.get(16).position.layoutManager = "org.apache.fop.layoutmgr.TextLayoutMana

 vecInlineBreaks.get(17).position = {
    iLeafPos: 23
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 vecInlineBreaks.get(17).position.layoutManager = "org.apache.fop.layoutmgr.TextLayoutMana

 vecInlineBreaks.get(18).position = {
    iLeafPos: 25
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 vecInlineBreaks.get(18).position.layoutManager = "org.apache.fop.layoutmgr.TextLayoutMana
```

6.2.5. TextLM

```
 this = "org.apache.fop.layoutmgr.TextLayoutManager@57ea52"

 this.chars = {
w, a, t, e, r, s, t, a, a, t, s, i, n, g, e, n, i, e, u, r,
  , m, i, n, i, s, t, e, r, s, p, o, r, t, e, f, e, u, i, l, l, e,
  , a, a, n, d, a, c, h, t, s, t, r, e, k, k, e, r, .,
  , V, e, r, n, e, d, e, r, l, a, n, d, s, t, e,
  , v, a, k, l, i, t, e, r, a, t, u, u, r,
  , v, e, r, s, c, h, i, l, l, e, n, d,
  , v, e, r, h, o, l, l, a, n, d, s, t, e,
  , v, a, k, l, i, t, e, r, a, t, u, u, r, .,
  , b, e, s, t, u, u, r, s, t, a, k, e, n,
  , l, a, n, d, s, t, a, a, l,
  , b, e, l, a, n, g, r, i, j, k, .
}

 tbpNext = {
    iLeafPos: 0
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 vecAreaInfo = "[
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@107bd0d,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@12922f6,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@1b66b06,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@12c9557,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@9f0d,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@ca3783,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@2a6ff,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@21d23b,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@7124af,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@1f7708,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@1bfbfb8,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@1c3e9ba,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@125d61e,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@10c6cfc,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@c72243,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@19a8416,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@155d3a3,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@1b994de,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@dc9766,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@57e787,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@1217e67,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@1f1bd98,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@1d686c1,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@128edf2,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@1dddba,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@c7e8a7,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@7b4703,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@1732ed2,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@1071521,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@1fc3c84,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@e93999,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@1c486f2,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@1779885,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@be76c7,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@682406,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@115126e,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@6d2380,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@135b1f3,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@35e6e3,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@c9630a,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@185572a,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@11daa0e,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@879860,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@24de7d,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@8b058b,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@1192059,
org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@9ac0f5
```

```
]"

 vecAreaInfo.get(0) = {
    iStartIndex: 0
    iBreakIndex: 2
    iWScount: 0
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1304)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(1) = {
    iStartIndex: 2
    iBreakIndex: 5
    iWScount: 0
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1308)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(2) = {
    iStartIndex: 5
    iBreakIndex: 11
    iWScount: 0
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1310)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(3) = {
    iStartIndex: 11
    iBreakIndex: 13
    iWScount: 0
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1312)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(4) = {
    iStartIndex: 13
    iBreakIndex: 15
    iWScount: 0
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1314)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(5) = {
    iStartIndex: 13
    iBreakIndex: 20
    iWScount: 0
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1316)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(6) = {
    iStartIndex: 20
    iBreakIndex: 42
    iWScount: 1
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1318)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(7) = {
    iStartIndex: 20
    iBreakIndex: 23
    iWScount: 1
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1320)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(8) = {
    iStartIndex: 23
    iBreakIndex: 26
```

```
    iWScount: 0
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1322)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(9) = {
    iStartIndex: 26
    iBreakIndex: 30
    iWScount: 0
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1324)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(10) = {
    iStartIndex: 26
    iBreakIndex: 42
    iWScount: 0
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1326)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(11) = {
    iStartIndex: 26
    iBreakIndex: 29
    iWScount: 0
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1328)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(12) = {
    iStartIndex: 29
    iBreakIndex: 30
    iWScount: 0
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1330)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(13) = {
    iStartIndex: 30
    iBreakIndex: 33
    iWScount: 0
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1332)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(14) = {
    iStartIndex: 33
    iBreakIndex: 35
    iWScount: 0
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1334)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(15) = {
    iStartIndex: 35
    iBreakIndex: 40
    iWScount: 0
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1336)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(16) = {
    iStartIndex: 40
    iBreakIndex: 42
    iWScount: 0
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1338)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}
```

```
 vecAreaInfo.get(17) = {
    iStartIndex: 42
    iBreakIndex: 60
    iWScount: 1
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1340)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}

 vecAreaInfo.get(18) = {
    iStartIndex: 43
    iBreakIndex: 60
    iWScount: 0
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1342)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1246)
}
```

6.2.6. BlockLM2

```
 this = "org.apache.fop.layoutmgr.BlockLayoutManager@144b18f"

 lfp = {
    iLeafPos: 0
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 iStartPos = 0

 childBreaks = "[org.apache.fop.layoutmgr.BreakPoss@145f939]"

 childBreaks.get(0).position = {
    dAdjust: 0.0
    ipdAdjust: -1.0
    startIndent: 0
    lineHeight: 14400
    baseline: 12750
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 1
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 childBreaks.get(0).position.layoutManager = "org.apache.fop.layoutmgr.LineLayoutManager@d
```

6.2.7. LineLM

```
 this = "org.apache.fop.layoutmgr.LineLayoutManager@df2d38"

 lbp = {
    dAdjust: 0.0
    ipdAdjust: -1.0
    startIndent: 0
    lineHeight: 14400
    baseline: 12750
    org.apache.fop.layoutmgr.LeafPosition.iLeafPos: 1
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layoutmgr.
}

 iStartPos = 0

 vecInlineBreaks = "[
org.apache.fop.layoutmgr.BreakPoss@eb67e8,
org.apache.fop.layoutmgr.BreakPoss@f2ea42
]"

 vecInlineBreaks.get(0).position = {
```

```
    iLeafPos: 0
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 vecInlineBreaks.get(0).position.layoutManager = "org.apache.fop.layoutmgr.TextLayou

 vecInlineBreaks.get(1).position = {
    iLeafPos: 0
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 vecInlineBreaks.get(1).position.layoutManager = "org.apache.fop.layoutmgr.AddLMVisi
```

6.2.8. TextLM

```
 this = "org.apache.fop.layoutmgr.TextLayoutManager@e265d0"

 this.chars = {

}

 tbpNext = {
    iLeafPos: 0
    org.apache.fop.layoutmgr.Position.layoutManager: instance of org.apache.fop.layou
}

 vecAreaInfo = "[org.apache.fop.layoutmgr.TextLayoutManager$AreaInfo@996cca]"

 vecAreaInfo.get(0) = {
    iStartIndex: 0
    iBreakIndex: 1
    iWScount: 1
    ipdArea: instance of org.apache.fop.traits.MinOptMax(id=1396)
    this$0: instance of org.apache.fop.layoutmgr.TextLayoutManager(id=1385)
}
```

6.2.9. AddLMVisitor$2

No data

Chapter 7

Phase 3: Rendering the pages

It is the task of the rendering phase to describe the area tree in the target page description language, so that viewers for that language can render the pages. Rendering is done page by page. For each page the rendering system is handed a PageViewport, and it walks the area subtree below it. For each area it retrieves the traits and data, and generates the required output.

The layout of a page is not finished until all forward references on that page have been resolved. As a consequence pages are finished out of pagination order. Some renderers support out of order rendering of pages, `AbstractRenderer.supportsOutOfOrder()`. If a renderer does, a finished page is handed over to it immediately. Otherwise, the layout system keeps finished pages until all preceding pages are also finished and have been handed over to the renderer. In principle, the PDF renderer supports out of order rendering. In current FOP (27 June 2004) this has been disabled because the support is broken.

This stack at a deep position, rendering a leader in a block in a flow in the body region, shows some details of rendering. Note how the hierarchy of the area tree can be recognized. The lower frames show how the rendering system is called by the layout system:

```
main[1] where
  [1] org.apache.fop.render.pdf.PDFRenderer.renderLeader (PDFRenderer.java:1,266)
  [2] org.apache.fop.render.AbstractRenderer.serveVisitor (AbstractRenderer.java:832
  [3] org.apache.fop.area.inline.Leader.acceptVisitor (Leader.java:118)
  [4] org.apache.fop.render.AbstractRenderer.renderLineArea (AbstractRenderer.java:6
  [5] org.apache.fop.render.pdf.PDFRenderer.renderLineArea (PDFRenderer.java:830)
  [6] org.apache.fop.render.AbstractRenderer.renderBlocks (AbstractRenderer.java:547
  [7] org.apache.fop.render.AbstractRenderer.renderBlock (AbstractRenderer.java:588)
  [8] org.apache.fop.render.pdf.PDFRenderer.renderBlock (PDFRenderer.java:513)
  [9] org.apache.fop.render.AbstractRenderer.renderBlocks (AbstractRenderer.java:538
  [10] org.apache.fop.render.AbstractRenderer.renderFlow (AbstractRenderer.java:473)
  [11] org.apache.fop.render.AbstractRenderer.renderMainReference (AbstractRenderer.
  [12] org.apache.fop.render.AbstractRenderer.renderBodyRegion (AbstractRenderer.jav
  [13] org.apache.fop.render.AbstractRenderer.renderRegionViewport (AbstractRenderer
  [14] org.apache.fop.render.AbstractRenderer.renderPageAreas (AbstractRenderer.java
  [15] org.apache.fop.render.pdf.PDFRenderer.renderPage (PDFRenderer.java:471)
  [16] org.apache.fop.area.RenderPagesModel.addPage (RenderPagesModel.java:117)
  [17] org.apache.fop.area.AreaTree.addPage (AreaTree.java:143)
  [18] org.apache.fop.layoutmgr.PageLayoutManager.finishPage (PageLayoutManager.java
  [19] org.apache.fop.layoutmgr.PageLayoutManager.doLayout (PageLayoutManager.java:2
```

Obviously there is a lot to be documented about the rendering system, and about each renderer separately. Because I do not (yet) know much about the rendering system, I will have to leave that task to others. I only add the obvious: Rendering requires precise programming: spacing and progress calculations, saving and restoring dimensions, etc. It also requires tracking the state in the output format.

Chapter 8

The trees in FOP

8.1. Overview

1. The FO document. Each XML document and therefore also an FO document has a hierarchical structure that can be modeled as a tree.

2. The FO DOM tree. This tree has the same hierarchical structure as an FO document, and is often built from an XML document.

FOP can work from either an FO document or an FO DOM tree. FOP does not build a DOM tree from an FO document.

3. The FO tree. FOP builds this tree from either the FO document or the FO DOM tree. The FO nodes in this tree correspond to the elements in the FO document or the XML nodes in the FO DOM tree. Note, however, that they are different from DOM tree nodes.

When a sufficient part of the FO tree has been built (in current FOP the subtree of a PageSequence FO node), the layout process is started. This process builds three trees.

4. The LM tree. This tree corresponds closely to the FO tree because each FO node creates a layout manager. But there are deviations. For example, a BlockLayoutManager creates a LineLayoutManager for each consecutive sequence of child FO nodes that generate inline areas.

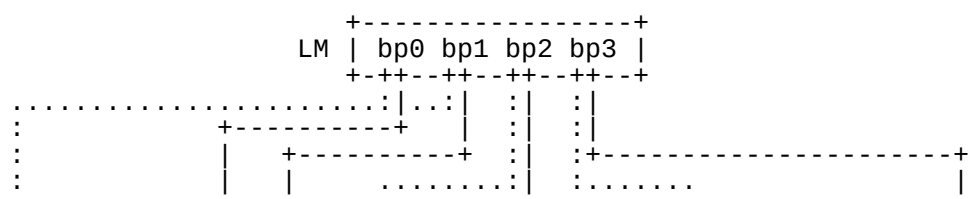
5. The BP tree. Each layout manager returns zero or more BreakPoss to its parent. These BreakPoss are connected to BreakPoss that the layout manager received from its child layout managers. This is not a real tree. There are many BP without a parent BP. They are connected to their siblings by their LM.

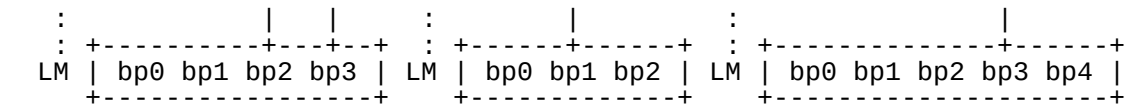
6. The Area tree. Using the information stored in the BP tree and in the LMs connected with the BPs, layout areas are constructed. These areas are placed within other areas, which is expressed in a tree hierarchy.

8.2. The tree of BreakPoss

Each LM contains a list of BPs belonging to and returned to it by the childLMs during the `getNextBreakPoss` stack. These are the BPs that end an area of the childLM. The BP contains an index `position.iLeafPos`, which connects it to the BP with that index in the list of BPs of its own LM (the childLM).

For example, BlockLM's list `childBreaks` contains the BPs that end a line (if the childLM is a LineLM). LineLM's list `vecInlineBreaks` contains the BPs that were returned to it as possible linebreaks by TextLM (if the childLM is a TextLM). TextLM's list `vecAreaInfo` contains `AreaInfo` objects. A BP in BlockLM's list `childBreaks` belongs e.g. to a LineLM. Its index `position.iLeafPos` points to the BP with that index in `vecInlineBreaks` in ListLM. That BP belongs e.g. to a TextLM, and its index `position.iLeafPos` points to the `AreaInfo` object with that index in `vecAreaInfo` in TextLM.





The BPs are held in a list by the LM shown in front of them. They are associated with one of the childLMs, which is shown by the dotted lines. Their member `position.iLeafPos` connects them with the BP in their LM's list with that index, as shown by the dashed lines.

8.3. Example of an FO and area tree

8.3.1. The FO file

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
   xmlns:svg="http://www.w3.org/2000/svg">
 <fo:layout-master-set>
   <fo:simple-page-master master-name="simpleA4"
      page-height="29.7cm" page-width="21cm"
      margin-top="2cm"
      margin-bottom="2cm" margin-left="2cm"
      margin-right="2cm">
    <fo:region-body/>
   </fo:simple-page-master>
 </fo:layout-master-set>
 <fo:page-sequence master-reference="simpleA4">
   <fo:flow flow-name="xsl-region-body">
    <fo:block font-size="16pt" font-weight="bold"
       space-after="5mm">Test FO
    </fo:block>
   </fo:flow>
 </fo:page-sequence>
</fo:root>
```

8.3.2. The corresponding FO tree

In the listing below the notation has been shortened; `.[n]` denotes the nth child, for which the full notation is `.children.elementData[n]` . A number of static members are not shown.

