# 11. XSL Formatting Objects

## 11.1 Introduction

### 11.1.1 Fiche d'identité

**XSL-FO** *(XSL Formatting Objects)* est une application **XML** qui décrit la présentation *(mise en page)* d'un document texte, généralement à des fins d'impression. La sémantique de l'application **XSL-FO** est par conséquent orientée **présentation**.

**XSL 1.0** *(aka. XSL-FO)* a fait l'objet d'une première recommandation émise par le **W3C** en octobre 2001, suivie par **XSL 1.1** en décembre 2006. **XSL-F0 2.0** a fait l'objet d'un document de travail *(Working Draft)* en janvier 2012, qui n'a pas évolué depuis.

☞ Recommandation XSL 1.0
[https://www.w3.org/TR/2001/REC-xsl-20011015/]
☞ Spécifications XSL 1.1
[http://www.w3.org/TR/xsl11/]
☞ Document de travail sur XSL 2.0
[https://www.w3.org/TR/xslfo20/]
☞ L'ensemble des spécifications XSL au W3C
[http://www.w3.org/Style/XSL/]

**XSL-FO** constitue l'aboutissement du groupe de travail du **W3C** sur les feuilles de style. C'est pour cette raison que le terme **XSL** fait parfois référence à **XSL-FO**, notamment dans les spécifications.

**> Place dans l'écosystème**

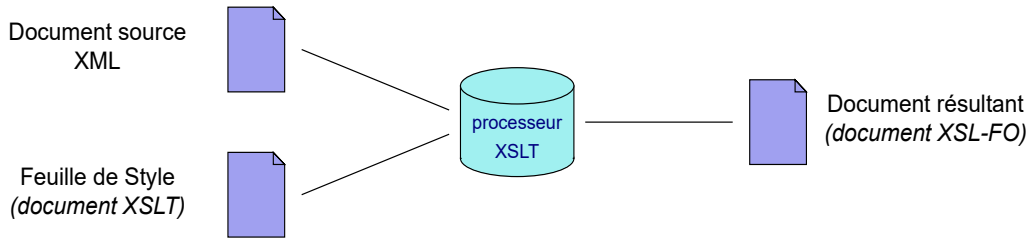
**XSL-FO** a été conçu pour émettre des documents de complexité simple à moyenne *(rapports, factures, documents techniques)* et n'est malheureusement pas en mesure de produire des documents typographiquement parfaits comme ceux issus des outils de publication professionnels comme **QuarkXPress** ou **InDesign**, ou générés via **LaTeX**.
☞ cf. Wikipedia
[https://en.wikipedia.org/wiki/XSL_Formatting_Objects]

Il existe également un document de travail pour permettre la création de documents paginés via **CSS**. Toutefois, ce document émis en 2013 n'a pas évolué depuis :
☞ CSS Paged Media Module Level 3
[https://www.w3.org/TR/css3-page/]

Attention toutefois à ne pas sous-estimer **XSL-FO** qui permet très facilement de générer des documents professionnels de qualité, à condition de maîtriser **XML** et **XSLT** et de ne pas viser la perfection typographique d'un ouvrage papier. Pour information, la version pdf de ce cours a été générée via XSL-FO.

### 11.1.2 Rappel des principes

**XSL-FO** est issu des besoins identifiés par le groupe de travail en charge de la définition de l'application de feuilles de styles pour **XML**. Le principe retenu pour la visualisation d'un document, consiste à le transformer à l'aide d'une feuille de style **XSLT** afin d'obtenir un document **XSL-FO** décrivant la mise en page :



Contrairement à **XHTML**, **XSL-FO** comprend la notion de **pages**, et permet de gérer des pages d'entêtes, de titre, des hauts de page, bas de page, la numérotation, des pages paires et impaires, *etc...*

Bien que rien dans le principe n'interdise l'idée qu'un navigateur puisse *(un jour)* visualiser directement les documents **XSL-FO**, leur exploitation est plus particulièrement pertinente sur support papier.

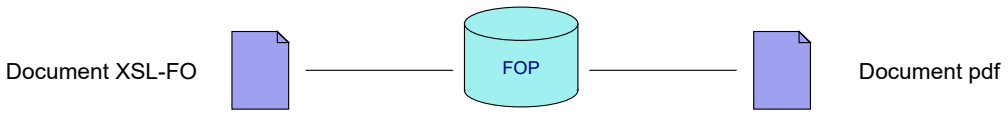
Là encore, rien n'interdirait que le format **XSL-FO** soit directement interprété par des imprimantes compatibles, comme c'est la cas pour **Postscript** par exemple. Malheureusement, il n'existe à l'heure actuelle *(fin 2016)* aucune imprimante de ce type.

Comment alors exploiter un document **XSL-FO** ?

- Il existe des applications spécialisées permettant de visualiser un document **XSL-FO** sur écran *(généralement utilisées en phase de développement)*.
- D'autres applications permettent de traduire la syntaxe **XSL-FO** en un format d'impression *(pdf, pcl, postscript, rtf, svg, mif-framemaker...)* :

☞ Antenna House Formatter V6
[https://www.antennahouse.com/product/ahf63/index.html]
☞ Apache FOP
[http://xmlgraphics.apache.org/fop/]
☞ Render-X XEP engine
[http://www.renderx.net/Content/tools/xep.html]

**FOP** est une application de ce type, gratuite et open-source, développée et maintenue par la fondation Apache, qui permet l'impression de documents **XSL-FO** en les transformant en documents au format **pdf**




FOP est l'application utilisée pour obtenir la version pdf de ce cours.

### 11.1.3 Avertissement

**XSL-FO** est une application relativement complexe.

Les spécifications comportent plus de 400 pages. Certaines fonctionnalités, bien que simples à énoncer sont très difficiles à implémenter, et il n'existe pas d'implémentation exhaustive de l'ensemble des fonctionnalités décrites par les spécifications.

Par suite, ce cours est lui aussi loin d'être exhaustif et ne doit en aucun cas être considéré comme une information de référence sur **XSL-FO**. Le parti-pris a été choisi d'aborder cette technologie sous un aspect essentiellement pragmatique, à travers les fonctionnalités offertes par **FOP** V2.0.

## 11.2 Les éléments d'un document XSL-FO

### 11.2.1 Document XSL-FO

**XSL-FO** est une application compatible avec les espaces de noms **XML**. L'**URI d'espace de noms** réservé est *"http://www.w3.org/1999/XSL/Format"*, et le préfixe le plus couramment employé est *"fo"*.

L'élément racine d'un document **XSL-FO** s'appelle *"root"*.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
 ...
</fo:root>
```

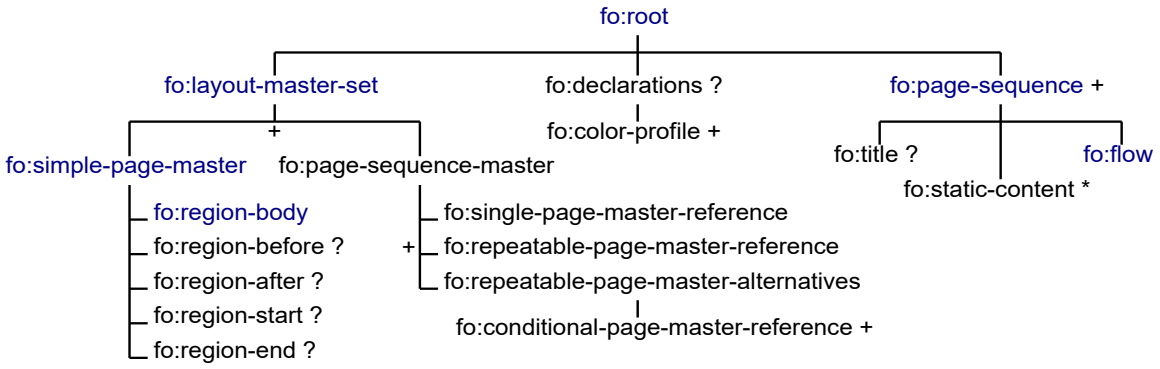
### 11.2.2 Structure d'un document XSL-FO

L'élément racine d'un document **XSL-FO** comporte toujours :

- une unique occurrence de l'élément *"fo:layout-master-set"*,
- un élément *"fo:declarations"* optionnel,
- un ou plusieurs éléments *"fo:page-sequence"* :

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
 <fo:layout-master-set>
  ...
 </fo:layout-master-set>
 <fo:page-sequence master-reference="...">
  ...
 </fo:page-sequence>
</fo:root>
```

La figure ci-dessous illustre la structure générale d'un document **XSL-FO** telle qu'imposée par les spécifications. Les éléments en couleur sont ceux qui permettent de constituer un document simple *"à moindre frais"*.




L'élément *"fo:layout-master-set"* définit un ou plusieurs modèles de pages *(pages de titre, pages courantes, paires, impaires, ...)*, appelés *"page-masters"* qui décrivent le découpage *(entête avec le titre, zone de contenu, zone de bas de page, marges...)*, ainsi que la taille et la position des différentes zones.

Les éléments *"fo:page-sequence"* mettent en place le contenu des pages, texte, et attributs de mise en forme.

