Introduction à XSL/FO

Code: xml-xslfo

Originaux

url: **http://tecfa.unige.ch/guides/tie/html/xml-xslfo/xml-xslfo.html**

url: **http://tecfa.unige.ch/guides/tie/pdf/files/xml-xslfo.pdf**

Auteurs et version

- Daniel K. Schneider - Vivian Synteta

- Version: 0.4 (qualité brouillon !! - modifée le 14/8/01 par DKS)

Prérequis

- XML de base

  Module technique précédent: xml-dom

  Module technique précédent: xml-tech (matière obligatoire!)

  Module technique précédent: xml-xslt (transformations XSL)

Autres modules

Module technique suivant: xml-ser (server-side XML avec XSLT)

Abstract

XSL / FO est un langage qui permet de formatter l'affichage et / ou l'impression d'un document XML. Il s'agit de l'équivalent (en plus puissant) des style-sheet CSS.

Objectifs

- Formattage de base avec XSL / FO
- Utilisation de Apache / FOP

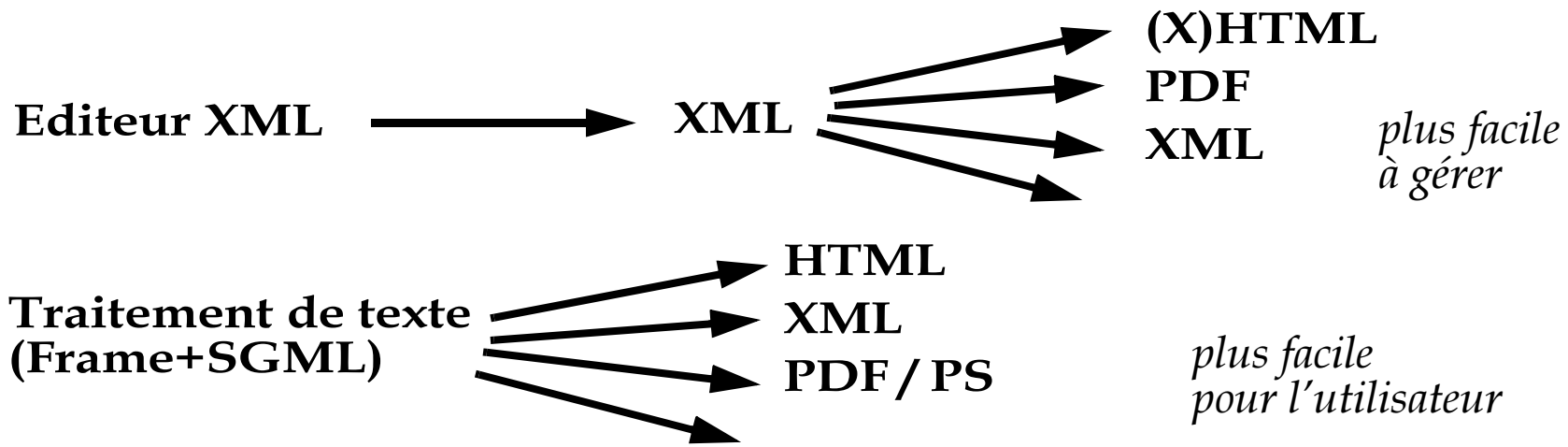
1. Table des matières détaillée

2. Le contexte: publier avec XML

2.1 Edition "manuelle" de XML

- Outils permettant d'éditer un "arbre" (quelques programmes Java gratuits)
- Outils d'édition de texte structuré (éditeurs de programmation comme Emacs)
- Outils semi-professionnels (comme XMetal ou EpcEdit): assez chers.
- Outils professionnels SGML / XML comme FrameMaker+SGML: chers.)
- Plug-ins pour traitement de texte (médiocres encore)
- Filtres vers XML (HTML, RTF, Latex, etc.): médiocres par nature

2 grandes options:



2.2 Langages de base pour systèmes de "textes" on-line

- Markup: Langage pour caractériser des éléments d'information
- Style: Langage pour définir la mise en page d'une classe d'objets
- Linking: Langage pour représenter des liens entre éléments et objets
- Assemblage: Assembler du texte à partir de fragments d'autres textes
- Scripting: Interface et langages pour créer des applications client-side

	monde HTML	**monde XML**	**monde SGML**
Linking	(<A> Tag dans HTML)	Xlink (+ Xpointer & Xpath)	HyTime & TEI
Assemblage	"calculs server-side"	XInclude (+ Xpointer & Xpath) ou entités ou "calculs server-side"	Entités SGML
Style	CSS2 CSS1	XSL (CSS)	DSSL
Markup	HTML	applications XML (XHTML, Docbook)	applications SGML (Docbook, TEI, ...)
Multimédia	formats "exotiques" (Flash, Gif, Jpeg)	formalismes XML (SVG, SMIL, MathML)	
Interface entre Markup et Scripting	Document Object Model (DOM)		
Scripting	Javascript, JScript, ECMAScript, ...		

2.3 Vocabulaires pour "publishing"

1. Text markup général: Latex en mieux

   Vocabulaires "neutres" mais très détaillés pour rédiger des textes larges. Ces vocabulaires ont souvent leur origine dans le monde "SGML".