The root:

```
 root = "fo:root at line 2:44"

 root = {
    layoutMasterSet: instance of org.apache.fop.fo.pagination.LayoutMasterSet(id=1089)
    pageSequences: instance of java.util.ArrayList(id=1102)
    runningPageNumberCounter: 0
    foTreeControl: instance of org.apache.fop.apps.Document(id=1103)
    org.apache.fop.fo.FObj.propertyListTable: instance of org.apache.fop.fo.Property$Maker
    org.apache.fop.fo.FObj.propertyList: instance of org.apache.fop.fo.PropertyList(id=110
    org.apache.fop.fo.FObj.propMgr: instance of org.apache.fop.fo.PropertyManager(id=1105)
    org.apache.fop.fo.FObj.id: null
    org.apache.fop.fo.FObj.children: instance of java.util.ArrayList(id=1106)
    org.apache.fop.fo.FObj.markers: null
    org.apache.fop.fo.FObj.systemId: "file:/path/to/fo-file"
    org.apache.fop.fo.FObj.line: 2
    org.apache.fop.fo.FObj.column: 44
    org.apache.fop.fo.FONode.parent: null
    org.apache.fop.fo.FONode.name: "fo:root"
}
```

The root has no properties; the namespace nodes do not result in properties:

```
 root.propertyList = "{}"
```

The root has two children:

```
 root.children = "[
  fo:layout-master-set at line 3:25
  fo:page-sequence at line 12:49
]"
```

The first child of root is the layout master set:

```
 root.[0] = "fo:layout-master-set at line 3:25"

 root.[0] = {
    simplePageMasters: instance of java.util.HashMap(id=1111)
    pageSequenceMasters: instance of java.util.HashMap(id=1112)
    org.apache.fop.fo.FObj.propertyListTable: instance of org.apache.fop.fo.Property
    org.apache.fop.fo.FObj.propertyList: instance of org.apache.fop.fo.PropertyList(
    org.apache.fop.fo.FObj.propMgr: instance of org.apache.fop.fo.PropertyManager(id
    org.apache.fop.fo.FObj.id: null
    org.apache.fop.fo.FObj.children: instance of java.util.ArrayList(id=1115)
    org.apache.fop.fo.FObj.markers: null
    org.apache.fop.fo.FObj.systemId: "file:/path/to/fo-file"
    org.apache.fop.fo.FObj.line: 3
    org.apache.fop.fo.FObj.column: 25
    org.apache.fop.fo.FONode.parent: instance of org.apache.fop.fo.pagination.Root(i
    org.apache.fop.fo.FONode.name: "fo:layout-master-set"
}

 root.[0].propertyList = "{}"
```

The layout master set contains a simple page master:

```
 root.[0].children = "[
  fo:simple-page-master at line 8:28
]"

 root.[0].[0] = "fo:simple-page-master at line 8:28"

 root.[0].[0] = {
    regions: instance of java.util.HashMap(id=1120)
    masterName: "simpleA4"
    org.apache.fop.fo.FObj.propertyListTable: instance of org.apache.fop.fo.Property
    org.apache.fop.fo.FObj.propertyList: instance of org.apache.fop.fo.PropertyList(
    org.apache.fop.fo.FObj.propMgr: instance of org.apache.fop.fo.PropertyManager(id
    org.apache.fop.fo.FObj.id: null
    org.apache.fop.fo.FObj.children: null
    org.apache.fop.fo.FObj.markers: null
    org.apache.fop.fo.FObj.systemId: "file:/path/to/fo-file"
    org.apache.fop.fo.FObj.line: 8
    org.apache.fop.fo.FObj.column: 28
    org.apache.fop.fo.FONode.parent: instance of org.apache.fop.fo.pagination.Layout
    org.apache.fop.fo.FONode.name: "fo:simple-page-master"
}
```

The properties of the simple page master:

```
 root.[0].[0].propertyList = "{
  master-name=org.apache.fop.fo.StringProperty@1958bf9
  margin-top=org.apache.fop.fo.LengthProperty@118958e
  margin-right=org.apache.fop.fo.LengthProperty@102b2b6
  margin-bottom=org.apache.fop.fo.LengthProperty@22d166
  margin-left=org.apache.fop.fo.LengthProperty@1e1962d
  page-width=org.apache.fop.fo.LengthProperty@14a75bb
  page-height=org.apache.fop.fo.LengthProperty@17779e3
}"

 root.[0].[0].propertyList.get("master-name") = {
    str: "simpleA4"
    org.apache.fop.fo.Property.specVal: null
}

 root.[0].[0].propertyList.get("page-height") = {
    length: instance of org.apache.fop.datatypes.FixedLength(id=1329)
    org.apache.fop.fo.Property.specVal: null
}

 root.[0].[0].propertyList.get("page-height").length = "841889mpt"
```

The simple page master has only one region, the body:

```
 root.[0].[0].regions = "{body=fo:region-body at line 9:24}"
```

The region body:

```
 root.[0].[0].regions.get("body") = "fo:region-body at line 9:24"

 root.[0].[0].regions.get("body") = {
    backgroundColor: null
    org.apache.fop.fo.pagination.Region.layoutMaster: instance of org.apache.fop.fo.pagina
    org.apache.fop.fo.pagination.Region.regionName: "xsl-region-body"
    org.apache.fop.fo.pagination.Region.overflow: 8
    org.apache.fop.fo.pagination.Region.wm: 49
    org.apache.fop.fo.FObj.propertyListTable: instance of org.apache.fop.fo.Property$Maker
    org.apache.fop.fo.FObj.propertyList: instance of org.apache.fop.fo.PropertyList(id=134
    org.apache.fop.fo.FObj.propMgr: instance of org.apache.fop.fo.PropertyManager(id=1345)
    org.apache.fop.fo.FObj.id: null
    org.apache.fop.fo.FObj.children: null
    org.apache.fop.fo.FObj.markers: null
    org.apache.fop.fo.FObj.systemId: "file:/path/to/fo-file"
    org.apache.fop.fo.FObj.line: 9
    org.apache.fop.fo.FObj.column: 24
    org.apache.fop.fo.FONode.parent: instance of org.apache.fop.fo.pagination.SimplePageMa
    org.apache.fop.fo.FONode.name: "fo:region-body"
}

 root.[0].[0].regions.get("body").propertyList = "{}"
```

The second child of root is the page sequence:

```
 root.[1] = "fo:page-sequence at line 12:49"

 root.[1] = {
    root: instance of org.apache.fop.fo.pagination.Root(id=1088)
    layoutMasterSet: instance of org.apache.fop.fo.pagination.LayoutMasterSet(id=1089)
    flowMap: instance of java.util.HashMap(id=1090)
    sequenceStarted: true
    ipnValue: "auto"
    currentPageNumber: 1
    explicitFirstNumber: 0
```

```
    firstPageNumber: 1
    pageNumberGenerator: instance of org.apache.fop.fo.pagination.PageNumberGenerato
    forcePageCount: 8
    pageCount: 0
    isForcing: false
    pageNumberType: 1
    thisIsFirstPage: true
    simplePageMaster: instance of org.apache.fop.fo.pagination.SimplePageMaster(id=1
    pageSequenceMaster: null
    mainFlow: instance of org.apache.fop.fo.pagination.Flow(id=1092)
    titleFO: null
    org.apache.fop.fo.FObj.propertyListTable: instance of org.apache.fop.fo.Property
    org.apache.fop.fo.FObj.propertyList: instance of org.apache.fop.fo.PropertyList(
    org.apache.fop.fo.FObj.propMgr: instance of org.apache.fop.fo.PropertyManager(id
    org.apache.fop.fo.FObj.id: null
    org.apache.fop.fo.FObj.children: instance of java.util.ArrayList(id=1098)
    org.apache.fop.fo.FObj.markers: null
    org.apache.fop.fo.FObj.systemId: "file:/path/to/fo-file"
    org.apache.fop.fo.FObj.line: 12
    org.apache.fop.fo.FObj.column: 49
    org.apache.fop.fo.FONode.parent: instance of org.apache.fop.fo.pagination.Root(i
    org.apache.fop.fo.FONode.name: "fo:page-sequence"
}
```

The page-sequence has one property, the reference to the page master:

```
 root.[1].propertyList = "{
  master-reference=org.apache.fop.fo.StringProperty@104e28b
}"

 root.[1].propertyList.get("master-reference") = {
    str: "simpleA4"
    org.apache.fop.fo.Property.specVal: null
}
```

The page-sequence has one child, the flow:

```
 root.[1].children = "[
  fo:flow at line 13:42
]"

 root.[1].[0] = "fo:flow at line 13:42"

 root.[1].[0] = {
    pageSequence: instance of org.apache.fop.fo.pagination.PageSequence(id=1081)
    markerSnapshot: null
    flowName: "xsl-region-body"
    contentWidth: 0
    org.apache.fop.fo.FObj.propertyListTable: instance of org.apache.fop.fo.Property
    org.apache.fop.fo.FObj.propertyList: instance of org.apache.fop.fo.PropertyList(
    org.apache.fop.fo.FObj.propMgr: instance of org.apache.fop.fo.PropertyManager(id
    org.apache.fop.fo.FObj.id: null
    org.apache.fop.fo.FObj.children: instance of java.util.ArrayList(id=1369)
    org.apache.fop.fo.FObj.markers: null
    org.apache.fop.fo.FObj.systemId: "file:/path/to/fo-file"
    org.apache.fop.fo.FObj.line: 13
    org.apache.fop.fo.FObj.column: 42
    org.apache.fop.fo.FONode.parent: instance of org.apache.fop.fo.pagination.PageSe
    org.apache.fop.fo.FONode.name: "fo:flow"
}
```

The flow has one property, the flow name:

```
 root.[1].[0].propertyList = "{
  flow-name=org.apache.fop.fo.StringProperty@6458a6
}"

 root.[1].[0].propertyList.get("flow-name") = {
    str: "xsl-region-body"
    org.apache.fop.fo.Property.specVal: null
}
```

The flow has one child, a block:

```
 root.children.elementData[1].children.elementData[0].children = "[
  fo:block at line 15:28
]"

 root.[1].[0].[0] = "fo:block at line 15:28"

 root.[1].[0].[0] = {
    align: 0
    alignLast: 0
    breakAfter: 0
    lineHeight: 0
    startIndent: 0
    endIndent: 0
    spaceBefore: 0
    spaceAfter: 0
    textIndent: 0
    keepWithNext: 0
    backgroundColor: null
    blockWidows: 0
    blockOrphans: 0
    id: null
    span: 59
    wsTreatment: 41
    lfTreatment: 98
    bWScollapse: true
    anythingLaidOut: false
    firstInlineChild: null
    org.apache.fop.fo.FObjMixed.textInfo: instance of org.apache.fop.fo.TextInfo(id=1377)
    org.apache.fop.fo.FObj.propertyListTable: instance of org.apache.fop.fo.Property$Maker
    org.apache.fop.fo.FObj.propertyList: instance of org.apache.fop.fo.PropertyList(id=137
    org.apache.fop.fo.FObj.propMgr: instance of org.apache.fop.fo.PropertyManager(id=1379)
    org.apache.fop.fo.FObj.id: null
    org.apache.fop.fo.FObj.children: instance of java.util.ArrayList(id=1380)
    org.apache.fop.fo.FObj.markers: null
    org.apache.fop.fo.FObj.systemId: "file:/path/to/fo-file"
    org.apache.fop.fo.FObj.line: 15
    org.apache.fop.fo.FObj.column: 28
    org.apache.fop.fo.FONode.parent: instance of org.apache.fop.fo.pagination.Flow(id=1092
    org.apache.fop.fo.FONode.name: "fo:block"
}

 root.[1].[0].[0].propertyList = "{
  font-size=org.apache.fop.fo.LengthProperty@ae4646
  font-weight=org.apache.fop.fo.StringProperty@187b287
  space-after=org.apache.fop.fo.SpaceProperty@1d9e2c7
}"
```

The block has two children:

```
 root.[1].[0].[0].children = "[
  fo:text at line 15:35
  fo:text at line 16:7
]"
```

```
 root.[1].[0].[0].[0] = "fo:text at line 15:35"

 root.[1].[0].[0].[0] = {
    ca: instance of char[7] (id=1386)
    start: 0
    length: 7
    textInfo: instance of org.apache.fop.fo.TextInfo(id=1377)
    lastFOTextProcessed: instance of org.apache.fop.fo.FOText(id=1387)
    prevFOTextThisBlock: null
    nextFOTextThisBlock: instance of org.apache.fop.fo.FOText(id=1387)
    ancestorBlock: instance of org.apache.fop.fo.flow.Block(id=1375)
    org.apache.fop.fo.FObj.propertyListTable: instance of org.apache.fop.fo.Property
    org.apache.fop.fo.FObj.propertyList: null
    org.apache.fop.fo.FObj.propMgr: null
    org.apache.fop.fo.FObj.id: null
    org.apache.fop.fo.FObj.children: null
    org.apache.fop.fo.FObj.markers: null
    org.apache.fop.fo.FObj.systemId: "file:/path/to/fo-file"
    org.apache.fop.fo.FObj.line: 15
    org.apache.fop.fo.FObj.column: 35
    org.apache.fop.fo.FONode.parent: instance of org.apache.fop.fo.flow.Block(id=137
    org.apache.fop.fo.FONode.name: "fo:text"
}
```

This text node contains the text "Test FO":

```
 root.[1].[0].[0].[0].ca = {
T, e, s, t,  , F, O
}

 root.[1].[0].[0].[1] = "fo:text at line 16:7"

 root.[1].[0].[0].[1] = {
    ca: instance of char[1] (id=1390)
    start: 0
    length: 1
    textInfo: instance of org.apache.fop.fo.TextInfo(id=1377)
    lastFOTextProcessed: instance of org.apache.fop.fo.FOText(id=1387)
    prevFOTextThisBlock: instance of org.apache.fop.fo.FOText(id=1384)
    nextFOTextThisBlock: null
    ancestorBlock: instance of org.apache.fop.fo.flow.Block(id=1375)
    org.apache.fop.fo.FObj.propertyListTable: instance of org.apache.fop.fo.Property
    org.apache.fop.fo.FObj.propertyList: null
    org.apache.fop.fo.FObj.propMgr: null
    org.apache.fop.fo.FObj.id: null
    org.apache.fop.fo.FObj.children: null
    org.apache.fop.fo.FObj.markers: null
    org.apache.fop.fo.FObj.systemId: "file:/path/to/fo-file"
    org.apache.fop.fo.FObj.line: 16
    org.apache.fop.fo.FObj.column: 7
    org.apache.fop.fo.FONode.parent: instance of org.apache.fop.fo.flow.Block(id=137
    org.apache.fop.fo.FONode.name: "fo:text"
}
```

This text node contains the text "\n":

```
 root.[1].[0].[0].[1].ca = {
}
```

8.3.3. The corresponding area tree

- PageViewport has a Page page and a Rectangle2D viewArea (reference/viewport pair).
- Page has five RegionViewports.
- RegionViewport has a RegionReference region and a Rectangle2D viewArea (reference/viewport pair).
- BodyRegion has a MainReference mainReference, a BeforeFloat beforeFloat, and a Footnote footnote.
- MainReference has a list of Spans.
- Span has a list of Flows.
- Flow has a list of Blocks.
- Block has a list of Blocks or LineAreas.
- LineArea has a list of InlineAreas.
- Text Area (subclass of InlineArea) has text.