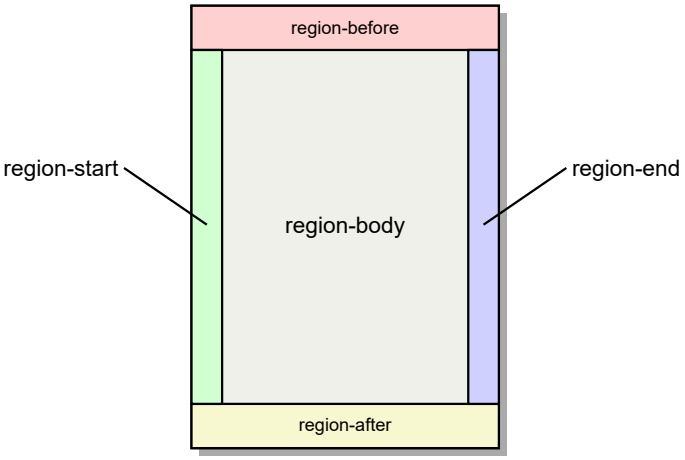
### 11.2.3 Modèle de page

L'élément *"fo:simple-page-master"* est utilisé par le processeur d'impression lors de la génération de pages de texte. Il décrit la géométrie des pages.

Une page peut être subdivisée en plusieurs zones : le corps de la page *(region-body)*, l'entête *(region-before)*, le bas de page *(region-after)*, la marge gauche *(region-start)*, la marge droite *(region-end)*. La seule zone obligatoire est *"region-body"*.

**N.B.** Les zones portent ces noms assez étranges pour cause d'internationalisation. En effet, lorsque l'écriture se fait de droite à gauche par exemple, *"region-start"* se trouve à droite, contrairement à ce qui serait le cas d'une zone qui serait appelée *"region-left"*...

La figure suivante illustre la position respective de ces cinq zones, dans le cas le plus classique *(orientation portrait, écriture de gauche à droite)* :




L'exemple ci-dessous met en place les éléments structurels minimaux nécessaires à un document **XSL-FO** :

```
<fo:layout-master-set>
 <fo:simple-page-master    master-name="page-unique"
                           page-height="29.7cm"  page-width="21cm"
                           margin-top="1.5cm"    margin-bottom="2cm"
                           margin-left="2.5cm"   margin-right="1cm">
  <fo:region-body background-color="#CCCCCC"/>
 </fo:simple-page-master>
</fo:layout-master-set>

<fo:page-sequence master-reference="page-unique">
 <fo:flow flow-name="xsl-region-body">
  <fo:block>Hello World !</fo:block>
 </fo:flow>
</fo:page-sequence>
```

Tester l'exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/start/simple-page-master-ex1.html]

Noter les attributs qui décrivent la géométrie de la page. Les noms des attributs, et les principes sous-jacents attachés aux propriétés qu'ils recouvrent sont très massivement dérivés de **CSS**.

Remarquer également comment l'élément *"fo:page-sequence"* référence le modèle de page à utiliser à l'aide de la valeur de l'attribut *"master-reference"*, ainsi que l'attribut *"flow-name"* de l'élément *"flow"* qui indique dans quelle zone se trouvera le texte concerné.

**N.B.** Les marges s'appellent bien *"margin-left"*, *etc...* et non pas *margin-start*... comme les zones, car elles ne sont pas effectées par le sens de l'écriture.

### 11.2.4 Contenu informationnel

Chacun des éléments *"fo:page-sequence"* possède un unique élément *"fo:flow"* qui contient lui-même le texte qui sera affiché après pagination par le processeur d'impression :

```
<fo:flow flow-name="xsl-region-body">
 <fo:block>Hello World !</fo:block>
 ...
</fo:flow>
```

Comme illustré par l'exemple ci-dessus, le gros du texte est traditionnellement affiché dans la zone qui correspond au corps de la page.

Lorsque d'autres zones ont été définies, elles contiennent habituellement des informations répétitives d'une page à l'autre. Ces informations sont mises en place grâce à des éléments *"fo:static-content"* :

```
<fo:static-content flow-name="xsl-region-before" font-weight="bold">
 <fo:block text-align="center" space-before="0.5em" font-size="12pt">
  Exemple de document XSL-FO
 </fo:block>
</fo:static-content>
```

```
<fo:static-content flow-name="xsl-region-after" text-align="center">
 <fo:block space-before="0.5em">
  Document XSL-FO © Daniel Muller
 </fo:block>
</fo:static-content>
```

On peut observer sur cet exemple que **XSL-FO** a repris le concept d'**héritage des propriétés** introduit par **CSS** : les attributs de présentation *(marges, style, police, couleurs, ...)* sont transmis de père en fils dans l'arbre **XML**, ce qui évite d'attacher les attributs correspondants à chacun des éléments.

Les zones ci-dessus venant prendre place **dans les marges de la zone principale**, il faut par ailleurs donner au corps de la page des marges suffisantes pour abriter l'entête et le bas de page :

```
<fo:region-before extent="1.5cm"/>
<fo:region-after extent="1.5cm"/>
<fo:region-body margin-top="1.6cm" margin-bottom="1.6cm"/>
```

Exemple complet :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/start/page-sequence-ex2.html]

**N.B.** Dans l'exemple ci-dessus on aurait tout aussi bien pu spécifier les marges à l'aide des propriétés *"space-before"* et *"space-after"* *(plutôt que margin-top et margin-bottom)*

Vérifier :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/start/page-sequence-ex3.html]

### 11.2.5 Blocs de texte

Les éléments *"fo:flow"* et *"fo:static-content"* contiennent du texte formatté en blocs. Les blocs s'empilent dans la direction principale de progression du texte *(i.e. de haut en bas, conformément aux habitudes locales)*.

Les éléments *"fo:block"* sont à rapprocher des **éléments blocs** *(block-level elements)* de **HTML**, et servent pour la mise en page des titres, des paragraphes, des légendes de tables et de figures, etc...

En général, chaque élément *"fo:block"* porte les propriétés de mise en forme qui lui sont spécifiques, et profite de celles qui sont *"mises en facteur"* par héritage :

```
<fo:flow flow-name="xsl-region-body" font-size="10pt"
         text-align="justify" start-indent="5pt" end-indent="5pt">
 <fo:block font-weight="bold" space-before="0.5em">
  L'enfance de Zéphyrin.
 </fo:block>
 <fo:block space-before="0.5em">
  C'est le 1er janvier, à minuit une seconde sexagésimale de temps moyen,
  que le jeune Brioché poussa ses premiers vagissements. A son baptême,
  il reçut les prénoms harmonieux, poétiques et distingués de Pancrace,
  Eusèbe, Zéphyrin, ce dont il parut se soucier comme un cloporte
  d'un ophicléide.
 </fo:block>
 <fo:block space-before="0.5em" font-style="italic">
  Extrait de : l'idée fixe du Savant Cosinus.
 </fo:block>
</fo:flow>
```

Exemple complet :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/start/block-ex1.html]

**N.B.** On pourrait être tenté de reporter la propriété *"space-before"* commune à l'ensemble des éléments sur leur parent *"fo:flow"* de manière à les en faire hériter. Malheureusement *"space-before"* n'est pas une propriété héritée, contrairement à *"start-indent"* par exemple.

**Remarque :** *"fo:block"* n'est pas le seul élément bloc. Il en existe d'autres comme *"fo:table"* ou *"fo:list-block"* qui seront étudiés en leur temps.

### 11.2.6 Eléments texte

On peut souvent être amené à vouloir modifier la présentation du texte *"en ligne"*, sans pour autant changer de bloc. Cette fonctionnalité est assurée par l'élément *"fo:inline"* :

```
<fo:block space-before="0.5em" font-style="italic">
 Extrait de :
 <fo:inline font-weight="bold">
  l'idée fixe du Savant Cosinus
 </fo:inline>.
</fo:block>
```

Exemple complet :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/start/inline-ex1.html]

## 11.3 Présentation des blocs

### 11.3.1 Alignement du texte

L'alignement des blocs de texte dans la direction perpendiculaire à la direction de progression des blocs *(i.e. de gauche à droite dans notre cas)* peut s'effectuer à l'aide des propriétés *"start-indent" (gauche)* et *"end-indent" (droite)* :

```
<fo:block start-indent="2em" end-indent="2em">
 ...
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/block/text-align-ex1.html]

**N.B.** La valeur par défaut est *"0"* : le texte est aligné sur les bords gauche et droit du bloc.

L'espacement entre blocs se règle préférentiellement par les propriétés *"space-before" (haut)* et *"space-after" (bas)*. La valeur par défaut est *"0" (pas d'espace supplémentaire)*.

```
<fo:block font-weight="bold" space-before="1em" space-after="1em">
 ...
</fo:block>
<fo:block space-before="0.5em">
 ...
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/block/text-align-ex2.html]

A noter que **XSL-FO** possède des règles complexes pour le calcul des espaces entre blocs, qui sont en première approximation déterminés de manière à satisfaire la plus importante des demandes entre la marge bas du bloc supérieur et la marge haut du bloc inférieur.

### 11.3.2 Bordures

Les bordures des blocs sont invisibles par défaut. Chacun des bords *(haut, bas, gauche, droite)* de chacun des blocs peut être rendu visible sous des formes diverses. Les propriétés relatives aux bordures sont la couleur, le style et la largeur.

### > Propriétés des bordures

La couleur s'exprime à l'aide de la propriété *"border-color"* avec des valeurs conformes aux standards introduits par **CSS**, notation hexadécimale *"#RRGGBB"* ou fonction *"rgb(r,g,b)"*. La valeur par défaut est la couleur du fond.