The structure of the area tree is as follows:

```
PageViewport
|
+-Page
  |
  +-RegionViewport
    |
    +-BodyRegion
      |
      +-MainReference
      | |
      | +-Span
      | | |
      | | +-Flow
      | |
      | +-Span
      | | |
      | | +-Flow
      | |   |
      | |   +-Block
      | |   | |
      | |   | +-LineArea
      | |   |   |
      | |   |   +-TextArea
      | |   |
      | |   +-Block
      | |
      | +-Span
      | | |
      | | +-Flow
      | |
      | +-Span
      | | |
      | | +-Flow
      | |
      | +-Span
      |   |
      |   +-Flow
      |
      +blocks
      |
      +CTM
```

In the listing below members of an `arraylist` are indicated by `[n]`, which stands for `get(n)`. If the `arraylist` is called `children`, the word `children` has been omitted, so that `[n]` then stands for `children.get(n)`.

Type: `org.apache.fop.area.PageViewport`:

```
 curPage = "PageViewport: page=1"

 curPage = {
    page: instance of org.apache.fop.area.Page(id=1394)
    viewArea: instance of java.awt.Rectangle(id=1395)
    clip: false
    pageNumber: "1"
    idReferences: null
    unresolved: null
    pendingResolved: null
    markerFirstStart: null
    markerLastStart: null
    markerFirstAny: null
    markerLastEnd: null
    markerLastAny: null
}
```

Type: `org.apache.fop.area.Page`:

```
 curPage.page = {
    regionBefore: null
    regionStart: null
    regionBody: instance of org.apache.fop.area.RegionViewport(id=1397)
    regionEnd: null
    regionAfter: null
    unresolved: null
}
```

Type: `org.apache.fop.area.RegionViewport`:

```
 curPage.page.regionBody = {
    region: instance of org.apache.fop.area.BodyRegion(id=1077)
    viewArea: instance of java.awt.Rectangle(id=1399)
    clip: false
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 0
    org.apache.fop.area.Area.props: null
}
```

Type: `org.apache.fop.area.BodyRegion`:

```
 curPage.page.regionBody.region = {
    beforeFloat: null
    mainReference: instance of org.apache.fop.area.MainReference(id=1401)
    footnote: null
    columnGap: 18000
    columnCount: 1
    refIPD: 0
    org.apache.fop.area.RegionReference.regionClass: 2
    org.apache.fop.area.RegionReference.ctm: instance of org.apache.fop.area.CTM(id=
    org.apache.fop.area.RegionReference.blocks: instance of java.util.ArrayList(id=14
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 0
    org.apache.fop.area.Area.props: null
}
```

Type: `org.apache.fop.area.MainReference`:

```
 curPage.page.regionBody.region.mainReference = {
    spanAreas: instance of java.util.ArrayList(id=1405)
    columnGap: 0
    width: 0
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 0
    org.apache.fop.area.Area.props: null
}
```

The main reference contains five span areas. Four are empty. Number 1 contains the text of this page.

```
 curPage.page.regionBody.region.mainReference.spanAreas = "[
  org.apache.fop.area.Span@53c3f5
  org.apache.fop.area.Span@101ac93
  org.apache.fop.area.Span@125d61e
  org.apache.fop.area.Span@155d3a3
  org.apache.fop.area.Span@718242
]"
```

Type: `org.apache.fop.area.Span`:

```
 curPage.page.regionBody.region.mainReference.spanAreas[0] = {
    flowAreas: instance of java.util.ArrayList(id=1409)
    height: 0
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 481891
    org.apache.fop.area.Area.props: null
}
```

```
 curPage.page.regionBody.region.mainReference.spanAreas[0].flowAreas = "[
  org.apache.fop.area.Flow@e33e18
]"
```

Type: `org.apache.fop.area.Flow`:

```
 curPage.page.regionBody.region.mainReference.spanAreas[0].flowAreas[0] = {
    stacking: 2
    width: 0
    org.apache.fop.area.BlockParent.xOffset: 0
    org.apache.fop.area.BlockParent.yOffset: 0
    org.apache.fop.area.BlockParent.width: 0
    org.apache.fop.area.BlockParent.height: 0
    org.apache.fop.area.BlockParent.children: null
    org.apache.fop.area.BlockParent.orientation: 0
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 481891
    org.apache.fop.area.Area.props: null
}
```

Type: `org.apache.fop.area.Span`:

```
 curPage.page.regionBody.region.mainReference.spanAreas[1] = {
    flowAreas: instance of java.util.ArrayList(id=1412)
```

```
    height: 0
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 481891
    org.apache.fop.area.Area.props: null
}


 curPage.page.regionBody.region.mainReference.spanAreas[0].flowAreas = "[
  org.apache.fop.area.Flow@e33e18
]"
```

Type: `org.apache.fop.area.Flow`:

```
 curPage.page.regionBody.region.mainReference.spanAreas[1].flowAreas[0] = {
    stacking: 2
    width: 0
    org.apache.fop.area.BlockParent.xOffset: 0
    org.apache.fop.area.BlockParent.yOffset: 0
    org.apache.fop.area.BlockParent.width: 0
    org.apache.fop.area.BlockParent.height: 0
    org.apache.fop.area.BlockParent.children: instance of java.util.ArrayList(id=141
    org.apache.fop.area.BlockParent.orientation: 0
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 481891
    org.apache.fop.area.Area.props: null
}


 curPage.page.regionBody.region.mainReference.spanAreas[1].flowAreas[0].children = "
  org.apache.fop.area.Block@61f533
  org.apache.fop.area.Block@12922f6
]"
```

Type: `org.apache.fop.area.Block`:

```
 curPage.page.regionBody.region.mainReference.spanAreas[1].flowAreas[0].[0] = {
    stacking: 2
    positioning: 0
    org.apache.fop.area.BlockParent.xOffset: 0
    org.apache.fop.area.BlockParent.yOffset: 0
    org.apache.fop.area.BlockParent.width: 481891
    org.apache.fop.area.BlockParent.height: 19200
    org.apache.fop.area.BlockParent.children: instance of java.util.ArrayList(id=141
    org.apache.fop.area.BlockParent.orientation: 0
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 481891
    org.apache.fop.area.Area.props: null
}


 curPage.page.regionBody.region.mainReference.spanAreas[1].flowAreas[0].[0].children
  org.apache.fop.area.LineArea@9f0d
]"
```

Type: `org.apache.fop.area.LineArea`:

```
 curPage.page.regionBody.region.mainReference.spanAreas[1].flowAreas[0].[0].[0] = {
    stacking: 0
    startIndent: 0
```

```
    length: 0
    lineHeight: 19200
    baseLine: 0
    inlineAreas: instance of java.util.ArrayList(id=1422)
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 0
    org.apache.fop.area.Area.props: null
}
```

```
 curPage.page.regionBody.region.mainReference.spanAreas[1].flowAreas[0].[0].[0].inlineArea
  org.apache.fop.area.inline.TextArea@21d23b
]"
```

Type: `org.apache.fop.area.inline.TextArea`:

```
 curPage.page.regionBody.region.mainReference.spanAreas[1].flowAreas[0].[0].[0].inlineArea
    text: "Test FO"
    iTSadjust: 0
    org.apache.fop.area.inline.InlineArea.height: 14800
    org.apache.fop.area.inline.InlineArea.contentIPD: 59568
    org.apache.fop.area.inline.InlineArea.verticalPosition: 13688
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 0
    org.apache.fop.area.Area.props: instance of java.util.HashMap(id=1426)
}
```

```
 curPage.page.regionBody.region.mainReference.spanAreas[1].flowAreas[0].[0].[0].inlineArea
  4=16000
  3=F3
  7=#000000
}"
```

```
3  org.apache.fop.area.Trait.FONT_NAME

4  org.apache.fop.area.Trait.FONT_SIZE

7  org.apache.fop.area.Trait.COLOR
```

Type: `org.apache.fop.area.Block`:

```
 curPage.page.regionBody.region.mainReference.spanAreas[1].flowAreas[0].[1] = {
    stacking: 2
    positioning: 0
    org.apache.fop.area.BlockParent.xOffset: 0
    org.apache.fop.area.BlockParent.yOffset: 0
    org.apache.fop.area.BlockParent.width: 0
    org.apache.fop.area.BlockParent.height: 14173
    org.apache.fop.area.BlockParent.children: null
    org.apache.fop.area.BlockParent.orientation: 0
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 0
    org.apache.fop.area.Area.props: null
}
```

Type: `org.apache.fop.area.Span`:

```
 curPage.page.regionBody.region.mainReference.spanAreas[2] = {
    flowAreas: instance of java.util.ArrayList(id=1429)
    height: 0
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 481891
    org.apache.fop.area.Area.props: null
}
```

```
 curPage.page.regionBody.region.mainReference.spanAreas[2].flowAreas = "[
  org.apache.fop.area.Flow@c72243
]"
```

Type: `org.apache.fop.area.Flow`:

```
 curPage.page.regionBody.region.mainReference.spanAreas[2].flowAreas[0] = {
    stacking: 2
    width: 0
    org.apache.fop.area.BlockParent.xOffset: 0
    org.apache.fop.area.BlockParent.yOffset: 0
    org.apache.fop.area.BlockParent.width: 0
    org.apache.fop.area.BlockParent.height: 0
    org.apache.fop.area.BlockParent.children: null
    org.apache.fop.area.BlockParent.orientation: 0
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 481891
    org.apache.fop.area.Area.props: null
}
```

Type: `org.apache.fop.area.Span`:

```
 curPage.page.regionBody.region.mainReference.spanAreas[3] = {
    flowAreas: instance of java.util.ArrayList(id=1433)
    height: 0
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 481891
    org.apache.fop.area.Area.props: null
}
```

```
 curPage.page.regionBody.region.mainReference.spanAreas[3].flowAreas = "[
  org.apache.fop.area.Flow@dc9766
]"
```

Type: `org.apache.fop.area.Flow`:

```
 curPage.page.regionBody.region.mainReference.spanAreas[3].flowAreas[0] = {
    stacking: 2
    width: 0
    org.apache.fop.area.BlockParent.xOffset: 0
    org.apache.fop.area.BlockParent.yOffset: 0
    org.apache.fop.area.BlockParent.width: 0
    org.apache.fop.area.BlockParent.height: 0
    org.apache.fop.area.BlockParent.children: null
    org.apache.fop.area.BlockParent.orientation: 0
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 481891
    org.apache.fop.area.Area.props: null
}
```

Type: `org.apache.fop.area.Span`:

```
 curPage.page.regionBody.region.mainReference.spanAreas[4] = {
    flowAreas: instance of java.util.ArrayList(id=1436)
    height: 0
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 481891
    org.apache.fop.area.Area.props: null
}
```

```
 curPage.page.regionBody.region.mainReference.spanAreas[4].flowAreas = "[
  org.apache.fop.area.Flow@1ec58a
]"
```

Type: `org.apache.fop.area.Flow`:

```
 curPage.page.regionBody.region.mainReference.spanAreas[4].flowAreas[0] = {
    stacking: 2
    width: 0
    org.apache.fop.area.BlockParent.xOffset: 0
    org.apache.fop.area.BlockParent.yOffset: 0
    org.apache.fop.area.BlockParent.width: 0
    org.apache.fop.area.BlockParent.height: 0
    org.apache.fop.area.BlockParent.children: null
    org.apache.fop.area.BlockParent.orientation: 0
    org.apache.fop.area.Area.areaClass: 0
    org.apache.fop.area.Area.ipd: 481891
    org.apache.fop.area.Area.props: null
}
```

```
 curPage.page.regionBody.region.blocks = "[]"
```

Type: `org.apache.fop.area.CTM`:

```
 curPage.page.regionBody.region.ctm = {
    a: 1.0
    b: 0.0
    c: 0.0
    d: 1.0
    e: 56692.0
    f: 56692.0
    CTM_LRTB: instance of org.apache.fop.area.CTM(id=1439)
    CTM_RLTB: instance of org.apache.fop.area.CTM(id=1440)
    CTM_TBRL: instance of org.apache.fop.area.CTM(id=1441)
}
```

Chapter 9

Properties

9.1. The basic setup of the property subsystem

9.1.1. Property values

The FO nodes in the FO tree contain property values as specified by the user. Each property value is represented by a object of type `org.apache.fop.fo.Property`, or of a subtype thereof.

There are various types of property values: `CharacterProperty`, `ColorTypeProperty`, `CondLengthProperty`, `EnumProperty`, `KeepProperty`, `LengthPairProperty`, `LengthProperty`, `LengthRangeProperty`, `ListProperty`, `NCnameProperty`, `NumberProperty`, `NumericProperty`, `RelativeNumericProperty`, `StringProperty`. The type `ToBeImplementedProperty` is used for properties that are not yet implemented. Some of these types have subtypes: `AutoLength`, `FixedLength`, `PercentLength`, `TableColLength` are subclasses of `LengthProperty`; `SpaceProperty` is a subclass of `LengthRangeProperty`. Each of these types is a subtype of `org.apache.fop.fo.Property`.

Property values may implement one or more of the interfaces defined in the package `org.apache.fop.datatypes`: `Numeric`, `Length`, `ColorType`

Some properties actually represent a set of properties, such as a minimum, an optimum and a maximum. These are represented by a property value which implements the `CompoundDatatype` interface. They contain a property value for each member property.

9.1.2. The property list of an FO node

Property values are held by the FO node corresponding to the fo element on which they are specified by the user. FO nodes contain a property list of type `PropertyList`, which extends `HashMap`.

The property types are known by the property name as used on the FO element, e.g. `font-size`. For efficiency of implementation, each type of property type is also known by an `integer`, the `propID`. The `propID`s are defined in the interface `org.apache.fop.fo.Constants`, which gives them a symbolic name of the form `PR_` + property name in capitals and spaces replaced by underscores, e.g. `PR_FONT_SIZE`. Wherever possible, the code uses the `propID`s instead of the property names.

When an FO requests a property, it does so by `propId`. The request is eventually answered by `PropertyList.getExplicitBaseProp`, but before it can do so, it has to retrieve the property name from `FOPropertyMapping.getPropertyName`. A particular inefficiency as a consequence is found in `FopPropValFunction.java` and some other classes in the `fo.expr` package:

```
return pInfo.getPropertyList().get(FOPropertyMapping.getPropertyId(propName))
```

Here `propName` -> `propId` mapping is done, which later is reverted again.

```
[1] org.apache.fop.fo.FOPropertyMapping.getPropertyName (FOPropertyMapping.java:48
[2] org.apache.fop.fo.PropertyList.getExplicitBaseProp (PropertyList.java:230)
[3] org.apache.fop.fo.Property$Maker.findProperty (Property.java:281)
[4] org.apache.fop.fo.Property$Maker.get (Property.java:314)
[5] org.apache.fop.fo.PropertyList.get (PropertyList.java:281)
[6] org.apache.fop.fo.PropertyList.get (PropertyList.java:267)
[7] org.apache.fop.fo.FObj.getProperty (FObj.java:261)
```

9.1.3. Property makers

Property value objects are created by a property maker of type `org.apache.fop.fo.PropertyMaker`, or of a subtype thereof. For each property type there is a property maker object, which knows the property type, its default value, and some other characteristics.

The types of property makers are: `CharacterProperty.Maker`, `ColorTypeProperty.Maker`, `CompoundPropertyMaker`, `EnumProperty.Maker`, `LengthProperty.Maker`, `ListProperty.Maker`, `NumberProperty.Maker`, `StringProperty.Maker`, and `ToBeImplementedProperty.Maker`.

The property makers are lazily constructed when the `FObj` constructor wants to create its static member `propertyListTable`. The constructor calls `FOPropertyMapping.getGenericMappings()`, which constructs and returns `Property.Maker[Constants.PROPERTY_COUNT+1] s_generics`. The FObj constructor then copies this array of `PropertyMaker`s into `propertyListTable`.

`public static PropertyMaker[] getGenericMappings()` first creates the shorthand property makers, so that they can be used in the creation of the makers of the real properties, and a set of generic property makers, which act as templates for the real property makers. Next it creates the makers for all property types. Related property types are grouped, e.g. `createFontProperties()`.

An example is the creation of the maker for the `font-size` property type:

```
m  = new LengthProperty.Maker(PR_FONT_SIZE);
m.setInherited(true);
m.setDefault("12pt");
m.setPercentBase(LengthBase.INH_FONTSIZE);
addPropertyMaker("font-size", m);
```

Since `font-size` is a length, its maker is a `LengthProperty.Maker`. It is inherited, and its default value is 12 pt. If the user specifies the `font-size` value as a percentage, then the actual value is calculated from the `font-size` value inherited from the parent FO node.

9.1.4. Shorthand properties

9.1.4.1. Overview

Shorthand properties are properties which are shorthand for a number of properties. In other words, they specify the value of a number of properties in a single attribute. All shorthand properties can take a list of values, which are space separated in the FO file. The FO spec specifies how this list of values determines the values of the properties for which this is the shorthand (the target properties.). The length of the list of values for a single shorthand property may vary. For each length the attribution of these values to the target properties is different.

When the FO tree is constructed, shorthand property values are parsed and stored like any other property value. Because the value can be a list, it is always of type `ListProperty`.

The meaning of shorthand properties is only dealt with when the value of one of the target properties is retrieved. For that purpose each target property maker knows the shorthand properties that may set its value, and when the target property value is retrieved, its maker checks with each of its shorthand property makers if it has a value. Note that the value of a shorthand property is never retrieved directly, because shorthand properties have no direct meaning for the layout.

When the shorthand property value has been retrieved, the value for the target property must be extracted from the list. That is done by a shorthand parser, which implements `ShorthandParser`. There are two implementing types: `GenericShorthandParser` and `BoxPropShorthandParser`. Their method `convertValueForProperty` knows how each specified value determines the value of the possible target properties. A shorthand parser object is added to the shorthand property maker when the maker is created.

Note that `CompoundPropertyMaker` also has a member `shorthandMaker`. I am not sure if this has anything to do with shorthand properties. It seems more related to `CompoundPropertyMaker` delegating certain tasks to a subproperty maker, viz. the one which is the `shorthandMaker`.

9.1.4.2. Example of a shorthand property

The property `margin` is shorthand for the four properties `margin-top`, `margin-right`, `margin-bottom`, `margin-left`. Its value can consist of 1 to 4 width values.

When the property maker for `margin` is created, it gets a `BoxPropShorthandParser` as shorthand parser:

```
m  = new ListProperty.Maker(PR_MARGIN);
m.setInherited(false);
m.setDefault("");
m.setDatatypeParser(new BoxPropShorthandParser());
m.setPercentBase(LengthBase.BLOCK_WIDTH);
addPropertyMaker("margin", m);
```

When the property maker for `margin-top` is created, the `margin` maker is registered with it as a shorthand maker:

```
m  = new LengthProperty.Maker(PR_MARGIN_TOP);
m.setInherited(false);
m.setDefault("0pt");
m.addShorthand(s_generics[PR_MARGIN]);
m.setPercentBase(LengthBase.BLOCK_WIDTH);
addPropertyMaker("margin-top", m);
```

The maker for `border-top-width` has three shorthands: `border-top`, `border-width`, and `border`:

```
this.shorthands = instance of org.apache.fop.fo.properties.PropertyMaker[3] (id=772
this.shorthands[0] = "org.apache.fop.fo.properties.ListProperty$Maker@1e1dadb"
this.shorthands[0].propId = 52
this.shorthands[1] = "org.apache.fop.fo.properties.ListProperty$Maker@bac9b9"
this.shorthands[1].propId = 56
this.shorthands[2] = "org.apache.fop.fo.properties.ListProperty$Maker@8ceeea"
this.shorthands[2].propId = 18
```

9.1.4.3. Parsing a shorthand property

The value of a shorthand property is parsed and the value of a target property is extracted in this call stack:

```
[1] org.apache.fop.fo.BoxPropShorthandParser.convertValueForProperty (BoxPropShortl
[2] org.apache.fop.fo.GenericShorthandParser.getValueForProperty (GenericShorthandl
[3] org.apache.fop.fo.properties.PropertyMaker.getShorthand (PropertyMaker.java:61
[4] org.apache.fop.fo.properties.PropertyMaker.findProperty (PropertyMaker.java:27
[5] org.apache.fop.fo.properties.PropertyMaker.get (PropertyMaker.java:305)
[6] org.apache.fop.fo.PropertyList.get (PropertyList.java:282)
[7] org.apache.fop.fo.PropertyList.get (PropertyList.java:268)
[8] org.apache.fop.fo.PropertyManager.getMarginProps (PropertyManager.java:301)
```

The extraction proceeds as follows:

- `PropertyMaker.getShorthand`
  - `parser.getValueForProperty(propId, listprop, propertyMaker, propertyList)`; `propId` is the ID of the target property, `listprop` is the shorthand property value, of type `ListProperty`, which was retrieved
    - if the shorthand value is `inherit`, get the value for the target property from the parent.
    - else `convertValueForProperty(propId, listProperty, maker, propertyList)`
      - get from the shorthand list of values the value that corresponds to the target property
      - if the retrieved value is not `null`, convert the property, `maker.convertShorthandProperty(propertyList, p, null)`

- first try to convert it in the normal way: `maker.convertProperty(prop, propertyList, fo)`
- if this gives a `null` value, test if the value is an enumerated value or a keyword; if so, process it.

9.1.5. Corresponding properties

A number of traits can be specified by two alternative properties, e.g. `border-left-width` and `border-start-width`. These are called corresponding properties. One of a pair of corresponding properties is an absolute property, the other is a relative property. The meaning of the relative property depends on the writing mode. When the value of a property is retrieved that has a corresponding property, the value of that corresponding property should also be taken into account.

Corresponding properties are registered with the property maker when it is created:

```
bwm  = new BorderWidthPropertyMaker(PR_BORDER_LEFT_WIDTH);
bwm.useGeneric(genericBorderWidth);
bwm.setBorderStyleId(PR_BORDER_LEFT_STYLE);
bwm.addShorthand(s_generics[PR_BORDER_LEFT]);
bwm.addShorthand(s_generics[PR_BORDER_WIDTH]);
bwm.addShorthand(s_generics[PR_BORDER]);
corr = new CorrespondingPropertyMaker(bwm);
corr.setCorresponding(PR_BORDER_START_WIDTH, PR_BORDER_END_WIDTH,
        PR_BORDER_AFTER_WIDTH);
addPropertyMaker("border-left-width", bwm);
```

There are always three corresponding properties, for the three writing modes `lr_tb`, `rl_tb`, `tb_rl`, in this order:

```
 corr = {
    baseMaker: instance of org.apache.fop.fo.properties.BorderWidthPropertyMaker(id=702)
    lr_tb: 50
    rl_tb: 36
    tb_rl: 22
    useParent: false
    relative: false
}
```

When a property value is retrieved, the value of the corresponding property may have priority. This is determined by the method `corresponding.isCorrespondingForced()`. This is true if

- this is a relative property
- and the corresponding property has been explicitly specified on this FO node

Relative properties are marked by the fact that their corresponding property maker has its member `relative` set to `true`; this is set when the property is created.

If the corresponding property has priority, its value is computed. Otherwise, if the value of the property itself has been explicitly specified on this FO node, it is used. Otherwise, the corresponding property is computed. Computation in this connection means that also the shorthand properties are checked.

Because shorthand properties only exist for absolute properties, the values are effectively checked in this order:

- An absolute property
  - The explicit value of this property.
  - The explicit value of the corresponding property.
  - The value of this property from the shorthand properties.
- A relative property
  - The explicit value of the corresponding property.
  - The explicit value of this property.
  - The value of the corresponding property from the shorthand properties.

9.1.6. Mapping between property names, IDs and makers

The property subsystem is set up in the class `FOPropertyMapping`. It creates a property maker object for each property type, and it creates mappings of the names, IDs and makers of the property types. It holds the following static maps:

```
property name <=> property ID     => property maker
                  |                       |
    s_htSubPropNames (<-)     s_htGeneric
        s_htPropIds      (->)
```

Each type of FObj holds a copy of `s_htGeneric` as its static member `FObj.propertyListTable`. According to design documents an FObj type may have its own specific makers for certain property types. Probably this is the reason that `FObj` holds its own copy of the list of makers. This allows subclasses to hold their own modified copy. As far as I know, this is not currently the case.

The mappings are filled in the static method

```
    private static void addPropertyMaker(String name, Property.Maker maker) {
        s_generics[maker.getPropId()] = maker;
        s_htPropNames.put(name, new Integer(maker.getPropId()));
        s_htPropIds.put(new Integer(maker.getPropId()), name);
    }
```

which is called for each property type.

The constants for property IDs are defined in the interface `org.apache.fop.fo.Constants`:

```
    int PR_ABSOLUTE_POSITION = 1;
    int PR_ACTIVE_STATE = 2;
        ...
    int PR_FONT_SIZE = 94;
    ...
    int PROPERTY_COUNT = 247;
```

Composite properties are defined by a compound number:

```
    int COMPOUND_SHIFT = 9;
    int CP_MAXIMUM = 5 << COMPOUND_SHIFT;
    int CP_MINIMUM = 6 << COMPOUND_SHIFT;
    int CP_OPTIMUM = 7 << COMPOUND_SHIFT;
        ...
```

Enumerated property values are also defined here:

```
    int ABSOLUTE = 1;
    int ABSOLUTE_COLORMETRIC = 2;
    ...
    int VISIBLE = 105;
    int WRAP = 106;
```

For fast access to important characteristic of property inheritance, `PropertyList` maintains a static array `boolean[Constants.PROPERTY_COUNT + 1] inheritableProperty`, which lists for each property type if it is inherited. It is constructed by asking the maker of each property type if it is inherited.

A few members of the array of `PropertyMaker`s `s_generics`. It is indexed by the `propID`. Member 0 is

null, and serves for unknown property types. Member 1 is for absolute-position, member 94 for font-size.

```
main[1] print org.apache.fop.fo.FOPropertyMapping.s_generics
 org.apache.fop.fo.FOPropertyMapping.s_generics = instance of org.apache.fop.fo.Property$Ma
main[1] print org.apache.fop.fo.FOPropertyMapping.s_generics[0]
 org.apache.fop.fo.FOPropertyMapping.s_generics[0] = null
main[1] print org.apache.fop.fo.FOPropertyMapping.s_generics[1]
 org.apache.fop.fo.FOPropertyMapping.s_generics[1] = "org.apache.fop.fo.EnumProperty$Maker(
main[1] print org.apache.fop.fo.FOPropertyMapping.s_generics[94]
 org.apache.fop.fo.FOPropertyMapping.s_generics[94] = "org.apache.fop.fo.LengthProperty$Ma
```

A few members of the mapping s_htPropIds from propID to property name. The s_htPropIds for compound properties are shifted:

```
main[1] print org.apache.fop.fo.FOPropertyMapping.s_htPropIds
 org.apache.fop.fo.FOPropertyMapping.s_htPropIds = "{
  1=absolute-position
  ...
  94=font-size
  ...
  247=z-index
  512=block-progression-direction
  1024=conditionality
  ...
  5632=within-page
 }"
```

A few members of the mappings s_htPropNames and s_htSubPropNames from property name to propID. The propIds for compound properties are shifted:

```
main[1] print org.apache.fop.fo.FOPropertyMapping.s_htPropNames
 org.apache.fop.fo.FOPropertyMapping.s_htPropNames = "{
  absolute-position=1
  ...
  font-size=94
  ...
  z-index=247
 }"


main[1] print org.apache.fop.fo.FOPropertyMapping.s_htSubPropNames
 org.apache.fop.fo.FOPropertyMapping.s_htSubPropNames = "{
  block-progression-direction=512
  conditionality=1024
  ...
  within-page=5632
 }"
```

9.1.7. Storing the property values based on their PropID

The class PropertySets contains a setup by which property values may be retrieved by their PropId instead of their name. In this setup PropertyList no longer extends HashMap but contains an array of property objects, called values. In order to prevent that each FObj should contain an array of size Constants.PROPERTY_COUNT, a mapping is setup from a static array for all FO types and all property types to an index for the possible properties of an FO type, PropertySets.mapping.

PropertySets.mapping is a short[Constants.ELEMENT_COUNT+1][] matrix, which for each FO type contains a mapping from PropID to a sparse array of indices, which enumerates the properties that are

valid for this FO type.

For an element `fo:bar` which supports 2 properties, `foo`, whose `PropID` is 21, and `baz`, whose `PropID` is 137, the array of indices has the values

```
indices[21] = 1
indices[137] = 2
```

and all other values are 0. Here indices denotes the row in `mapping` which corresponds to FO type bar.

The values are indices into the array `PropertyList.values`, which then looks like this:

```
values[0] = null // always null.
values[1] = reference to a 'foo' Property instance
values[2] = reference to a 'baz' Property instance
```

Example of `PropertySets.mapping`:

```
PropID ->         | 0 1 2 3 4 5 ... (Contants.PR_XXX)
Element |         |
        v         |
------------------|-------------------------------------------------
FO_BASIC_LINK     | 2 0 1 0 2 0
FO_BIDI_OVERRIDE  | 3 0 0 1 2 3
FO_BLOCK          | 2 1 0 0 2 0
...               | ....
                  |
```

A property value of an `FONode` can then be retrieved as `PropertyList.values[indices[propId]]`, where `indices = PropertySets.getPropertySet(elementId) = PropertySets.mapping[elementId].`

The matrix `PropertySets.mapping` is constructed in the routine `PropertySets.initialize()`.