La propriété *"border-width"* donne la largeur, qui peut prendre l'une des valeurs prédéfinies *"thin"*, *"medium"* ou *"thick"*, ainsi qu'une valeur numérique exprimée avec l'une des unités reconnues *(cm, mm, pt, pc, em, ex ...)*. La valeur par défaut est *"medium"*.

Les styles exprimés par la propriété *"border-style"* sont prédéfinis. Les valeurs autorisées sont : *"none" (pas de bordure)*, *"hidden" (bordure cachée, effet identique au précédent)*, *"dotted" (pointillés)*, *"dashed" (pointillés longs)*, *"solid" (cadre)*, *"double" (cadre double)*, *"groove" (cadre en relief rentrant)*, *"ridge" (cadre en relief sortant)*, *"inset" (ligne en relief rentrant)* et *"outset" (ligne en relief sortant)*. La valeur par défaut est *"none"*.

```
<fo:block border-color="#008800" border-width="0.3mm" border-style="solid">
...
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/block/border-ex1.html]

### > Côtés individuels

Le réglage individuel des diverses bordures se fait à l'aide des propriétés dont le nom générique peut se noter *"border-x-y"*, où *"x"* identifie le côté concerné et *"y"* est l'une des propriétés vues ci-dessus : *"color"*, *"width"*, ou *"style"*.

L'identification du côté peut prendre l'une des valeurs *"top" (haut)*, *"bottom" (bas)*, *"left" (gauche)*, *"right" (droite)*, *"before" (précédent, dans le sens de progression des blocs)*, *"after" (suivant, dans le sens de progression des blocs)*, *"start" (précédent, dans le sens de progression des caractères)*, ou *"end" (suivant, dans le sens de progression des caractères)*.

```
<fo:block border-left-style="solid" border-left-color="#CC0000">
 <!-- bloc accompagné d'un trait dans la marge gauche -->
</fo:block>
<fo:block border-bottom-style="solid" border-bottom-color="#000088">
 <!-- souligné -->
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/block/border-ex2.html]

### > Raccourcis typographiques

Conformément à ce qui avait déjà été introduit par **CSS**, **XSL-FO** permet de simplifier les spécifications relatives aux bordures grâces à des raccourcis syntaxiques.

Outre les attributs *"border-width"*, *"border-color"* et *"border-style"* introduits en début de section qui permettent de fixer d'un coup une propriété pour les quatre côtés, il existe des raccourcis permettant de fixer les trois propriétés *"width"*, *"style"* et *"color" (dans cet ordre)* pour un côté donné.

Ces attributs prennent alors la forme *"border-x" (où x représente comme plus haut le nom d'un côté)*, avec la propriété super-raccourcie *"border"* qui permet de fixer d'un coup les trois propriétés des quatre côtés :

```
<fo:block border-bottom="thin solid #000088">
 ...
</fo:block>
<fo:block border="2pt double black">
 ...
</fo:block>
```

### 11.3.3 Couleur et motif du fond

La décoration des blocs ne s'arrête pas à leurs bordures. Il est également possible de colorer leur fond, ou de l'illustrer à l'aide d'une image.

#### > Couleur de fond

La couleur du fond est indiquée à l'aide de la propriété *"background-color"*. La valeur indiquée peut être une couleur *(au sens de CSS)* ou le mot-clé *"transparent"*. La valeur par défaut est *"transparent"*.

L'exemple suivant, en colorant le fond des paragraphes, permet de visualiser l'espace vertical entre blocs :

```
<fo:block space-before="0.5em" background-color="#CCCCCC">
 ...
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/block/background-ex1.html]

#### > Motif du fond

De manière tout à fait similaire à ce qui se fait dans le domaine du web, **XSL-FO** permet de paver le fond des blocs à l'aide d'une image. La proprété qui permet d'attacher l'image au bloc est *"background-image"*. Les valeurs possibles sont une **URL** ou le mot-clé *"none"*. La valeur par défaut est *"none"*.

```
<fo:block space-before="0.5em" background-image="papier-peint.gif">
 ...
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/block/background-ex2.html]

Le comportement par défaut consiste à paver le fond du bloc concerné en répétant l'image horizontalement et verticalement. Il est possible de contrôler ce comportement à l'aide de la propriété *"background-repeat"* qui peut prendre les valeurs *"repeat"*, *"repeat-x"*, *"repeat-y"*, et *"no-repeat"*. La valeur par défaut est *"repeat"*.

Lorsque l'image n'est pas répétée, il est logique de pouvoir la positionner. Ceci est assuré à l'aide des propriétés *"background-position-horizontal"* et *"background-position-vertical"*. Les valeurs possibles s'expriment en pourcent de la dimension correspondante du bloc, à l'aide d'une dimension munie d'une unité, ou à l'aide d'un mot clé parmi *"left"*, *"center"* et *"right" (position horizontale)* ou *"top"*, *"center"* et *"bottom" (position verticale)*.

Une application possible des images de fond consiste à mettre un logo en *"signe d'eau"* sur le fond de la page. L'élément auquel appliquer l'image de fond est alors *"fo:region-body"* :

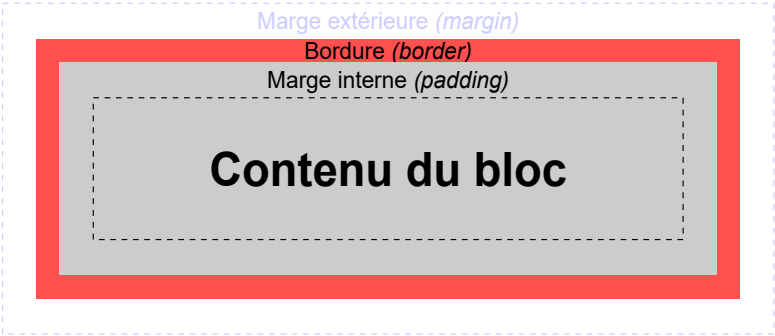
```
<fo:region-body background-image="logo.gif"
                background-repeat="no-repeat"
                background-position-horizontal="center"
                background-position-vertical="center"/>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/block/background-ex3.html]

### 11.3.4 Marges internes

Le comportement normal des blocs consiste à aligner le texte au plus près des bordures *(i.e. sans espace)*. Lorsque ceux-ci comportent un fond coloré ou lorsque les bordures sont matérialisées, il peut être intéressant de disposer d'un marge dite *"interne"* entre le texte et la bordure.

De fait, **XSL-FO** s'appuie sur un modèle identique à celui introduit par **CSS** *(Cascading Style Sheets)* pour la gestion des marges et des bordures :




Comme pour **CSS**, les marges internes sont globalement fixées par la propriété *"padding"*. La valeur par défaut est *"0"*.

```
<fo:block background-image="image.gif" padding="0.5em">
 ...
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/block/padding-ex1.html]

Attention à cette propriété qui, conformément aux recommandations, résulte en un comportement initialement surprenant *(comme illustré par l'exemple précédent)*.

Pour remédier à l'effet observé, on peut compenser la marge interne en réglant les propriétés *"start-indent"* et *"end-indent"* :

```
<fo:block space-before="0.5em" background-image="image.gif"
          padding="0.5em" start-indent="0.5em" end-indent="0.5em">
 ...
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/block/padding-ex2.html]

**N.B.** Les marges internes peuvent également se régler individuellement à l'aide des propriétés *"padding-x"*, où *"x"* prend l'une des valeurs *"top"*, *"bottom"*, *"left"*, *"right"*, *"before"*, *"after"*, *"start"* ou *"end"*, avec les significations habituelles.

### 11.3.5 Inclusion de blocs

Les blocs sont susceptibles de contenir du texte, mais aussi d'autres blocs.

Cette propriété permet de mettre en facteur un certain nombre de propriétés qui peuvent être héritées par les blocs internes, mais aussi d'obtenir des effets de présentation comme par exemple celui qui consiste à inclure plusieurs blocs consécutifs dans une même boîte :

```
<fo:block border="0.3mm #000088 solid" space-before="1em"
          padding="0.5em" start-indent="0.5em" end-indent="0.5em">
 <fo:block>
  ...
 </fo:block>
 <fo:block>
  ...
 </fo:block>
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/block/include-ex1.html]

## 11.4 Présentation du texte

### 11.4.1 Polices de caractères

Les propriétés qui permettent de déterminer la police et la forme des caractères sont calquées sur les propriétés **CSS** équivalentes.

**> font-family**

La propriété *"font-family"* sélectionne la police de caractère à partir de son nom. Comme pour **CSS**, il est possible de spécifier une liste de polices par ordre de priorité décroissante. La valeur par défaut de cette propriété dépend du processeur considéré et de sa version *(cf. système d'exploitation)*.

Outre les polices classiques, il est possible de spécifier une **police générique** *(en général en fin de liste)*. Les polices génériques sont *"serif"*, *"sans-serif"*, *"cursive"*, *"fantasy"*, et *"monospace"*.

```
<fo:page-sequence master-reference="simple"
                  font-family="Georgia, Times New Roman, serif">
 ...
</fo:page-sequence>
```

Noter ci-dessus comment la police a été attachée à l'élément *"page-sequence"*, qui la transmet en héritage à tous ses enfants, assurant par là-même une police unique pour l'ensemble du document.

**N.B.** Le processeur d'impression **FOP** ne connait pas les polices génériques *"cursive"* et *"fantasy"*. Toutefois, cette limitation n'est pas très grave. En effet, contrairement aux pages web, les documents **XSL-FO** sont en général exploités dans un environnement parfaitement contrôlé.