First it constructs the `elements` array. For each FO type this array contains an `Element` object. This object contains a list of properties which are valid for this type of FO, and a list of child `Element` objects. Each child `Element` object corresponds to an FO type that may occur as a child of this FO type.

Then the `for (boolean dirty = true; dirty; )` loop is executed. It effect is as follows (from an email by Finn Bock). For each FO type the `BitSet` of allowed properties is merged with the `BitSet` of allowed properties of its possible direct children. When for any FO type the `merge` subroutine modifies its `BitSet`, it sets the boolean variable `dirty` to `true` to signal that another iteration of the loop is required. By iterating over the loop until no further modifications are made, one makes sure that the merging process propagates from below to the top, that is, from any FO type to its farthest possible ancestor. This ensures that every FO type registers the allowed properties of itself and of all FO types that may ever appear in its subtree.

The matrix `PropertySets.mapping` is still not used in `PropertyList`, and the array `values` does not yet exist (19 May 2004). The properties are held by name in `PropertyList`, which extends `HashMap`.

9.2. Creating a property value

9.2.1. General

A property value is created by the maker for the property type, in its method `PropertyMaker.make(PropertyList, String, FObj)`, where the second argument is the property value as a string:

- If the specified value is `inherit`, get the property value from the parent FObj.
- Else if the value is an enumerated value, the corresponding property value is retrieved (for each possible

enumerated value only one property value object exists, of type EnumProperty).
- If this does not retrieve a property value,
  - If the value is a shorthand keyword, it is converted to the corresponding value.
  - The value is parsed, and a property value is created.
  - The method `PropertyMaker.convertProperty` is called, which is overridden in subclasses of PropertyMaker. CompoundPropertyMaker uses this method to convert the simple property value constructed to a compound property value:
    - Make a compound property value based on default values: `PropertyMaker.makeCompound`, overridden in CompoundPropertyMaker.
    - Set all components equal to the simple property value that is being converted.
- If this is a compound property maker, convert it to a compound property as above. (Is this the second time the property value is converted?)

The property may also record the value as specified in the fo element, as this may influence the traits of the areas created by this FO node and FO nodes in the subtree.

Subclasses of `PropertyMaker` may override this method. For example, `StringProperty.Maker` has a much simpler method.

Attributes of the fo elements are converted to property value objects in `PropertyList.convertAttributeToProperty`:

- If the property is not a component of a compound property,
  - Ask the maker for the property to create the property value.
- Else if the property is a component of a compound property,
  - Find the base property by a call to `Propertylist.findBaseProperty`:
    - If the base property value already exists, return it to `convertAttributeToProperty`.
    - If the base attribute is also specified (later) in the list of attributes, retrieve it, ask the maker for the base property to create the base property value, and return it to `convertAttributeToProperty`.
    - Return null to `convertAttributeToProperty`.
  - Ask the maker for the subproperty to create the subproperty value by a call to `PropertyMaker.make(Property, int, PropertyList, String, FObj)`, where the second argument is the subproperty ID and the fourth argument is the specified value. This method is overridden in `CompoundPropertyMaker`:
    - If the base property value does not yet exist, ask its maker to create it with default values for the components: `PropertyMaker.makeCompound`, which is overridden in `CompoundPropertyMaker`:
      - Create an empty property value.
      - Create property values for the subproperties with default values, and insert them into the compound property value.
    - Create the specified subproperty value and insert it into the compound property value, where it replaces the default subproperty value.
- Add the property to the property list.

9.2.2. Example of a compound property

In this example we illustrate the case where first the base attribute of a compound property is specified, and then the attribute for one of the components.

First the user specifies the attribute value `leader-length="120pt"`.

A simple length property value is constructed first:

```
 p.getClass() = "class org.apache.fop.fo.properties.FixedLength"
 p = {
    millipoints: 120000
    org.apache.fop.fo.properties.Property.specVal: null
}
 p = "120000mpt"
```

Then it is converted into a compound property value. First a compound property with default component values is created:

```
 p.getClass() = "class org.apache.fop.fo.properties.LengthRangeProperty"
 p = {
    minimum: instance of org.apache.fop.fo.properties.FixedLength(id=759)
    optimum: instance of org.apache.fop.fo.properties.FixedLength(id=760)
    maximum: instance of org.apache.fop.fo.properties.PercentLength(id=761)
    bfSet: 0
    bChecked: true
    org.apache.fop.fo.properties.Property.specVal: null
}
 p = "LengthRange[min:0mpt, max:100.0%, opt:12000mpt]"
```

Then all components are set equal to the simple property value:

```
 prop.getClass() = "class org.apache.fop.fo.properties.LengthRangeProperty"
 prop = {
    minimum: instance of org.apache.fop.fo.properties.FixedLength(id=744)
    optimum: instance of org.apache.fop.fo.properties.FixedLength(id=744)
    maximum: instance of org.apache.fop.fo.properties.FixedLength(id=744)
    bfSet: 7
    bChecked: true
    org.apache.fop.fo.properties.Property.specVal: null
}
 prop = "LengthRange[min:120000mpt, max:120000mpt, opt:120000mpt]"
```

The property makers involved:

```
 this = "org.apache.fop.fo.properties.LengthRangeProperty$Maker@55a338"
 this.subproperties = instance of org.apache.fop.fo.properties.PropertyMaker[11] (id
 getSubpropMaker(org.apache.fop.fo.Constants.CP_MINIMUM) = "org.apache.fop.fo.proper
 getSubpropMaker(org.apache.fop.fo.Constants.CP_MAXIMUM) = "org.apache.fop.fo.proper
 getSubpropMaker(org.apache.fop.fo.Constants.CP_OPTIMUM) = "org.apache.fop.fo.proper
```

Stack dump when making the compound property:

```
  [1] org.apache.fop.fo.properties.CompoundPropertyMaker.makeCompound (CompoundPrope
  [2] org.apache.fop.fo.properties.CompoundPropertyMaker.convertProperty (CompoundPr
  [3] org.apache.fop.fo.properties.LengthRangeProperty$Maker.convertProperty (Length
  [4] org.apache.fop.fo.properties.PropertyMaker.make (PropertyMaker.java:392)
  [5] org.apache.fop.fo.properties.CompoundPropertyMaker.make (CompoundPropertyMaker
  [6] org.apache.fop.fo.PropertyList.convertAttributeToProperty (PropertyList.java:4
  [7] org.apache.fop.fo.PropertyList.addAttributesToList (PropertyList.java:374)
  [8] org.apache.fop.fo.FObj.addProperties (FObj.java:133)
  [9] org.apache.fop.fo.FObj.processNode (FObj.java:96)
  [10] org.apache.fop.fo.FOTreeBuilder.startElement (FOTreeBuilder.java:234)
```

Subsequently, the user specifies a component, `leader-length.maximum="200pt"`.

First the subproperty is constructed as a simple length property:

```
 p.getClass() = "class org.apache.fop.fo.properties.FixedLength"
 p = {
    millipoints: 200000
    org.apache.fop.fo.properties.Property.specVal: null
}
 p = "200000mpt"
```

Then it is added to the compound property as the component `maximum`:

```
 prop.getClass() = "class org.apache.fop.fo.properties.LengthRangeProperty"
 prop = {
    minimum: instance of org.apache.fop.fo.properties.FixedLength(id=755)
    optimum: instance of org.apache.fop.fo.properties.FixedLength(id=755)
    maximum: instance of org.apache.fop.fo.properties.FixedLength(id=767)
    bfSet: 7
    bChecked: true
    org.apache.fop.fo.properties.Property.specVal: null
}
 prop = "LengthRange[min:120000mpt, max:200000mpt, opt:120000mpt]"
```

Stack dump when making the property:

```
  [1] org.apache.fop.fo.properties.PropertyMaker.make (PropertyMaker.java:378)
  [2] org.apache.fop.fo.properties.CompoundPropertyMaker.make (CompoundPropertyMaker.java:
  [3] org.apache.fop.fo.PropertyList.convertAttributeToProperty (PropertyList.java:423)
  [4] org.apache.fop.fo.PropertyList.addAttributesToList (PropertyList.java:374)
  [5] org.apache.fop.fo.FObj.addProperties (FObj.java:133)
  [6] org.apache.fop.fo.FObj.processNode (FObj.java:96)
  [7] org.apache.fop.fo.FOTreeBuilder.startElement (FOTreeBuilder.java:234)
```

9.2.3. Enumerated property values

The interface `Constants` defines values for each possible enumerated value of a property:

```
    int ABSOLUTE = 1;
    int ABSOLUTE_COLORMETRIC = 2;
    ...
    int VISIBLE = 105;
    int WRAP = 106;
```

In `FOPropertyMapping` a property value object is constructed for each possible enumerated value. See the `Property` array `enums` and the method `makeEnumProperty`. During its construction, each property maker that can have enumerated values gets a member `enums`, which, for each of its possible enumerated values, gets a reference to the appropriate enumerated property value object. See the method `PropertyMaker.addEnum`.

Example: The properties `hyphenate` and `precedence` both have the possible value `true`. Their makers have a reference to the same property value object:

```
 org.apache.fop.fo.FOPropertyMapping.s_generics
  [org.apache.fop.fo.Constants.PR_HYPHENATE].
  enums.get("true").hashCode() = 9236202
 org.apache.fop.fo.FOPropertyMapping.s_generics
  [org.apache.fop.fo.Constants.PR_PRECEDENCE].
  enums.get("true").hashCode() = 9236202
 org.apache.fop.fo.FOPropertyMapping.s_generics
  [org.apache.fop.fo.Constants.PR_HYPHENATE].enums.get("true") = "100"
 org.apache.fop.fo.FOPropertyMapping.s_generics
  [org.apache.fop.fo.Constants.PR_HYPHENATE].enums.get("true") = {
    value: 100
    org.apache.fop.fo.properties.Property.specVal: null
}
```

Example: `leader-pattern="rule"`.

```
 this = "org.apache.fop.fo.properties.EnumProperty$Maker@25c828"
 this.enums = "{rule=82, use-content=104, dots=21, space=88}"
 this.enums.get("rule").getClass() = "class org.apache.fop.fo.properties.EnumPropert
 this.enums.get("rule") = "82"
 this.enums.get("rule") = {
    value: 82
    org.apache.fop.fo.properties.Property.specVal: null
}
```

The maker's method `checkEnumValues` returns the appropriate property value object `enums.get(value)`: `newProp = "82"`.

```
  [1] org.apache.fop.fo.properties.PropertyMaker.checkEnumValues (PropertyMaker.java
  [2] org.apache.fop.fo.properties.EnumProperty$Maker.checkEnumValues (EnumProperty.
  [3] org.apache.fop.fo.properties.PropertyMaker.make (PropertyMaker.java:383)
  [4] org.apache.fop.fo.PropertyList.convertAttributeToProperty (PropertyList.java:4
  [5] org.apache.fop.fo.PropertyList.addAttributesToList (PropertyList.java:374)
```

9.2.4. Example of a property with keywords

The value of the property `border-top-width` can be set to a width, but it can also be indicated by one of the keywords `thin`, `medium` and `thick`. The width values to which these keywords correspond are by default set by the implementation. When the property maker is constructed in `FOPropertyMapping`, it gets a hash map of keyword values.

```
 this = "org.apache.fop.fo.properties.BorderWidthPropertyMaker@1cf4a2c"
 this.propId = 55
 this.keywords = "{medium=1pt, thin=0.5pt, thick=2pt}"
}
```

The method `checkValueKeywords` returns the mapped value: `value = "1pt"`. Subsequently a property value object is created as if that value had been specified.

```
  [1] org.apache.fop.fo.properties.PropertyMaker.checkValueKeywords (PropertyMaker.j
  [2] org.apache.fop.fo.properties.PropertyMaker.make (PropertyMaker.java:387)
  [3] org.apache.fop.fo.PropertyList.convertAttributeToProperty (PropertyList.java:4
  [4] org.apache.fop.fo.PropertyList.addAttributesToList (PropertyList.java:374)
  [5] org.apache.fop.fo.FObj.addProperties (FObj.java:133)
```

9.2.5. Parsing a property with an absolute value

Property values are parsed in `PropertyParser.parseProperty`:

```
  [1] org.apache.fop.fo.expr.PropertyParser.parseProperty (PropertyParser.java:111)
  [2] org.apache.fop.fo.expr.PropertyParser.parse (PropertyParser.java:88)
  [3] org.apache.fop.fo.properties.PropertyMaker.make (PropertyMaker.java:389)
  [4] org.apache.fop.fo.PropertyList.convertAttributeToProperty (PropertyList.java:4
  [5] org.apache.fop.fo.PropertyList.addAttributesToList (PropertyList.java:374)
  [6] org.apache.fop.fo.FObj.addProperties (FObj.java:133)
```

Example: `<fo:simple-page-master master-name="simpleA4" margin="4pt">`, property being parsed: `margin="4pt"`, `propId` = 134.

The PropertyParser object:

```
 this = "org.apache.fop.fo.expr.PropertyParser@8530b8"
 this = {
    propInfo: instance of org.apache.fop.fo.expr.PropertyInfo(id=729)
    org.apache.fop.fo.expr.PropertyTokenizer.currentToken: 0
    org.apache.fop.fo.expr.PropertyTokenizer.currentTokenValue: null
    org.apache.fop.fo.expr.PropertyTokenizer.currentUnitLength: 0
    org.apache.fop.fo.expr.PropertyTokenizer.currentTokenStartIndex: 0
    org.apache.fop.fo.expr.PropertyTokenizer.expr: "4pt"
    org.apache.fop.fo.expr.PropertyTokenizer.exprIndex: 0
    org.apache.fop.fo.expr.PropertyTokenizer.exprLength: 3
    org.apache.fop.fo.expr.PropertyTokenizer.recognizeOperator: false
}
```

It has a member of type `PropertyInfo`, which contains contextual information:

```
 propInfo = "org.apache.fop.fo.expr.PropertyInfo@1abcc03"
 propInfo = {
    maker: instance of org.apache.fop.fo.properties.ListProperty$Maker(id=705)
    plist: instance of org.apache.fop.fo.PropertyList(id=737)
    fo: instance of org.apache.fop.fo.pagination.LayoutMasterSet(id=738)
    stkFunction: null
}
```

`fo` is the parent FO.

The property list of the current FO node:

```
 propInfo.plist = {
    writingModeTable: null
    writingMode: 0
    inheritableProperty: null
    parentPropertyList: instance of org.apache.fop.fo.PropertyList(id=743)
    namespace: "http://www.w3.org/1999/XSL/Format"
    elementName: "fo:simple-page-master"
    fobj: instance of org.apache.fop.fo.pagination.SimplePageMaster(id=746)
}
```

The property list up to now:

```
 propInfo.plist = "{master-name=simpleA4}"
```

Property `master-name`'s maker is `StringPropertyMaker`, which does not parse its value.