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/inline/font-ex1.html]

**> font-size**

*"font-size"* permet de fixer la taille de la police. Celle-ci peut être spécifiée sous forme absolue, relative, sous forme d'une dimension, ou d'un pourcentage.

Les valeurs absolues sont données sous forme de mots-clés. Les mots-clés autorisés sont : *"xx-small"*, *"x-small"*, *"small"*, *"medium"*, *"large"*, *"x-large"* et *"xx-large"*.

Les valeurs relatives permettent de modifier la taille de la police par rapport à celle de l'élément parent *(i.e. la taille "normale")*. Les mots-clés autorisés sont *"smaller"* et *"larger"*.

La taille de la police peut être donnée directement à l'aide d'un nombre associé à une unité comme *"10pt"*.

Une taille exprimée en pourcent est comptée par rapport à la taille *"normale"*, c'est à dire celle héritée de l'élément parent.

```
<fo:block font-size="12pt">
 ...
</fo:block>
```

La valeur par défaut de cette propriété est *"medium"*.

**N.B.** Contrairement à ce qui peut être préconisé pour le web, l'accessibilité d'un document destiné à être imprimé n'est pas forcément affectée par l'usage de dimensions absolues.

**> font-style**

Les valeurs autorisées pour cette propriété sont : *"normal"*, *"italic"*, *"oblique"* et *"backslant"*. Toutefois, le résultat obtenu dépend de la disponibilité de la police.

```
<fo:block font-style="italic">
 ...
</fo:block>
```

Outre *"normal"*, la valeur la plus couramment utilisée est *"italic"*. La valeur par défaut est *"normal"*.

### > font-weight

Cette propriété admet des valeurs absolues, relatives ou numériques. Toutefois, là encore, le résultat obtenu est lié à la disponibilité de la police. L'expérience montre que les valeurs à retenir sont *"normal" (valeur par défaut)*, et *"bold"*.

```
<fo:block font-weight="bold">
 ...
</fo:block>
```

### > Autres propriétés

Les spécifications **XSL-FO** prévoient d'autres propriétés, déjà introduites par **CSS**, qui permettent d'agir sur les caractéristiques de la police de caractères. Ces propriétés sont : *"font-stretch"*, *"font-size-adjust"* et *"font-variant"*.

**N.B.** Ces propriétés ne sont pas supportées par le processeur d'impression **FOP**.

## 11.4.2 Rendu des caractères

Le rendu des caractères est affecté par les propriétés décrites ci-dessous.

### > letter-spacing

La modification de l'espacement des caractères permet de donner plus ou moins de *"volume"* à certains textes comme des titres ou des légendes.

L'avantage de cette propriété est qu'elle ne modifie pas la forme des caractères eux-mêmes, ce qui fait que son fonctionnement ne dépend pas de l'éventuelle disponibilité d'une police. Sa mise en oeuvre est d'autre part plus facile pour les processeurs d'impression qu'une propriété comme *"font-stretch"*.

Les valeurs autorisées sont le mot-clé *"normal"* ou une dimension munie d'une unité qui indique le supplément d'espacement par rapport à la normale. La valeur par défaut est *"normal"*.

```
<fo:block font-family="Haettenschweiler" letter-spacing="0.2em">
 ...
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/inline/decoration-ex1.html]

**N.B.** Il est très fortement suggéré de spécifier l'espacement des caractères à l'aide de l'unité *"em"*, proportionnelle à la taille de la police...

### > text-decoration

Les valeurs autorisées pour la propriété *"text-decoration"* sont : *"none" (texte normal)*, *"underline" (souligné)*, *"overline" (surligné)* et *"line-through" (barré)*. La valeur par dafaut est *"none"*.

```
<fo:inline text-decoration="underline">
 ...
</fo:inline>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/inline/decoration-ex2.html]

### > color

La couleur du texte est spécifiée à l'aide de la propriété *"color"*. Les valeurs autorisées sont conformes aux standards introduits par **CSS**, notation hexadécimale *"#RRGGBB"* ou fonction *"rgb(r,g,b)"*. La valeur par défaut dépend du processeur.

```
<fo:block font-size="12pt" color="#000088" space-after="1.0em">
 ...
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/inline/decoration-ex3.html]

### 11.4.3 Alignement du texte

**> text-align**

Au sein de la boîte virtuelle qui délimite les paragraphes, le texte peut être aligné à gauche, à droite, centré ou justifié. Les valeurs correspondantes de la propriété *"text-align"* sont : *"start" (début de ligne)*, *"end" (fin de ligne)*, *"left" (gauche)*, *"right" (droite)*, *"center" (centré)* et *"justify" (justifié)*. La valeur par défaut est *"start"*.

```
<fo:block text-align="justify">
 ...
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/inline/justify-ex1.html]

**> text-indent**

La propriété *"text-indent"* donne l'indentation de la première ligne de texte d'un paragraphe. Les valeurs autorisées sont une dimension munie d'une unité, ou une valeur en pourcent de la largeur du bloc.

La valeur spécifiée peut être négative. Dans ce cas, la première ligne n'est pas en retrait mais en avant. La valeur par défaut est *"0pt"*.

```
<fo:block text-indent="2em">
 ...
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/inline/justify-ex2.html]

### 11.4.4 Alignement vertical

**> line-height**

Le réglage de l'interligne intra-paragraphe s'effectue à l'aide de la propriété *"line-height"*. Les valeurs possibles sont le mot-clé *"normal"*, une dimension munie d'une unité ou un pourcentage relatif à la taille de la police de caractères. La valeur par défaut est *"normal"*.

```
<fo:block line-height="1em">
 ...
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/inline/vertical-ex1.html]

**> vertical-align**

La propriété *"vertical-align"* permet de déplacer verticalement une séquence de texte en cours de ligne. Les valeurs les plus utiles sont : *"baseline" (normal)*, *"sub" (indice)*, *"super" (exposant)*, une dimension affectée d'une unité ou une valeur en pourcent relaive à la valeur de l'interligne. La valeur par défaut est *"baseline"*.

```
ax<fo:inline vertical-align="super" font-size="80%">2</fo:inline> + bx + c = 0
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/inline/vertical-ex2.html]

### 11.4.5 Traitement des espaces

**> white-space-collapse**

Le comportement naturel d'une application **XML** est en général de ne pas tenir compte des espaces surnuméraires, et de travailler avec des chaînes de caractères compactées. Cette propriété permet de contrôler cet aspect du comportement d'un processeur d'impression.

Les valeurs autorisées sont : **false** et **true**. La valeur par défaut est *"true"*.

```
<fo:block white-space-collapse="false">
 ...
</fo:block>
```

**> wrap-option**

Lorsque le texte disponible dans un bloc ne peut tenir sur une seule ligne, un processeur d'impression introduit automatiquement des sauts de ligne aux endroits adéquats. La propriété *"wrap-option"* permet éventuellement d'empêcher ceci.

Les valeurs autorisées sont *"wrap"* et *"no-wrap"*. La valeur par défaut est *"wrap"*.

```
<fo:block wrap-option="no-wrap">
 ...
</fo:block>
```

**N.B.** Cette propriété est à manier avec précautions, car elle peut facilement conduire à des lignes de texte qui dépassent la largeur de la page. Un contrôle fin du comportement du processeur concernant le contenu de boîtes relativement petites en taille comme des cellules de tableau, semble être son domaine de prédilection.

**> line-feed-treatment**

Dans l'optique de créer un bloc de texte préformatté *(comme l'élément PRE en HTML)*, il faut tenir compte de tous les espaces, ne pas introduire de saut de ligne supplémentaires *(ce que permettent les propriété ci-dessus)*, mais aussi respecter les sauts de ligne contenus dans le texte du bloc.

Cette dernière fonctionnalité est apportée par la propriété *"line-feed-treatment"*. Les valeurs autorisées sont : *"ignore"*, *"preserve"*, *"treat-as-space"* et *"treat-as-zero-width-space"*. La valeur par défaut est *"treat-as-space"*.

```
<fo:block linefeed-treatment="preserve">
 ...
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/inline/space-ex1.html]

### 11.4.6 Ligne de caractères

**> fo:leader**

L'élément *"fo:leader"* permet de tracer une ligne formée de caractères répétés, comme dans une table des matières, ou pour obtenir un séparateur horizontal. Cet élément est muni d'une série de propriétés qui conditionnent son comportement et son rendu visuel.

Il s'agit d'un **élément texte** *(inline element)* dont le comportement consiste a priori à remplir l'espace disponible dans la ligne courante.

**> leader-pattern**

La propriété *"leader-pattern"* permet de spécifier la nature de la ligne à tracer. Les valeurs possibles sont : *"space"*, *"rule"*, *"dots"* ou *"use-content"*. La valeur par défaut est **space**.