PropertyParser.parseProperty

- next(), which scans the next token; at its return:

```
    this = {
       propInfo: instance of org.apache.fop.fo.expr.PropertyInfo(id=667)
       org.apache.fop.fo.expr.PropertyTokenizer.currentToken: 12
       org.apache.fop.fo.expr.PropertyTokenizer.currentTokenValue: "4pt"
       org.apache.fop.fo.expr.PropertyTokenizer.currentUnitLength: 2
       org.apache.fop.fo.expr.PropertyTokenizer.currentTokenStartIndex: 0
       org.apache.fop.fo.expr.PropertyTokenizer.expr: "4pt"
       org.apache.fop.fo.expr.PropertyTokenizer.exprIndex: 3
       org.apache.fop.fo.expr.PropertyTokenizer.exprLength: 3
       org.apache.fop.fo.expr.PropertyTokenizer.recognizeOperator: true
   }
```

  i.e. the whole expression is a single token, it is of type `PropertyTokenizer.TOK_NUMERIC` (= 12), the unit is 2 chars long.
- Loop forever.

- Analyse the expression. Start with `parseAdditiveExpr`
  - `parseMultiplicativeExpr`
    - `parseUnaryExpr`
      - `parsePrimaryExpr`; the unit may be a relative unit (em) which must be resolved against the font size
        - construct a property value object:

```
prop = "4000mpt"
prop.getClass() = "class org.apache.fop.fo.properties.FixedLength"
```

        - `next()`: scan the next token;

```
this = {
   propInfo: instance of org.apache.fop.fo.expr.PropertyInfo(id=729)
   org.apache.fop.fo.expr.PropertyTokenizer.currentToken: 0
   org.apache.fop.fo.expr.PropertyTokenizer.currentTokenValue: null
   org.apache.fop.fo.expr.PropertyTokenizer.currentUnitLength: 2
   org.apache.fop.fo.expr.PropertyTokenizer.currentTokenStartIndex: 
   org.apache.fop.fo.expr.PropertyTokenizer.expr: "4pt"
   org.apache.fop.fo.expr.PropertyTokenizer.exprIndex: 3
   org.apache.fop.fo.expr.PropertyTokenizer.exprLength: 3
   org.apache.fop.fo.expr.PropertyTokenizer.recognizeOperator: true
}
```

        the next token is of type `currentToken = PropertyTokenizer.TOK_EOF`.
- If `currentToken = PropertyTokenizer.TOK_EOF`, break the loop.

Return the property: `p = "4000mpt"`.

9.3. Retrieving a property value

9.3.1. Overview

For all FO types the FO spec specifies a large number of property types for which the user may specify a value in order to modify the resulting layout of the document. Many of these properties have a default value, many others inherit their value from the parent FO if they are not specified. In principle the layout process must retrieve the value of each possible property type of an FO node in order to determine the appropriate value of the corresponding trait.

Retrieving a property value goes through these steps:

- First determine if the property value was specified in one or other way. This is done in the method `propertyMaker.findProperty`
  - if this property has a corresponding property and if the corresponding property is forced, i.e. if this property is relative and if a value for the corresponding property was explicitly specified, compute and return it.
  - else
    - if a value for this property was explicitly specified, compute and return it
    - else if this property has a corresponding property, compute its value; if it is not null return it
    - else if a value for a relevant shorthand property was specified, compute and return it
    - else if this property is inheritable, find it at the parent; this repeats the whole process on the parent, and possibly its parents, up to the root node; if successful, return the found property value
- If no property value is found, a default property value object is made; the default value is stored in the property maker as `defaultValue`:
  - if the default property value object was calculated earlier, it was cached as `defaultProperty`; return it
  - else make it in the same way as property value objects are made when the FO tree is constructed, in `propertyMaker.make`.

Compute corresponding, as used in `findProperty`, proceeds as follows:

- use parent's property list or this property list?
- check explicit or shorthand for corresponding
- convert property

`PropertyManager`, `CommonBorderAndPadding`, `CommonBackground`, `CommonMarginBlock`, `CommonHyphenation` are convenience classes used in the calculation of the traits. `PropertyManager` has methods to return objects of these types. A `CommonBorderAndPadding` object and a `CommonHyphenation` object are cached. A `CommonBackground` object and a `CommonMarginBlock` object are calculated when requested. Similarly for many other convenience classes `Common*` for which `PropertyManager` can return an object. These classes are in package `fo.properties`. Similar classes called `*Props` are in package `traits`. Of these `PropertyManager` can only return a `BlockProps` object.

9.3.2. Detailed overview

The retrieval of a property value is started with a call to `PropertyList.get(int propId)`, which calls `PropertyList.get(propId, true, true)`.

`PropertyList.get(int propId, boolean bTryInherit, boolean bTryDefault)`:

- Find the maker for this property as `FObj.propertyListTable[propId]`, variable `propertyMaker`
- `propertyMaker.get(int subpropId, PropertyList propertyList, boolean bTryInherit, boolean bTryDefault)`
  - `propertyMaker.findProperty(PropertyList propertyList, boolean bTryInherit)`
    - if corresponding and corresponding is forced
      - evaluate condition: `CorrespondingPropertyMaker.isCorrespondingForced(PropertyList propertyList)`
        - - return false if this is not a relative property (`corresponding.relative`)
          - return true if corresponding property was explicitly specified on this node or on its parent, depending on the type of property (`corresponding.useParent`)
      - `corresponding.compute(PropertyList propertyList)`; this subroutine and the subroutines it calls refer to the corresponding property
        - `propertyList.getExplicitOrShorthand(correspondingId)`; propertyList is that of this node or of the parent, depending on the type of property (corresponding.useParent)
          - `propertyList.getExplicitBaseProp(int correspondingId)` (see below)
          - if `(null)` `propertyList.getShorthand(int correspondingId)`
            - `propertyMaker.getShorthand(this)` (see below)
        - if `(not null)` convert property
    - else
      - `propertyList.getExplicitBaseProp(int propId)` from `propertyList` as a hash map; note that this requires a conversion from `propId` to `propName` via `s_htPropNames`; example: `propertyList = "{master-name=simpleA4}"`
      - if `(null)` `propertyMaker.compute(PropertyList propertyList)`:
        - if `(corresponding)` `corresponding.compute(PropertyList propertyList)`, as above
      - if `(null)` `propertyMaker.getShorthand(PropertyList propertyList)`
        - if `(maker.shorthands)` then for each shorthand
          - `propertyList.getExplicit(int shorthand.propId)`; a shorthand must be `ListProperty`
          - if `(not null)` parse the shorthand property value, `parser.getValueForProperty(propId, listprop, propertyMaker, this, propertyList)`; here `propId` is the `propId` of this property, not of the shorthand; a shorthand may contain values for several properties, and this method retrieves the value for the current property; if `(not null)` return it

          Note: the first shorthand property maker in `maker.shorthands` that returns a good property is used
      - if `(null && bTryInherit)` `propertyMaker.findProperty(parentPropertyList, true)`: the whole process is repeated on the parent, and possibly its parents, up to the root node.

- if (null && bTryDefault) propertyMaker.make(PropertyList propertyList)
  - if there is a cached value, defaultProperty, return it
  - propertyMaker.make(PropertyList propertyList, String value, FObj fo); value is the default value stored in the property maker, fo is the parent FO
    - if default value is inherit, propertyList.getFromParent(propId)
      - if (parentPropertyList != null) parentPropertyList.get(propId); the whole process is repeated on the parent, and possibly its parents, up to the root node.
      - propertyMaker.make(PropertyList propertyList); this seems to create an endless recursion; parentPropertyList == null in the root node.
    - else check enumerated values propertyMaker.checkEnumValues(value): get the enumerated property from propertyMaker.enums
    - if (null)
      - propertyMaker.checkValueKeywords(String value); check if the value is a keyword in propertyMaker.keywords; if so, substitute the value with the value for which the keyword stands
      - parse the value p = PropertyParser.parse(pvalue, new PropertyInfo(propertyMaker, propertyList, fo))
      - convert the property propertyMaker.convertProperty(p, propertyList, fo)
      - if (null) throw org.apache.fop.fo.expr.PropertyException, which immediately is catched and rethrown as FOPException (why?)
  - catch FOPException; this means that this method may return a null property
- catch FOPException; this means that this method may return a null property

The call stack at the findProperty and makecalls is:

```
[1] org.apache.fop.fo.properties.PropertyMaker.findProperty (PropertyMaker.java:24
[2] org.apache.fop.fo.properties.PropertyMaker.get (PropertyMaker.java:282)
[3] org.apache.fop.fo.PropertyList.get (PropertyList.java:252)
[4] org.apache.fop.fo.PropertyList.get (PropertyList.java:238)
[5] org.apache.fop.fo.FObj.getProperty (FObj.java:163)
[6] org.apache.fop.layoutmgr.PageLayoutManager.createPageAreas (PageLayoutManager.
```

```
[1] org.apache.fop.fo.properties.PropertyMaker.make (PropertyMaker.java:387)
[2] org.apache.fop.fo.properties.PropertyMaker.make (PropertyMaker.java:369)
[3] org.apache.fop.fo.properties.PropertyMaker.get (PropertyMaker.java:285)
[4] org.apache.fop.fo.PropertyList.get (PropertyList.java:252)
[5] org.apache.fop.fo.PropertyList.get (PropertyList.java:238)
[6] org.apache.fop.fo.FObj.getProperty (FObj.java:163)
[7] org.apache.fop.layoutmgr.PageLayoutManager.createPageAreas (PageLayoutManager.
```

For properties whose maker is compound property maker:

```
[1] org.apache.fop.fo.properties.PropertyMaker.findProperty (PropertyMaker.java:24
[2] org.apache.fop.fo.properties.PropertyMaker.get (PropertyMaker.java:282)
[3] org.apache.fop.fo.properties.CompoundPropertyMaker.get (CompoundPropertyMaker.
[4] org.apache.fop.fo.PropertyList.get (PropertyList.java:252)
[5] org.apache.fop.fo.PropertyList.get (PropertyList.java:238)
[6] org.apache.fop.fo.flow.Leader.getLength (Leader.java:135)
[7] org.apache.fop.layoutmgr.AddLMVisitor.getLeaderAllocIPD (AddLMVisitor.java:305
[8] org.apache.fop.layoutmgr.AddLMVisitor$2.getAllocationIPD (AddLMVisitor.java:29
[9] org.apache.fop.layoutmgr.LeafNodeLayoutManager.getNextBreakPoss (LeafNodeLayou
```

9.3.3. Examples: Retrieving border and padding values

In this section we follow in detail how the border and padding values for the body region are retrieved. The relevant part of the input FO file is:

```
<fo:simple-page-master master-name="simpleA4" margin="4pt">
```

```
<fo:region-body margin="4pt+20%" border="3pt solid black"
  border-before-width="2pt" border-right-width="4pt"
  border-start-width="inherit"/>
</fo:simple-page-master>
```

This section was written after I added the cache to the look-up of property values.

9.3.3.1. Retrieving the `margin-top` value on `region-body`

This what we are retrieving:

```
propertyList.getFOName() = "fo:region-body"
org.apache.fop.fo.FOPropertyMapping.getPropertyName(propId) = "margin-top"
```

The margin values are retrieved by the method `PropertyManager.getMarginProps`. This is the call stack that leads up to it and on to the retrieval of the value of `margin-top`:

```
[1] org.apache.fop.fo.properties.PropertyMaker.findProperty (PropertyMaker.java:240)
[2] org.apache.fop.fo.PropertyList.findProperty (PropertyList.java:289)
[3] org.apache.fop.fo.properties.PropertyMaker.get (PropertyMaker.java:291)
[4] org.apache.fop.fo.PropertyList.get (PropertyList.java:261)
[5] org.apache.fop.fo.PropertyList.get (PropertyList.java:247)
[6] org.apache.fop.fo.PropertyManager.getMarginProps (PropertyManager.java:264)
[7] org.apache.fop.fo.pagination.RegionBody.getViewportRectangle (RegionBody.java:58)
[8] org.apache.fop.layoutmgr.PageLayoutManager.makeRegionViewport (PageLayoutManager.jav
[9] org.apache.fop.layoutmgr.PageLayoutManager.createPageAreas (PageLayoutManager.java:7
[10] org.apache.fop.layoutmgr.PageLayoutManager.createPage (PageLayoutManager.java:721)
[11] org.apache.fop.layoutmgr.PageLayoutManager.makeNewPage (PageLayoutManager.java:441)
[12] org.apache.fop.layoutmgr.PageLayoutManager.doLayout (PageLayoutManager.java:191)
```

The retrieval proceeds as follows:

- `PropertyList.findProperty`: Is the value in the cache? No.
- `PropertyMaker.findProperty`: `corresponding != null`? No.
- `PropertyMaker.findProperty`: Is this property explicitly specified? No.
- `PropertyMaker.findProperty`: Can the corresponding property be computed? No, there is no corresponding property.
- `PropertyMaker.findProperty`: Is a shorthand property for this property specified? Yes, `margin` is a shorthand for it. The property value is retrieved as:

  ```
  listprop = "[(4000mpt +20.0%)]"
  ```

  It is a list property as are all shorthand properties. The value for `margin-top` is extracted from it as:

  ```
  p = "(4000mpt +20.0%)"
  ```

- `PropertyList.findProperty`: Add the value to the cache.

9.3.3.2. Retrieving the `border-before-style` value on `region-body`

This what we are retrieving:

```
propertyList.getFOName() = "fo:region-body"
org.apache.fop.fo.FOPropertyMapping.getPropertyName(propId) = "border-before-style"
```

The border values are retrieved by the method `PropertyManager.getBorderAndPadding`. This is the

call stack that leads up to it and on to the retrieval of the value of `border-before-style`:

```
[1] org.apache.fop.fo.properties.PropertyMaker.findProperty (PropertyMaker.java:24
[2] org.apache.fop.fo.PropertyList.findProperty (PropertyList.java:289)
[3] org.apache.fop.fo.properties.PropertyMaker.get (PropertyMaker.java:291)
[4] org.apache.fop.fo.PropertyList.get (PropertyList.java:261)
[5] org.apache.fop.fo.PropertyList.get (PropertyList.java:247)
[6] org.apache.fop.fo.PropertyManager.initBorderInfo (PropertyManager.java:155)
[7] org.apache.fop.fo.PropertyManager.getBorderAndPadding (PropertyManager.java:14
[8] org.apache.fop.fo.PropertyManager.getMarginProps (PropertyManager.java:289)
[9] org.apache.fop.fo.pagination.RegionBody.getViewportRectangle (RegionBody.java:
[10] org.apache.fop.layoutmgr.PageLayoutManager.makeRegionViewport (PageLayoutMana
[11] org.apache.fop.layoutmgr.PageLayoutManager.createPageAreas (PageLayoutManager
[12] org.apache.fop.layoutmgr.PageLayoutManager.createPage (PageLayoutManager.java
[13] org.apache.fop.layoutmgr.PageLayoutManager.makeNewPage (PageLayoutManager.jav
[14] org.apache.fop.layoutmgr.PageLayoutManager.doLayout (PageLayoutManager.java:1
```

The retrieval proceeds as follows:

- `PropertyList.findProperty`: Is the value in the cache? No.
- `PropertyMaker.findProperty`: `corresponding != null`, but `corresponding.isCorrespondingForced()` returns `false`. The corresponding property is `border-top-style`, which is not specified:

  ```
  org.apache.fop.fo.FOPropertyMapping.getPropertyName(correspondingId) = "border-to
  ```

  This is the corresponding property maker:

  ```
  this = {
     baseMaker: instance of org.apache.fop.fo.properties.EnumProperty$Maker(id=816)
     lr_tb: 54
     rl_tb: 54
     tb_rl: 43
     useParent: false
     relative: true
  }
  ```

- `PropertyMaker.findProperty`: Is this property explicitly specified? No.
- `PropertyMaker.findProperty`: Can the corresponding property be computed? Yes, it can be derived from the property `border`, which is a shorthand for the corresponding property `border-top-style`. The returned value is `87`, which stands for `SOLID`.