L'espace disponible peut être rempli de diverses façons :

- par des espaces *(valeur "space")*,
- par une ligne *(valeur "rule")*,
- par des pointillés *(valeur "dots")*,
- par une séquence de caractères arbitraire*(valeur "use-content")*.

```
<fo:block text-align-last="justify" linefeed-treatment="preserve">
 Chapitre premier <fo:leader leader-pattern="dots" /> p. 12
 Chapitre 2 <fo:leader leader-pattern="dots" /> p. 24
 Chapitre trois <fo:leader leader-pattern="dots" /> p. 3564
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/inline/leader-ex1.html]

**N.B.** Pour que *"fo:leader"* fonctionne avec **FOP** il est nécessaire, comme illustré ci-dessus, que le paragraphe contenant les leaders possède un attribut *"text-align-last"* valant *"justify"*.
☞ FOP FAQ
[https://xmlgraphics.apache.org/fop/faq.html#leader-expansion]

**> leader-alignment**

Cette propriété s'applique lorsque *"leader-pattern"* vaut *"dots"* ou *"use-content"*. Elle permet d'aligner les motifs entre deux lignes générées par *"fo:leader"*.

La valeur *"reference-area"* aligne les motifs sur le bord du bloc parent, la valeur *"page"* sur le bord de la page. La valeur par défaut est *"none"* *(pas d'alignement)*.

```
...<fo:leader leader-pattern="dots" leader-alignment="reference-area"/>...
```

**N.B.** Cette propriété n'est pas *(plus)* supportée par le processeur d'impression **FOP**.

**> leader-pattern-width**

En imposant la largeur totale d'un motif, la propriété *"leader-pattern-width"* permet en fait de régler l'espace entre deux motifs consécutifs. Elle s'applique lorsque *"leader-pattern"* vaut *"dots"* ou *"use-content"*.

Les valeurs possibles sont *"use-font-metrics"*, une longueur munie d'une unité, ou un pourcentage de la largeur de la boîte parent. La valeur par défaut est *"use-font-metrics"*.

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/inline/leader-ex3.html]

**> rule-style**

Lorsque *"leader-pattern"* vaut *"rule"*, le style de la ligne est indiqué par la propriété *"rule-style"*. Les valeurs possibles sont : *"none" (pas de ligne - intérêt limité ?)*, *"dotted" (ligne de tirets courts)*, *"dashed" (ligne de tirets longs)*, *"solid" (ligne continue)*, *"double" (ligne double)*, *"groove" (ligne dont la moitié inférieure est blanche)*, *"ridge" (ligne dont la moitié supérieure est blanche)*. La valeur par défat est *"solid"*.

```
... <fo:leader leader-pattern="rule" rule-style="double"/> ...
```

**> rule-thickness**

La largeur de la ligne tracée lorsque *"leader-pattern"* vaut *"rule"*, est réglable grâce à la propriété *"rule-thickness"*. La valeur doit être une dimension munie d'une unité. La valeur par défaut est *"1pt"*.

```
... <fo:leader leader-pattern="rule" rule-thickness="3pt"/> ...
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/inline/leader-ex5.html]

## 11.5 Listes

### 11.5.1 Modèle des listes

**> Structure**

Le modèle des listes implémenté par **XSL-FO** repose sur quatre éléments distincts qui, une fois correctement emboîtés et configurés, permettent de mettre en place l'ensemble des listes habituellement rencontrées, et en particulier les listes **HTML** *"UL"*, *"OL"*, et *"DL"*.

Si on devait décrire la structure d'une liste **XSL-FO** en utilisant la syntaxe **DTD**, on écrirait :

```
<!ELEMENT fo:list-block       (fo:list-item+) >
<!ELEMENT fo:list-item        (fo:list-item-label,fo:list-item-body) >
<!ELEMENT fo:list-item-label  (%block;)+ >
<!ELEMENT fo:list-item-body   (%block;)+ >
```

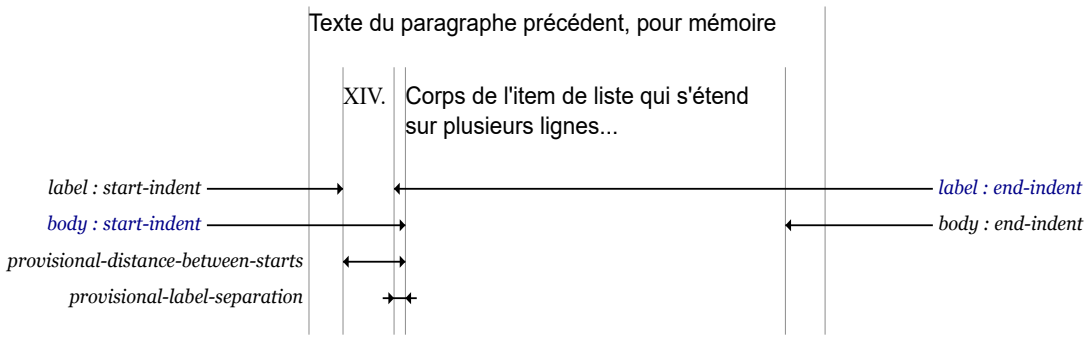
Autrement dit, une liste est constituée d'un élément *"fo:list-block"*, contenant une liste d'items matérialisés par des éléments *"fo:list-item"*, comportant chacun une entête *"fo:list-item-label"* et un corps *"fo:list-item-body"*.

Sémantiquement, l'entête d'item de liste correspond à la puce, le numéro, ou la définition de la liste, et contient techniquement un bloc, voire une liste de blocs *(cas inhabituel)*.

Le corps de l'item est classique et peut contenir un ou plusieurs blocs d'information.

### > Configuration

Le réglage fin de l'allure visuelle désiré pour la liste nécessite de comprendre les interactions entre les marges de l'entête et du corps d'un item de liste :




Par défaut, les marges *"start-indent"* et *"end-indent"* de l'entête et du corps d'item de liste sont alignés sur ceux des paragraphes normaux. Si la liste doit être mise en retrait par rapport à eux, il est loisible de modifier les propriétés *"start-indent"* de l'entête, et *"end-indent"* du corps.

La géométrie de la liste impose par contre obligatoirement de fixer de manière appropriée les propriétés *"start-indent"* du corps et *"end-indent"* de l'entête.

**XSL-FO** préconise de fixer ces valeurs grâce à deux fonctions, qui calculent les valeurs requises à partir des deux propriétés *"provisional-distance-between-starts"* et *"provisional-label-separation"*. Ces deux grandeurs sont des propriétés de la liste, et sont respectivement initialisées par défaut avec les valeurs *"24pt"* et *"6pt"*.

```
<fo:list-block provisional-label-separation="2pt"
               provisional-distance-between-starts="8pt">
 <fo:list-item>
  <fo:list-item-label end-indent="label-end()">
   <fo:block>1. </fo:block>
  </fo:list-item-label>
  <fo:list-item-body start-indent="body-start()">
   <fo:block>2 cuillères à soupe de matignon</fo:block>
  </fo:list-item-body>
 </fo:list-item>
 ...
</fo:list-block>
```

### 11.5.2 Liste à puces

Pour construire une liste à puces à l'aide du modèle de listes **XSL-FO**, le bloc d'entête d'item de liste contiendra simplement la puce, alors que le corps comprendra le texte de l'item de liste.

De fait, l'exercice le plus laborieux consistera sans doute à découvrir un caractère satisfaisant pouvant servir de puce, disponible dans la police de caractères désirée.

Voici un exemple d'item de liste à puces :

```
<fo:list-item>
 <fo:list-item-label end-indent="label-end()">
  <fo:block font-size="300%" line-height="12pt">
   <fo:inline>&#xB7;</fo:inline>
  </fo:block>
 </fo:list-item-label>
 <fo:list-item-body start-indent="body-start()">
  <fo:block>2 cuillères à soupe de matignon</fo:block>
 </fo:list-item-body>
</fo:list-item>
```

Exemple complet :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/list/ul-ex1.html]

### 11.5.3 Liste ordonnée

Les listes ordonnées se construisent de manière tout à fait similaire aux listes à puce.

La difficulté consiste à gérer le fait que la longueur des entêtes est susceptible de varier en fonction du nombre de caractères du numéro de l'item. Il suffit pour cela de prévoir une largeur suffisante, et d'aligner les labels à droite dans le bloc qui leur est réservé :

```
<fo:list-block provisional-label-separation="5pt"
               provisional-distance-between-starts="3em">
 ...
 <fo:list-item>
  <fo:list-item-label end-indent="label-end()" text-align="end">
   <fo:block>3.</fo:block>
  </fo:list-item-label>
  <fo:list-item-body start-indent="body-start()">
   <fo:block>
    Placer les bols dans un four préalablement chauffé à 200°C.
   </fo:block>
  </fo:list-item-body>
 </fo:list-item>
 ...
</fo:list-block>
```

Exemple complet :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/list/ol-ex1.html]

### 11.5.4 Liste descriptive

Les listes descriptives posent un problème particulier si on désire que l'entête puisse avoir un longueur relativement importante.