  Note that the shorthand was not used in the computation of `corresponding.isCorrespondingForced()`, but it is in the computation of the specified value of the corresponding property.
- `PropertyList.findProperty`: Add the value to the cache.

9.3.3.3. Retrieving the `border-before-width` value on `region-body`

This what we are retrieving:

```
propertyList.getFOName() = "fo:region-body"
org.apache.fop.fo.FOPropertyMapping.getPropertyName(propId) = "border-before-width"
```

The border values are retrieved by the method `PropertyManager.getBorderAndPadding`. This is the call stack that leads up to it and on to the retrieval of the value of `border-before-width`:

```
main[1] where
```

```
[1] org.apache.fop.fo.properties.PropertyMaker.findProperty (PropertyMaker.java:240)
[2] org.apache.fop.fo.PropertyList.findProperty (PropertyList.java:289)
[3] org.apache.fop.fo.properties.PropertyMaker.get (PropertyMaker.java:291)
[4] org.apache.fop.fo.properties.CompoundPropertyMaker.get (CompoundPropertyMaker.java:14
[5] org.apache.fop.fo.PropertyList.get (PropertyList.java:261)
[6] org.apache.fop.fo.PropertyList.get (PropertyList.java:247)
[7] org.apache.fop.fo.PropertyManager.initBorderInfo (PropertyManager.java:157)
[8] org.apache.fop.fo.PropertyManager.getBorderAndPadding (PropertyManager.java:143)
[9] org.apache.fop.fo.PropertyManager.getMarginProps (PropertyManager.java:289)
[10] org.apache.fop.fo.pagination.RegionBody.getViewportRectangle (RegionBody.java:58)
[11] org.apache.fop.layoutmgr.PageLayoutManager.makeRegionViewport (PageLayoutManager.jav
[12] org.apache.fop.layoutmgr.PageLayoutManager.createPageAreas (PageLayoutManager.java:1
[13] org.apache.fop.layoutmgr.PageLayoutManager.createPage (PageLayoutManager.java:721)
[14] org.apache.fop.layoutmgr.PageLayoutManager.makeNewPage (PageLayoutManager.java:441)
[15] org.apache.fop.layoutmgr.PageLayoutManager.doLayout (PageLayoutManager.java:191)
```

The difference with the call stack for `border-before-style` is that `border-before-width` is a compound property, with a `minimum`, an `optimum` and a `maximum` value.

The retrieval proceeds as follows:

- `PropertyList.findProperty`: Is the value in the cache? No.
- `PropertyMaker.findProperty`: `corresponding != null`, but `corresponding.isCorrespondingForced()` returns `false`. The corresponding property is `border-top-width`, which is not specified:

  ```
  org.apache.fop.fo.FOPropertyMapping.getPropertyName(correspondingId) = "border-top-widt
  ```

- `PropertyMaker.findProperty`: Is this property explicitly specified? Yes. The property value is retrieved as:

  ```
  p = "CondLength[2000mpt]"
  ```

  The specified value was `2pt`. When this attribute value was added to the property list, it was converted to a `CondLength` type.
- `PropertyList.findProperty`: Add the value to the cache.

9.3.3.4. Retrieving the `border-end-width` value on `region-body`

This what we are retrieving:

```
propertyList.getFOName() = "fo:region-body"
org.apache.fop.fo.FOPropertyMapping.getPropertyName(propId) = "border-end-width"
```

The border values are retrieved by the method `PropertyManager.getBorderAndPadding`. The call stack that leads up to it and on to the retrieval of the value of `border-end-width` is identical to the call stack for `border-before-width`.

The retrieval proceeds as follows:

- `PropertyList.findProperty`: Is the value in the cache? No.
- `PropertyMaker.findProperty`: `corresponding != null`, and `corresponding.isCorrespondingForced()` returns `true`. The corresponding property is `border-right-width`, which is explicitly specified:

  ```
  org.apache.fop.fo.FOPropertyMapping.getPropertyName(correspondingId) = "border-right-wi
  ```

- `PropertyMaker.findProperty`: Compute the corresponding property value. It is retrieved as:

```
p = "CondLength[discard, 4000mpt]"
```

The specified value was `4pt`. When this attribute value was added to the property list, it was converted to a `CondLength` type.

- `PropertyList.findProperty`: Add the value to the cache.

9.4. Percent-based and mixed property values

9.4.1. Overview

Properties may have relative values, expressed as a percentage. The value is relative to a trait of the layout. Therefore relative values can only be evaluated when the layout is created. The FO tree must store them as an expression.

The FO spec specifies for each property that can have a relative value, which trait is the basis for the evaluation of the relative value. FOP maintains that information in the property maker, in its member `percentBase`. This is set when the maker is created, see `FOPropertyMapping`.

When the maker creates a relative property value object, it stores its own member `percentBase` in the property as member `iBaseType`. In this way the property value contains the information that is needed at layout time to find the trait that is the basis for the calculation of the actual value.

The possible values of the member `percentBase` are listed in class `LengthBase`. Both the interface `PercentBase` and the class `LengthBase` define static constants which are used as types. The difference is as follows (from an email by Finn Bock, edited by me): The idea is that the `PercentBase.XXX` types name the stored values and the `LengthBase.XXX` types name the algorithms for looking up a base value. Most of the time these map one-to-one to each other, but for some I imaged that they would be different. For example, `margin-top` should really use an algorithm like `BLOCK_IPD_OR_PAGEHEIGHT`, which would look for either `PercentBase.BLOCK_IPD` or `PercentBase.PAGE_HEIGHT`, depending on the FO element.

A `LengthBase` object contains a reference to a parent FO node and a reference to a property list. In this manner the `LengthBase` value contains the information that is needed at layout time to locate the FO node or property list which contains the trait that is the basis for the calculation of the actual value, irrespective of the FO node at which the property is resolved.

The method `LengthBase.getBaselength` uses the base type, member `iBaseType`, to determine relative to which layout trait the value should be resolved. If the trait is a property value, it is retrieved from the property list, in the usual manner. If it is a layout dimension, it is taken from the parent FO.

The interface `Numeric`, which is implemented by all classes that can participate in numeric operations, uses the notion of `dimension`, which denotes the type of numeric: for integers `dimension = 0`, for lengths `dimension = 1`.

The constructor of `RelativeNumericProperty` calculates the dimension of the new property value object as a combination of the dimensions of the operands, depending on the operation:

- multiplication adds dimensions: 0+0=0, 0+1=1+0=1,
- division subtracts dimensions: 0-0=0, 1-0=1, 1-1=0,
- other (addition, subtraction) does not change the dimensions

A `RelativeProperty` contains the operation between the operands in its member `operation`. It is an integer. Names for the possible integer values are listed as static members of class `RelativeProperty`.

Relative and mixed property values are retrieved in the usual manner. After retrieval they must be resolved against the relevant layout traits. This happens in the method `getValue` of `Property` and its subclasses.

A `RelativeNumericProperty` is resolved as follows:

- `RelativeNumericProperty.getValue`
  - `RelativeNumericProperty.getNumericValue`
    - `RelativeNumericProperty.getResolved`. This evaluates the expression tree by recursion. The nodes in the tree are `RelativeNumericProperty` objects. The leaf nodes are `FixedLength` and `PercentLength` objects. On each node `getNumericValue` is called.
      - The relative numeric property values call `getResolved` again, which descends the tree and calls `getNumericValue` on its child nodes.
      - The fixed lengths return their millipoints.
      - The percent lengths return `factor * lbase.getBaseLength()`
        - `factor` is a member of the percent length object.
        - `LengthBase.getBaselength` gets the base length from the parent FO
          - `FObj.getLayoutDimension`. The value is retrieved from the `Map layoutDimension`. If that does not contain the desired layout dimension, then its parent is consulted, all the way up to `fo:root`.

9.4.2. Parsing a mixed property value

Example: `<fo:region-body margin="4pt+20%"/>`, property being parsed: `margin="4pt+20%"`, `propId` = 134.

`PropertyParser.parseProperty`

- `next()`, which scans the next token; at its return:

```
 this = {
    propInfo: instance of org.apache.fop.fo.expr.PropertyInfo(id=736)
    org.apache.fop.fo.expr.PropertyTokenizer.currentToken: 12
    org.apache.fop.fo.expr.PropertyTokenizer.currentTokenValue: "4pt"
    org.apache.fop.fo.expr.PropertyTokenizer.currentUnitLength: 2
    org.apache.fop.fo.expr.PropertyTokenizer.currentTokenStartIndex: 0
    org.apache.fop.fo.expr.PropertyTokenizer.expr: "4pt+20%"
    org.apache.fop.fo.expr.PropertyTokenizer.exprIndex: 3
    org.apache.fop.fo.expr.PropertyTokenizer.exprLength: 7
    org.apache.fop.fo.expr.PropertyTokenizer.recognizeOperator: true
}
```

  i.e. the whole expression is a single token, it is of type `PropertyTokenizer.TOK_NUMERIC` (= 12), the unit is 2 chars long.
- Loop forever.
  - Analyse the expression. Start with `parseAdditiveExpr`
    - `parseMultiplicativeExpr`
      - `parseUnaryExpr`
        - `parsePrimaryExpr`; the unit may be a relative unit (em) which must be resolved against the font size
          - construct a property value object:

```
prop = "4000mpt"
prop.getClass() = "class org.apache.fop.fo.properties.FixedLength"
```

          - `next()`: scan for the next token;

```
 this = {
    propInfo: instance of org.apache.fop.fo.expr.PropertyInfo(id=736)
    org.apache.fop.fo.expr.PropertyTokenizer.currentToken: 8
    org.apache.fop.fo.expr.PropertyTokenizer.currentTokenValue: null
    org.apache.fop.fo.expr.PropertyTokenizer.currentUnitLength: 2
    org.apache.fop.fo.expr.PropertyTokenizer.currentTokenStartIndex: 3
    org.apache.fop.fo.expr.PropertyTokenizer.expr: "4pt+20%"
    org.apache.fop.fo.expr.PropertyTokenizer.exprIndex: 4
    org.apache.fop.fo.expr.PropertyTokenizer.exprLength: 7
    org.apache.fop.fo.expr.PropertyTokenizer.recognizeOperator: false
}
```

the next token is of type `currentToken = PropertyTokenizer.TOK_PLUS`.

- `next()`: scan for the next token;

```
this = {
   propInfo: instance of org.apache.fop.fo.expr.PropertyInfo(id=736)
   org.apache.fop.fo.expr.PropertyTokenizer.currentToken: 14
   org.apache.fop.fo.expr.PropertyTokenizer.currentTokenValue: "20%"
   org.apache.fop.fo.expr.PropertyTokenizer.currentUnitLength: 2
   org.apache.fop.fo.expr.PropertyTokenizer.currentTokenStartIndex: 4
   org.apache.fop.fo.expr.PropertyTokenizer.expr: "4pt+20%"
   org.apache.fop.fo.expr.PropertyTokenizer.exprIndex: 7
   org.apache.fop.fo.expr.PropertyTokenizer.exprLength: 7
   org.apache.fop.fo.expr.PropertyTokenizer.recognizeOperator: true
}
```

the next token is of type `currentToken = PropertyTokenizer.TOK_PERCENT`. The currentTokenValue `20%` is analysed:

- `parseMultiplicativeExpr`
  - `parseUnaryExpr`
    - `parsePrimaryExpr`
      - `propInfo.getPercentBase`: create a PercentBase property
        - `propInfo.getFunctionPercentBase` uses a stack of functions stkFunction, which currently `== null`
        - `if (null) maker.getPercentBase(fo, plist)`: create and return a `LengthBase` property, which implements `PercentBase`

```
pcBase = "org.apache.fop.datatypes.LengthBase@171f189"
pcBase = {
   parentFO: instance of org.apache.fop.fo.pagination.SimplePageMa
   propertyList: instance of org.apache.fop.fo.PropertyList(id=807
   iBaseType: 5
}
```

          the value of `iBaseType` is derived from `maker.percentBase == 5 == LengthBase.BLOCK_WIDTH`.
      - `pcBase.getDimension`; dimension: type of integer, int # 0, length # 1; used by `PercentBase` and `NumericProperty`; `LengthBase` has a dimension of 1
      - create a `PercentLength(pcval, pcBase)`:

```
prop = "20.0%"
prop = {
   factor: 0.2
   lbase: instance of org.apache.fop.datatypes.LengthBase(id=751)
   org.apache.fop.fo.properties.Property.specVal: null
}
```

        `factor` comes from `pcval`, `lbase = pcBase`
      - `next()`: scan for the next token; the next token is of type `currentToken = PropertyTokenizer.TOK_EOF`.
    - return the `LengthBase` property value object
- `evalAddition(NumericProperty op1, NumericProperty op2)`. `op1` and `op2` are now `Numeric`, which is an interface implemented by `LengthProperty`, of which `FixedLength` and `PercentLength` are subclasses:

```
op1 = instance of org.apache.fop.fo.properties.FixedLength(id=744)
op2 = instance of org.apache.fop.fo.properties.PercentLength(id=757)
```

  - `NumericOp.addition`
    - Construct a new `RelativeNumericProperty` by adding the two properties: `RelativeNumericProperty(RelativeNumericProperty.ADDITION, op1, op2)`
- If `currentToken = PropertyTokenizer.TOK_EOF`, break the loop.

return the `RelativeNumericProperty`:

```
 prop = "(4000mpt +20.0%)"
 prop = {
    operation: 1
    op1: instance of org.apache.fop.fo.properties.FixedLength(id=744)
    op2: instance of org.apache.fop.fo.properties.PercentLength(id=757)
    dimension: 1
    org.apache.fop.fo.properties.Property.specVal: null
}
```

The value 1 for the operation corresponds to `RelativeProperty.ADDITION`. The value 1 for the dimension indicates that this is a length. PropertyList:

```
 this = "{margin=[(4000mpt +20.0%)]}"
```

9.4.3. Resolving a mixed property value

Example: Resolving the value of the property `margin-left` of the FO node `fo:region-body`. This value was specified as `<fo:region-body margin="4pt+20%"/>`. `margin` is a shorthand property for `margin-left`.