L'astuce consiste d'une part à ne pas réduire la marge droite *"end-indent"* de l'entête, et d'autre part à positionner la description elle-même une ligne plus bas :

```
<fo:list-item space-before="0.5em">
 <fo:list-item-label><!-- pas de modification de end-indent -->
  <fo:block font-weight="bold">
   Gratinée lyonnaise
 </fo:block>
 </fo:list-item-label>
 <fo:list-item-body start-indent="body-start()">
  <fo:block margin-top="1.2em">
   Bol à soupe allant au four.
  </fo:block>
 </fo:list-item-body>
</fo:list-item>
```

Exemple complet :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/list/dl-ex1.html]

### 11.5.5 Listes imbriquées

Les listes peuvent évidemment être imbriquées. Ceci revient à retrouver un *(voire plusieurs)* éléments *"fo:list-block"* comme contenu d'un élément *"fo:list-item-body"* :

```
<!-- liste principale -->
<fo:list-block space-before="0.5em"  provisional-label-separation="5pt"
               provisional-distance-between-starts="2em">
 <fo:list-item>
  <fo:list-item-label end-indent="label-end()" text-align="end">
   <fo:block font-weight="bold">A.</fo:block>
  </fo:list-item-label>
  <fo:list-item-body start-indent="body-start()">
   <!-- premier item de liste principale -->
   <fo:block font-weight="bold">
    Définitions :
   </fo:block>

   <!-- liste descriptive imbriquée -->
   <fo:list-block space-before="0.5em" provisional-label-separation="5pt"
                  provisional-distance-between-starts="1em">
    ...
   </fo:list-block>

  </fo:list-item-body>
 <fo:list-item>
 <!-- autres items de liste principale -->
 ...
</fo:list-block>
```

Exemple complet :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/list/embed-ex1.html]

## 11.6 Tables

### 11.6.1 Modèle des tables

Le modèle des tables adopté par **XSL-FO** s'appuie fortement sur celui de **HTML**, plus précisément décrit par **CSS2**. Dans ce modèle une table est essentiellement constituée de lignes, elles-mêmes constituées de cellules.

L'élément principal s'appelle **fo:table**, et contient un *(ou plusieurs)* éléments *"fo:table-body"*. Le corps de table est consitué de lignes *"fo:table-row"*, comportant elles-même des cellules *"fo:table-cell"*.

```
<fo:table text-align="center" space-before="1em">
 <fo:table-body>
  <fo:table-row>
   <fo:table-cell><fo:block>Nabuchodonosor</fo:block></fo:table-cell>
   <fo:table-cell><fo:block>16 litres</fo:block></fo:table-cell>
   <fo:table-cell><fo:block>20</fo:block></fo:table-cell>
  </fo:table-row>
 </fo:table-body>
</fo:table>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/table/principe-ex1.html]

**> Description explicite de la largeur des colonnes**

Les tables disposent d'une propriété nommée *"table-layout"* dont la valeur peut être *"auto"* ou *"fixed"*.

La valeur par défaut est *"auto"*. Elle signifie que le processeur d'impression doit calculer automatiquement la largeur des colonnes de la table en fonction du contenu des cellules *(comme le font les navigateurs HTML par exemple)*.

La valeur *"fixed"* signifie que la largeur des colonnes est explicitement indiquée dans le document. Cette information est mise en place à l'aide d'éléments *"fo:table-column"* :

```
<fo:table table-layout="fixed" text-align="center" space-before="1em">
 <fo:table-column column-width="3cm"/>
 <fo:table-column column-width="3cm"/>
 <fo:table-column column-width="3cm"/>

 <fo:table-body> . . . </fo:table-body>
</fo:table>
```

Lorsque plusieurs colonnes consécutives ont la même largeur, il est possible de ne pas les répéter, en précisant le nombre de colonnes identiques à l'aide de la propriété *"number-columns-repeated"* :

```
<fo:table table-layout="fixed" text-align="center" space-before="1em">
 <fo:table-column column-width="3cm" number-columns-repeated="3"/>

 <fo:table-body> . . . </fo:table-body>
</fo:table>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/table/principe-ex2.html]

Pour obtenir des colonnes proportionnelles *(comme avec l'élément COL et la notation 1*, 2*, 3* en HTML)*, il est possible de calculer leur largeur à l'aide de la fonction *"proportional-column-width()"* :

```
<fo:table table-layout="fixed" width="10cm">
 <fo:table-column column-width="proportional-column-width(2)"/>
 <fo:table-column column-width="proportional-column-width(1)"/>
 <fo:table-column column-width="proportional-column-width(3)"/>

 <fo:table-body> . . . </fo:table-body>
</fo:table>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/table/principe-ex3.html]

Noter toutefois que lorsque *"table-layout"* vaut *"fixed"*, ceci implique comme ci-dessus de spécifier la largeur de la table à l'aide de la propriété *"width"*.

**N.B.** Le processeur d'impression **FOP** n'implémente pas la valeur *"auto"* de la propriété *"table-layout"*. Il impose systématiquement une description explicite des colonnes.

### 11.6.2 Visualisation des tables

Come pour tous les éléments **XSL-FO** *(Formatting Objects)*, les bordures de tables ainsi que les montants *(cf. bordures de cellules)* ne sont pas visibles par défaut.

Pour les rendre visibles, il faut jouer sur les propriétés des bordures des cellules ainsi que de la table elle-même.

```
<fo:table table-layout="fixed" border="0.5pt solid #000088">
 <fo:table-column column-width="3.3cm" number-columns-repeated="3"/>
 <fo:table-body start-indent="0pt">
  <fo:table-row>
   <fo:table-cell border-bottom="0.5pt solid #000088" padding="2pt 5pt">
    <fo:block>Nom</fo:block>
   </fo:table-cell>
   ...
  </fo:table-row>
  ...
 </fo:table-body>
</fo:table>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/table/border-ex1.html]

### 11.6.3 Position horizontale d'une table

**N.B.** Les limitations citées dans cette section , sont spécifiques au processeur d'impression **FOP**.

**> Positionnement horizontal**

**FOP** implémente le positionnement des tables de manière très imparfaite. Outre le fait qu'il est incapable de calculer lui-même les largeurs de colonnes *(cf. table-layout="auto")*, le positionnement horizontal d'une table est assez difficile à obtenir.

En effet, une table vient normalement se positionner à gauche le long de la marge de la page. Pour la décaler, on peut penser à utiliser les propriétés *"start-indent"* ou *"margin-left"* :

```
<fo:table table-layout="fixed" text-align="center" space-before="1em"
          width="9cm" margin-left="5cm" border="1px solid">
 . . .
</fo:table>
```

Résultat obtenu :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/table/position-ex1.html]

Dans ce cas, **FOP** émet des erreurs et on obtient un résultat inattendu, visiblement dû au fait que le décalage imposé par la marge gauche est *"hérité"* par les cellules. Or si les spécifications indiquent que la valeur de *"start-indent"* est transmise par héritage, ce n'est pas le cas de *"margin-left"*.
☞ XSL start-indent
[https://www.w3.org/TR/xsl/#start-indent]
☞ XSL margin-left
[https://www.w3.org/TR/xsl/#margin-left]

La solution consiste à s'assurer que la marge gauche des cellules est nulle :

```
<fo:table-cell text-align="left" margin-left="0">
 <fo:block>Salmanazar</fo:block>
</fo:table-cell>
```

Résultat :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/table/position-ex2.html]

### > Centrage horizontal

Pour centrer une table horizontalement dans la page, la méthode **FOP** consiste à inclure la table en question dans une autre table, possédant trois colonnes, dont les colonnes latérales sont proportionnelles :

```
<fo:table table-layout="fixed" width="100%">
 <fo:table-column column-width="proportional-column-width(1)"/>
 <fo:table-column column-width="9cm"/>
 <fo:table-column column-width="proportional-column-width(1)"/>
 <fo:table-body>
  <fo:table-row>
   <fo:table-cell><fo:block/></fo:table-cell><-- noter la cellule vide -->
   <fo:table-cell>

    <-- table affichée -->
    <fo:table table-layout="fixed" border="1px solid">
    </fo:table>

   </fo:table-cell>
   <fo:table-cell><fo:block/></fo:table-cell><-- noter la cellule vide -->
  </fo:table-row>
 </fo:table-body>
</fo:table>
```

Résultat :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/table/position-ex3.html]

Dans l'exemple ci-dessus on fixe la largeur de la table, et on adapte les marges latérales en fonction de l'espace restant. Il est également possible de raisonner en fixant la largeur des marges et en laissant l'espace restant à la table :

```
<fo:table table-layout="fixed" width="100%">
 <fo:table-column column-width="2cm"/>
 <fo:table-column column-width="from-parent(width)-4cm"/>
 <fo:table-column column-width="2cm"/>
 <fo:table-body>
  <fo:table-row>
   <fo:table-cell><fo:block/></fo:table-cell><-- noter la cellule vide -->
   <fo:table-cell>

    <-- table affichée -->
    <fo:table table-layout="fixed" border="1px solid">
    </fo:table>

   </fo:table-cell>
   <fo:table-cell><fo:block/></fo:table-cell><-- noter la cellule vide -->
  </fo:table-row>
 </fo:table-body>
</fo:table>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/table/position-ex4.html]

Remarquer ci-dessus l'usage de la fonction **XSL-FO** *"from-parent()"* qui renvoie la valeur d'une propriété de l'élément parent.

Noter également que la valeur de l'attribut *"column-width"* de la colonne centrale est *"calculée"* *(cf. présence d'une fonction et d'un opérateur)*. Une approche similaire a été mise en oeuvre pour la définition de la géométrie des items de listes. Ceci est une caractéristique de **XSL-FO** qui offre cette possibilité de manière générique pour tous ses attributs.

### 11.6.4 Tables complexes

La mise en place de cellules couvrant plusieurs lignes ou plusieurs colonnes se fait à l'aide des propriétés *"number-columns-spanned"* et *"number-rows-spanned"* de l'élément *"fo:table-cell"*.

```
...
<fo:table-row>
 <fo:table-cell><fo:block>1</fo:block></fo:table-cell>
 <fo:table-cell><fo:block>2</fo:block></fo:table-cell>
 <fo:table-cell><fo:block>3</fo:block></fo:table-cell>
 <fo:table-cell number-rows-spanned="2"><fo:block>=</fo:block></fo:table-cell>
</fo:table-row>
<fo:table-row>
 <fo:table-cell number-columns-spanned="2"><fo:block>0</fo:block></fo:table-cell>
 <fo:table-cell><fo:block>.</fo:block></fo:table-cell>
</fo:table-row>
...
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/table/colspan-ex1.html]

Noter au passage dans l'exemple ci-dessus le rendu des montants de table *"larges"* obtenus grâce à la valeur *"separate"* de la propriété *"border-collapse"*, et de la largeur de montant fixée par *"border-separation"* :

```
<fo:table table-layout="fixed" width="5cm" border="0.5pt solid #000088"
          border-separation="3pt" border-collapse="separate" >
 ...
</fo:table>
```

### 11.6.5 Entête et bas de table

La mise en page des tables inclut la possibilité de spécifier des informations présentes en haut et en bas de table, et qui seront automatiquement rappelées en haut et en bas de chacune des pages, dans le cas où la table serait éclatée sur plusieurs pages.

Ces informations sont à inclure dans des éléments *"fo:table-header"* et *"fo:table-footer"* qui s'intègrent au même niveau que le corps de la table *"fo:table-body"* et admettent le même contenu :

```
<fo:table-header>
 <fo:table-row>
  <fo:table-cell number-columns-spanned="2" color="#880000">
   <fo:block>Introduction à XML</fo:block>
  </fo:table-cell>
 </fo:table-row>
</fo:table-header>
```

```
<fo:table-footer>
 <fo:table-row>
  <fo:table-cell number-columns-spanned="2" font-size="8pt">
   <fo:block>
    <xsl:text>Cours "</xsl:text>
    <xsl:value-of select="../@titre"></xsl:value-of>
    <xsl:text>" © 2004 Daniel Muller</xsl:text>
   </fo:block>
  </fo:table-cell>
 </fo:table-row>
</fo:table-footer>
```

Exemple avec entêtes et bas de tables :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/table/header-ex1.html]

**N.B.** En principe, lorsqu'une même table contient plusieurs occurences de *"fo:table-header"* ou de *"fo:table-footer"*, le comportement résultant est laissé à l'appréciation du processeur d'impression. Dans le cas de **FOP**, une table ne peut comporter qu'une seule instance de chacun de ces éléments.

**N.B. XSL-FO** prévoit également un mécanisme de table avec légende basé sur les éléments *"fo:table-and-caption"* et *"fo:table-caption"* qui ne sont pas supportés par **FOP**. Comme illustré par l'exemple ci-dessus, les entêtes et/ou bas de table peuvent constituer une alternative intéressante à cette lacune.

## 11.7 Graphiques

### 11.7.1 Graphiques externes

Comme **HTML**, **XSL-FO** permet d'intégrer des images dans une page à l'aide d'un élément nommé *"fo:external-graphic"*.

L'**URL** de la ressource est classiquement spécifiée via une propriété nommée *"src"*, charge au processeur de la récupérer lors de la construction de la page.

```
<fo:block>
 <fo:external-graphic src="url(VGE.jpg)"/>
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/image/external-ex1.html]

Les formats d'image reconnus dépendent bien sûr du processeur d'impression.

Le processeur **FOP** par exemple, supporte de manière native les formats : *"BMP" (Microsoft Windows Bitmap)*, *"EPS" (Encapsulated Postscript)*, *"GIF" (Graphics Interchange Format)*, *"JPEG" (Joint Photographic Experts Group)*, et *"TIFF" (Tag Image Format File)*. Il peut être également configuré pour les formats *"PNG" (Portable Network Graphic)* et *"SVG" (Scalable Vector Graphics)*.

**> Taille de l'image**

La taille de l'image obtenue dans le cas d'une image bitmap dépend du nombre de pixels, et de la résolution par défaut adoptée par le processeur d'impression qui est par exemple de *72 dpi* pour **FOP**.

Cette taille peut être modifiée grâce aux propriétés *"content-width"* et *"content-height"* de l'élément *"fo:external-graphic"*. Les valeurs de ces propriétés doivent être munies d'une unité :

```
<fo:external-graphic content-width="6cm" src="url(VGE.jpg)"/>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/image/external-ex2.html]

Lorsqu'une seule dimension est spécifiée, la deuxième dimension est modifiée de sorte à ne pas étirer l'image. Lorsque deux dimensions sont indiquées, l'image peut être déformée.

**> Emplacement de l'image**

Comme en **HTML**, l'image est un élément directement inséré dans le flot du texte *(inline element)*. Cette propriété est en général mise à profit pour la génération de puces, ou autres pseudo-caractères du même type.

```
Les images ne servent pas uniquement de puces. Elles peuvent s'insérer
<fo:external-graphic src="url(triangle-right.gif)"/>n'importe
où<fo:external-graphic src="url(triangle-left.gif)"/>
dans le texte !
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/image/external-ex3.html]

Les images *"inline"* de l'exemple ci-dessus sont alignées verticalement au sein de la ligne grâce à la propriété *"alignment-baseline"* :

```
<fo:external-graphic
    content-height="1em"
    alignment-baseline="mathematical"
    src="url(triangle-right.gif)"/>
```

Les blocs flottants *(élément "fo:float")* sont partiellement implémentés par **FOP**. La solution la plus sûre pour positionner une image par rapport à du texte reste toutefois l'emploi de tables.

☞ **FOP** floats
[https://xmlgraphics.apache.org/fop/fo.html#floats]

Revoir l'exemple 2 ci-dessus :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/image/external-ex2.html]

### 11.7.2 Graphiques internes

*"fo:instream-foreign-object"* est un autre élément qui permet d'insérer des objets *non-XSL* dans un document **XSL-FO**.

La différence essentielle avec *"fo:external-graphic"* est que l'élément n'est pas référencé mais **inclus** dans le code source du document. La conclusion immédiate est qu'il doit a priori s'agir d'un format **XML** reconnu par le processeur d'impression visé.

Le seul format éligible à l'heure actuelle *(quel que soit le processeur d'impression)* est **SVG** :

```
<fo:block text-align="center">
 <fo:instream-foreign-object width="15cm">
  <svg xmlns="http://www.w3.org/2000/svg"
       width="15cm" height="3cm" viewBox="0 0 1000 200">
   ...
  </svg>
 </fo:instream-foreign-object>
</fo:block>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/image/instream-ex1.html]

Bien entendu, l'usage d'un format graphique **XML** apporte d'insignes avantages, comme le fait de pouvoir générer le graphique lui-même à l'aide de **XSLT** ou d'autres outils compatibles **XML** *(et sous cet aspect, l'exemple ci-dessus n'est pas exempt de magie...)*.

## 11.8 Gestion des pages

### 11.8.1 Numérotation des pages

L'élément *"fo:page-number"* permet d'insérer le numéro de la page courante.

Il est couramment utilisé pour la numérotation automatique des pages, et de ce fait en général inséré dans la section *"fo:static-content"* qui correspond au bas de page :

```
<fo:static-content flow-name="xsl-region-after"
                   text-align="center" font-style="italic" font-size="10pt">
 <fo:block font-weight="bold">
  - Page <fo:page-number/> -
 </fo:block>
</fo:static-content>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/page/number-ex1.html]

### 11.8.2 Référence à une page

L'élément *"fo:page-number-citation"* permet d'insérer le numéro de la page contenant l'élément dont l'identifiant correspond à la valeur de l'attribut *"ref-id"*. Cet élément est destiné à la génération de tables des matières, de références croisées et d'index.

Une astuce permet également de l'utiliser pour obtenir le nombre total de pages d'un document. Il suffit pour cela d'insérer en fin de document un bloc *(même vide)* portant un identifiant connu :

**sonde en fin de document**

```
<fo:flow flow-name="xsl-region-body">
 ...
 <fo:block id="last-page"/>
</fo:flow>
```

puis de faire référence à cet élément à chaque fois que nécessaire :

**référence**

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

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/page/citation-ex1.html]

### 11.8.3 Gestion des sauts de page

Le fonctionnement normal d'un processeur d'impression consiste à insérer automatiquement des sauts de page aux endroits adéquats. Des nécessités de présentation peuvent toutefois inciter l'auteur à vouloir contrôler la manière dont ils s'effectuent, voire à imposer certains sauts de page.

#### > Contrôle des sauts de page

On désire parfois interdire les sauts de page entre certains éléments, comme par exemple entre un titre et le premier paragraphe d'une section. Cette possibilité est offerte grâce aux propriétés *"keep-with-next.within-page"*, *"keep-with-previous.within-page"* et *"keep-together.within-page"*.