The property list of `fo:region-body` reads:

```
 this = "{margin=[(4000mpt +20.0%)]}"
```

The retrieved property value is a `RelativeNumericProperty`:

```
 prop = "(4000mpt +20.0%)"
 prop = {
    operation: 1
    op1: instance of org.apache.fop.fo.properties.FixedLength(id=817)
    op2: instance of org.apache.fop.fo.properties.PercentLength(id=818)
    dimension: 1
    org.apache.fop.fo.properties.Property.specVal: null
}
```

The value 1 for the operation corresponds to `RelativeProperty.ADDITION`. The value 1 for the dimension indicates that this is a length.

```
 op2 = "20.0%"
 op2 = {
    factor: 0.2
    lbase: instance of org.apache.fop.datatypes.LengthBase(id=751)
    org.apache.fop.fo.properties.Property.specVal: null
}
 lbase = "org.apache.fop.datatypes.LengthBase@171f189"
 lbase = {
    parentFO: instance of org.apache.fop.fo.pagination.SimplePageMaster(id=786)
    propertyList: instance of org.apache.fop.fo.PropertyList(id=807)
    iBaseType: 5
}
```

The `RelativeNumericProperty` is resolved in the method call

```
 props.marginLeft =
   this.propertyList.get(PR_MARGIN_LEFT).getLength().getValue();
```

in `PropertyManager.getMarginProps()`. `getLength()` is a sort of cast; it returns the property

value if it is a length. The `getValue()` method invoked is `RelativeNumericProperty.getValue()`. This calls `RelativeNumericProperty.getNumericValue()`, which calls `RelativeNumeric-Property.getResolved()`. This invokes the operation on its two operands, `NumericOp.addition2`, which invokes `getNumericValue()` on each operand. `PercentLength.getNumericValue()` calls `LengthBase.getBaseLength` on its member `lbase`.

Due to its value `iBaseType == 5 == LengthBase.BLOCK_WIDTH` this invokes `par-entFO.getLayoutDimension(PercentBase.BLOCK_IPD).intValue()` on its parent FO. The simple page master FO node does not have any layout dimensions, its member `layoutDimension` is `null`. Therefore it consults its parent FO. This goes all the way up to the root FO node.

The root FO node does have the required layout dimensions, which are the page dimensions. These have been set on it by the `PageLayoutManager` when the page was created in its method `createPageAreas`:

```
((FObj) fobj.getParent()).setLayoutDimension(PercentBase.BLOCK_IPD,pageWidth)
((FObj) fobj.getParent()).setLayoutDimension(PercentBase.BLOCK_BPD,pageHeight)
PercentBase.BLOCK_IPD = 2, PercentBase.BLOCK_BPD = 3
```

As a result:

```
layoutDimension = "{2=576000, 3=792000}"
key = "2"
getName() = "fo:root"
```

```
[1] org.apache.fop.fo.FObj.getLayoutDimension (FObj.java:241)
[2] org.apache.fop.datatypes.LengthBase.getBaseLength (LengthBase.java:120)
[3] org.apache.fop.fo.properties.PercentLength.getNumericValue (PercentLength.java
[4] org.apache.fop.fo.expr.NumericOp.addition2 (NumericOp.java:52)
[5] org.apache.fop.fo.expr.RelativeNumericProperty.getResolved (RelativeNumericPro
[6] org.apache.fop.fo.expr.RelativeNumericProperty.getNumericValue (RelativeNumeri
[7] org.apache.fop.fo.expr.RelativeNumericProperty.getValue (RelativeNumericProper
[8] org.apache.fop.fo.PropertyManager.getMarginProps (PropertyManager.java:267)
```

`PercentLength.getNumericValue()` returns the page width, which is multiplied by the requested factor of 0.2, and added to the value of the fixed length, 4000. The resulting value is returned and used for the variable `marginLeft`:

```
value = 119200.0
props.marginLeft = 119200
```

Chapter 10

Fonts

10.1. Font setup

Terminology:

- Index, font index: The index of a character in a font, i.e. the place of the glyph for a character in a font.
- Code point: The same as font index.
- Character value: The two-byte (`char`) value by which a character is represented in memory and in Unicode. Note that this only works straightforwardly for the basal plane (BMP) of Unicode, i.e. for `characters <= 0xFFFF`.
- Unicode code point: The same as the character value.

During compilation for each of the 14 base fonts a class is generated from the XML font metric files. Each font class contains the metric information and an encoding table (a `CodePointMapping` object). The metric information is static, the encoding table is an object member.

During compilation also a class `CodePointMapping` is generated, which contains the known encodings as static values. For each known encoding it contains a table as a static final array of int. The array holds an alternation of font index and character value; in fact it is a mapping from `table[2i]` to `table[2i+1]`, where `table[2i]` is the font index and `table[2i+1]` is the character.

When an encoding is needed in the process, a `CodePointMapping` object is created from the encoding table. It contains a table (`char array`) called `latin1Map` of the font indices for the characters of the Latin1 range (`0-0xFF`) in character value order. It also contains two tables (`char array`s), called characters and codepoints, for the higher character values. The table characters contains the character values in order, and the table codepoints contains the corresponding font indexes for this encoding in the same order. The characters can be retrieved from these tables as follows:

```
char <= 0xFF: index = latin1Map[character]
char > 0xFF:
         find i such that characters[i] == char;
         then index = codepoints[i]
```

In the code the characters are retrieved from the `CodePointMapping` object with its method `mapChar(char c)`.

In FOP's preparation stage the fonts are set up in the method `Driver.getContentHandler`. It calls the `renderer`'s method `setupFontInfo(currentDocument)`. The `Document` object `currentDocument` (which is the `foTreeControl` object) is able to store the font setup info and has methods to access the fonts registered with it.

The `PrintRenderer` (PostScript and PDF) then calls `FontSetup.setup(fontInfo, fontList)`, where `fontInfo` is `the` Document object and `fontList` is the list of user configured fonts registered with the `renderer` in its member `fontList`.

```
  [1] org.apache.fop.fonts.FontSetup.setup (FontSetup.java:98)
  [2] org.apache.fop.render.PrintRenderer.setupFontInfo (PrintRenderer.java:77)
  [3] org.apache.fop.apps.Driver.getContentHandler (Driver.java:551)
  [4] org.apache.fop.apps.Driver.render (Driver.java:602)
  [5] org.apache.fop.apps.Driver.render (Driver.java:589)
  [6] org.apache.fop.apps.Fop.main (Fop.java:102)
```

`FontSetup.setup` takes three actions:

1. An object is created for each of the base 14 fonts and registered with the `fontInfo` object in its member fonts.

2. A series of triplets (family, style, weight) is set up. To each triplet a font is assigned; this font will be used when a font with the characteristics of that triplet is requested. The triplets registered with `fontInfo` in its member `triplets`. The member `triplets` is a map which uses a string of the form `family,style,weight` as a key. There is also a class `FontTriplet`, which is not used.

3. The user configured fonts are added.

In the following listing `treeBuilder` is the tree builder object set up in the preparation stage, and `foTreeControl` is the document object. The list of user configured fonts of the `renderer` is empty, and the list of used fonts is still empty.

```
 treeBuilder.foTreeControl.fonts = "{
  F1=org.apache.fop.fonts.base14.Helvetica@e3c624,
  F2=org.apache.fop.fonts.base14.HelveticaOblique@e020c9,
  F3=org.apache.fop.fonts.base14.HelveticaBold@13e58d4,
  F4=org.apache.fop.fonts.base14.HelveticaBoldOblique@15a6029,
  F5=org.apache.fop.fonts.base14.TimesRoman@17494c8,
  F6=org.apache.fop.fonts.base14.TimesItalic@1e57e8f,
  F7=org.apache.fop.fonts.base14.TimesBold@888e6c,
  F8=org.apache.fop.fonts.base14.TimesBoldItalic@d3db51,
  F9=org.apache.fop.fonts.base14.Courier@5f6303,
  F10=org.apache.fop.fonts.base14.CourierOblique@117f31e,
  F11=org.apache.fop.fonts.base14.CourierBold@1d7fbfb,
  F12=org.apache.fop.fonts.base14.CourierBoldOblique@5d9084
  F13=org.apache.fop.fonts.base14.Symbol@39e5b5,
  F14=org.apache.fop.fonts.base14.ZapfDingbats@1b5998f,
}"


 treeBuilder.foTreeControl.triplets = "{
  Computer-Modern-Typewriter,normal,400=F9,
  Courier,italic,400=F10,
  Courier,italic,700=F12,
  Courier,normal,400=F9,
  Courier,normal,700=F11,
  Courier,oblique,400=F10,
  Courier,oblique,700=F12,
  Helvetica,italic,400=F2,
  Helvetica,italic,700=F4,
  Helvetica,normal,400=F1,
  Helvetica,normal,700=F3,
  Helvetica,oblique,400=F2,
  Helvetica,oblique,700=F4,
  Symbol,normal,400=F13,
  Times Roman,italic,400=F6,
  Times Roman,italic,700=F8,
  Times Roman,normal,400=F5,
  Times Roman,normal,700=F7,
  Times Roman,oblique,400=F6,
  Times Roman,oblique,700=F8,
  Times,italic,400=F6,
  Times,italic,700=F8,
  Times,normal,400=F5,
  Times,normal,700=F7,
  Times,oblique,400=F6,
  Times,oblique,700=F8,
  Times-Roman,italic,400=F6,
```

```
  Times-Roman,italic,700=F8,
  Times-Roman,normal,400=F5,
  Times-Roman,normal,700=F7,
  Times-Roman,oblique,400=F6,
  Times-Roman,oblique,700=F8,
  ZapfDingbats,normal,400=F14,
  any,italic,400=F6,
  any,italic,700=F8,
  any,normal,400=F5,
  any,normal,700=F7,
  any,oblique,400=F6,
  any,oblique,700=F8,
  monospace,italic,400=F10,
  monospace,italic,700=F12,
  monospace,normal,400=F9,
  monospace,normal,700=F11,
  monospace,oblique,400=F10,
  monospace,oblique,700=F12,
  sans-serif,italic,400=F2,
  sans-serif,italic,700=F4,
  sans-serif,normal,400=F1,
  sans-serif,normal,700=F3,
  sans-serif,oblique,400=F2,
  sans-serif,oblique,700=F4,
  serif,italic,400=F6,
  serif,italic,700=F8,
  serif,normal,400=F5,
  serif,normal,700=F7,
  serif,oblique,400=F6
  serif,oblique,700=F8,
}"


 treeBuilder.foTreeControl.atModel.renderer.fontList = null
 treeBuilder.foTreeControl.usedFonts = "{}"
```

User configured fonts should be listed in the member `fontList` of the `renderer`. The objects in the list are `EmbedFontInfo` objects. They are created from the path to the metrics file, boolean kerning, the list of triplets for which this font may be used, the path to the font file. The triplets are `FontTriplet` objects. The list may be created from an Avalon configuration object with `FontSetup.buildFontListFromConfiguration(Configuration cfg)`.

`>FontSetup.addConfiguredFonts` creates a `LazyFont` font object from each `EmbedFontInfo` object. `LazyFont` fonts are not loaded until they are actually used. This makes it possible to register a large number of fonts at low cost.

Font weights are integers between 100 and 900. `Font.NORMAL` and `Font.BOLD` are set to 400 and 700, respectively. See `FontUtil.parseCSS2FontWeight`.

10.2. Classes and interfaces used in the font package

```
IF FontMetrics
SubIF FontDescriptor

IF MutableFont

Abstract Class TypeFace: FontMetrics
Classes Courier etc.

Abstract Class TypeFace: FontMetrics
Abstract Class CustomFont: FontDescriptor, MutableFont
Abstract Class CIDFont, Class SingleByteFont
```

```
Class MultiByteFont (sub CIDFont)

Abstract Class TypeFace: FontMetrics
Abstract Class CustomFont: FontDescriptor, MutableFont
Class SingleByteFont

Abstract Class TypeFace: FontMetrics
Class LazyFont: FontDescriptor

Abstract Class TypeFace: FontMetrics
Class FontMetricsMapper, for AWT fonts
```

`SingleByteFont`, `MultiByteFont`: A font is not really single or multibyte. Rather the name `SingleByteFont` indicates that the font does not contain more than 256 glyphs; the implementation is optimized for this. In `MultiByteFont` (actually CIDFont Type2) the implementation is optimized for fonts with an unknown number of glyphs.

Chapter 11

Configuration

Configuration is based on the `Configurable` and `Configuration` interfaces of `avalon`:

```
org.apache.avalon.framework.configuration.Configurable
org.apache.avalon.framework.configuration.Configuration
```

A type that implements `Configurable` can be configured by calling its method `configure(Configuration configuration)`, where the argument is the `Configuration` object that holds the user configuration settings. It can also be configured by calling the static method `ContainerUtil.configure(object, cfg)` of the class

```
ContainerUtil = org.apache.avalon.framework.container.ContainerUtil
```

This method checks if `object` implements `Configurable`. If not, no configuration is attempted.

The following classes implement `Configurable`:

- `render.AbstractRenderer` and its subclasses (all renderers). Only `render.pdf.PDFRenderer` and `render.ps.PSRenderer` have meaningful implementations of the `configure` method. The command line module configures each renderer from the user configuration file with the subconfiguration `renderers/renderer[@mime=$mimetype]`.
- `svg.PDFTranscoder`
- `svg.PDFDocumentGraphics2D`. This class is configured by `svg.PDFTranscoder` via `ContainerUtil.configure(graphics, this.cfg)`.

In addition `render.ps.AbstractPSTranscoder` has a member `Configuration cfg`. It configures graphics via `ContainerUtil.configure(graphics, this.cfg)`. The graphics are of type `render.ps.AbstractPSDocumentGraphics2D`, which does not implement `Configurable`, so that no configuration takes place.

`render.pdf.PDFRenderer` and `svg.PDFDocumentGraphics2D` both call `fonts.FontSetup.buildFontListFromConfiguration(cfg)` and `pdf.PDFFilterList.buildFilterMapFromConfiguration(cfg)`.

Configuration info is used by:

- `fonts.FontSetup.buildFontListFromConfiguration(cfg)`. It uses `fonts/font/font-triplet` from the renderer subconfiguration.
- `pdf.PDFFilterList.buildFilterMapFromConfiguration(cfg)`. It uses `filterList/value` from the renderer subconfiguration
- `render.ps.PSRenderer.configure(cfg)`. It uses `auto-rotate-landscape` as a Boolean.