```
<fo:block font-weight="bold" space-before="1em" keep-with-next.within-page="1">
 Calme précurseur d'événements orageux.
</fo:block>
```

La valeur par défaut de ces propriétés est *"auto"*, ce qui autorise un saut de page entre éléments. La valeur *"always"* interdit le saut de page, tandis qu'une valeur numérique entière décourage d'autant plus le saut de page que la valeur est élevée.

Bien utilisé, ce mécanisme est très puissant pour prioriser les endroits où les sauts de page sont autorisés, et permet de générer des documents de manière automatique sans quasiment aucune retouche manuelle.

Exemple sans contrôle des sauts de page :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/page/break-ex5.html]

Le même avec contrôle :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/page/break-ex6.html]

#### > Insertion d'un saut de page

Les propriétés *"break-after"* et *"break-before"* servent à forcer un saut de page. Les valeurs possibles sont *"auto"*, *"column"*, *"page"*, *"even-page"*, et *"odd-page"*. La valeur par défaut est *"auto"*.

Ces diverses valeurs permettent de provoquer à l'endroit adéquat, un simple saut de page *(valeur "page")*, ou d'imposer que l'élément concerné *(i.e. l'élément courant ou le suivant)* se trouve sur une page paire *(even-page)* ou impaire *(odd-page)*, ou dans la colonne suivante en cas d'impression multi-colonne *(vue plus loin)*.

Voici comment introduire un saut de page à l'aide d'un bloc vide :

```
<fo:block break-after="page"/>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/page/break-ex1.html]

**> Veuves et orphelines**

La propriété *"orphans"* spécifie le nombre minimum de lignes d'un paragraphe coupé en bas de page. De manière similaire, la propriété *"widows"* impose un nombre minimum de lignes pour un paragraphe coupé en haut de page.

```
<fo:flow flow-name="xsl-region-body" orphans="3" widows="3">
```

Exemple avec les valeurs par défaut (orphans=2, widows=2) :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/page/break-ex2.html]

Exemple avec orphans=3 :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/page/break-ex3.html]

Exemple avec widows=3 :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/page/break-ex4.html]

## 11.9 Mise en page avancée

### 11.9.1 Impression multi-colonnes

La génération d'un document dont le texte est disposé sur plusieurs colonnes est élémentaire.

Il suffit pour cela de préciser le nombre de colonnes désiré grâce à la propriété *"column-count"* de l'élément *"fo:region-body"*. La valeur par défaut de cette propriété est bien sûr *1*.

Une fois le texte s'affichant sur plusieurs colonnes, l'espace entre celles-ci est ajusté à l'aide de la propriété *"column-gap"*, dont la valeur par défaut est *"12pt"*.

```
<fo:simple-page-master master-name="page-unique" ... >
 <fo:region-before extent="1.5cm"/>
 <fo:region-after extent="2cm"/>
 <fo:region-body ... column-count="2" column-gap="1cm"/>
</fo:simple-page-master>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/adv-page/column-ex1.html]

### 11.9.2 Séquence avec page de titre

Si de nombreux documents simples *(bons de commande, factures, articles, ...)* peuvent se contenter d'un unique modèle de page, il en est d'autres *(thèses, cours, livres, ...)* qui nécessitent d'enchaîner des pages dont la présentation varie en fonction du contexte.

Le modèle développé ci-dessous permet d'enchaîner une page de titre, suivie par une série de pages identiques. La page de titre possède des marges différentes des pages normales, et ne comporte pas d'entête ni de bas de page. Le contenu des pages normales est présenté sur deux colonnes.

La première chose à faire consiste à décrire les divers modèles de page qui seront utilisés par la suite :

```
<fo:layout-master-set>

 <fo:simple-page-master master-name="premiere-page" ... >
  <fo:region-body margin-top="10cm" margin-bottom="2.1cm"/>
 </fo:simple-page-master>

 <fo:simple-page-master master-name="page-normale" ... >
  <fo:region-before extent="1.5cm"/>
  <fo:region-after extent="2cm"/>
  <fo:region-body ... column-count="2" column-gap="1.5cm"/>
 </fo:simple-page-master>
 ...
</fo:layout-master-set>
```

La seconde étape conduit à mettre en place la séquence suivant laquelle on passe d'un modèle de page à un autre :

```
<fo:layout-master-set>
 ...
 <fo:page-sequence-master master-name="document">
  <fo:repeatable-page-master-alternatives>
   <fo:conditional-page-master-reference
       master-reference="premiere-page" page-position="first"/>
   <fo:conditional-page-master-reference
       master-reference="page-normale" page-position="rest"/>
  </fo:repeatable-page-master-alternatives>
 </fo:page-sequence-master>
</fo:layout-master-set>
```

Il n'y a plus ensuite qu'à faire référence au modèle de document tel que défini ci-dessus lors de la génération d'une séquence de pages :

```
<fo:page-sequence master-reference="document" ... >
 ...
</fo:page-sequence>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/adv-page/cover-ex1.html]

Pour obtenir un document comportant plusieurs séquences introduites par des pages de titre *(cf. chapitres d'un livre par exemple)*, il suffit de faire se succéder plusieurs éléments *"fo:page-sequence"* *(la condition page-position="first" s'adresse à la première page d'une séquence)*.

### 11.9.3 Pages paires et impaires

#### > Séquence simple

Le principe permettant de générer un document avec un modèle différent pour les pages paires et impaires est exactement le même. La seule différence se situe au niveau du test associé à l'élément *"fo:conditional-page-master-reference"* :

```
<fo:repeatable-page-master-alternatives>
 <fo:conditional-page-master-reference
     master-reference="page-paire" odd-or-even="even"/>
 <fo:conditional-page-master-reference
     master-reference="page-impaire" odd-or-even="odd"/>
</fo:repeatable-page-master-alternatives>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/adv-page/evenodd-ex1.html]

#### > Entêtes et bas de page adaptés

Pour disposer d'entêtes ou de bas de pages différents suivant les modèles de page, il faut commencer par nommer les régions spécifiques au sein des modèles *(les valeurs "xsl-region-before" et "xsl-region-after" ne sont que des valeurs par défaut)* :

**modèle pages impaires**

```
<fo:simple-page-master master-name="page-impaire" ... >
 <fo:region-before extent="1.5cm" region-name="haut-impair/>
 <fo:region-after extent="2cm"/>
 <fo:region-body ... column-count="2" column-gap="1.5cm"/>
</fo:simple-page-master>
```

**modèle pages paires**

```
<fo:simple-page-master master-name="page-paire" ... >
 <fo:region-before extent="1.5cm" region-name="haut-pair/>
 <fo:region-after extent="2cm"/>
 <fo:region-body ... column-count="2" column-gap="1.5cm"/>
</fo:simple-page-master>
```

Il faut ensuite définir le contenu des zones ainsi évoquées à l'aide d'éléments *"fo-static-content"* :

**entête pages impaires**

```
<fo:static-content flow-name="haut-impair">
 <fo:block text-align="right" font-size="8pt">
  Exemple de document XSL-FO
 </fo:block>
</fo:static-content>
```

**entête pages paires**

```
<fo:static-content flow-name="haut-pair">
 <fo:block text-align="left" font-size="8pt">
  Exemple de document XSL-FO
 </fo:block>
</fo:static-content>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/adv-page/evenodd-ex2.html]

### 11.9.4 Séquences de pages complexes

#### > Séquences pair/impair avec page de titre

La logique permettant d'enchaîner des séquences de pages comportant à chaque fois une page de titre, suivie par des pages paires et impaires dont le modèle diffère, est plus complexe. En effet, on désire en général que le titre se trouve sur une page impaire

Cette contrainte est matérialisée par la propriété *"initial-page-number"* de l'élément *"fo:page-sequence"* concerné. Les valeurs possibles sont *"auto"*, *"auto-odd"*, *"auto-even"* ou un numéro de page. La valeur par défaut est *"auto"*.

```
 <fo:page-sequence ... initial-page-number="auto-odd">
  ...
 </fo:page-sequence>
```

Le modèle des pages à enchaîner dans une même séquence est :

```
<fo:repeatable-page-master-alternatives>
 <fo:conditional-page-master-reference
     master-reference="premiere-page" page-position="first"/>
 <fo:conditional-page-master-reference
     master-reference="page-paire" odd-or-even="even"/>
 <fo:conditional-page-master-reference
     master-reference="page-impaire" odd-or-even="odd"/>
</fo:repeatable-page-master-alternatives>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/adv-page/complex-ex1.html]

#### > Pages blanches

L'exemple ci-dessus a montré qu'un processeur d'impression peut être amené à insérer des pages blanches pour satisfaire certaines conditions d'enchaînement des séquences. Une dernière subtilité consiste à spécifier un modèle différent pour les pages blanches.

En prenant en compte cette dernière possibilité, les propriétés conditionnelles qui s'appliquent aux éléments *"fo:conditional-page-master-reference"* pour sélectionner le modèle de page approprié sont donc :

- *"page-position"* : valeurs possibles *"first"*, *"last"* ou *"rest"*,
- *"odd-or-even"* : valeurs possibles *"even"*, *"odd"* ou *"any"*,
- *"blank-or-not-blank"* : valeurs possibles *"blank"*, *"not-blank"* ou *"any"*.

Le modèle de la séquence de pages devient :

```
<fo:repeatable-page-master-alternatives>
 <fo:conditional-page-master-reference
     master-reference="premiere-page" page-position="first"/>
 <fo:conditional-page-master-reference
     master-reference="page-blanche" blank-or-not-blank="blank"/>
 <fo:conditional-page-master-reference
     master-reference="page-paire" odd-or-even="even"/>
 <fo:conditional-page-master-reference
     master-reference="page-impaire" odd-or-even="odd"/>
</fo:repeatable-page-master-alternatives>
```

et pour compléter le tout, les sauts de page après la page de titre ont été modifiés de manière à ce que la première page de contenu soit elle aussi une page impaire :

```
<fo:block break-after="odd-page"/>
```

Exemple :
[http://dmolinarius.github.io/demofiles/mod-84/xslfo/adv-page/complex-ex2.html]