- Exemple Docbook **http://www.docbook.org/xml/**
  - répandu dans le monde technique
  - support au niveau des outils XML (et SGML) haut de gamme
  - très détaillé (> 300 éléments)

2. Text markup décentralisé

   Schémas servant à générer et assembler du texte à partir de multiples sources selon besoin
   - Exemple DITA **http://www-106.ibm.com/developerworks/xml/library/x-dita3/index.html**

3. Par domaine: EduML ??

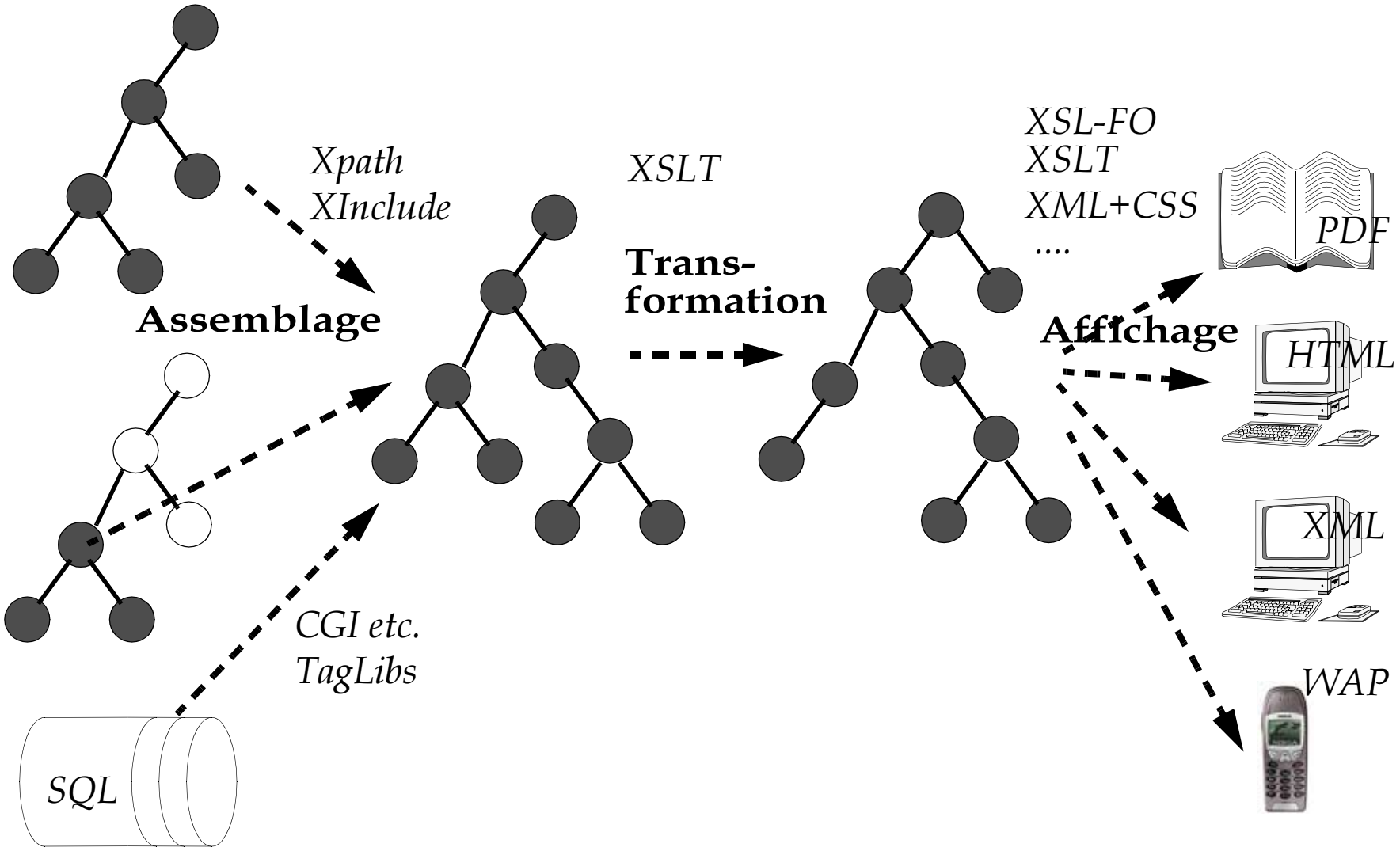
   Vocabulaires pour un domaine précis (souvent juste pour échanger des données)
   - Il n'existe pas encore de standard accepté pour les textes "educatifs".

4. Pour échange de données:
   - par exemple des News

5. Multimédia: SVG, Web3D, MathML, ...

Vers plus d'efficacité, vers un nouveau Babylon, vers une bureaucratisation ???

2.4 Exemple d'une chaîne de production "Web publishing"



3. Principes de base XSL/FO

Spécification de XSL

url: http://www.w3.org/TR/xsl/

But

- Qualité d'affichage de haut niveau (équivalent à celle d'un bon traitement de texte)
- Adaptation aux média (browser, imprimante, ...)
- Multi-culturel

Usage

- XSL + XSLT + XSLFO (-> FO) -> format imprimable / affichable
- FO -> format imprimable / affichable

  (FO est du XML avec XSL/FO mélange)

Statut

- XSL/FO n'est pas encore une recommendation du W3C (mais presque)
- Mozilla a prévu une implémentation de XSL et XSLT (pour le moment juste CSS2). Il existe des plugins expérimentaux pour XSL-FO.
- Microsoft fait comme toujours un peu sa propre cuisine (XSL est traduit en HTML)
- Il existe plusieurs outils "server-side" (dont Cocoon) qui incluent un processeur FO

3.1 Organisation d'une "page" FO

Exemple 3-1: Un simple stylesheet XSLT + FO

url: **http://tecfa.unige.ch/guides/xml/cocoon/xslfo/hello-page.sxml**
url: **http://tecfa.unige.ch/guides/xml/cocoon/xslfo/hello-page.sxml.text**
url: **http://tecfa.unige.ch/guides/xml/cocoon/xslfo/hello-page-xslfo.xsl**

- Note: L'exemple ci-dessus est composé d'un fichier XML et d'un fichier XSLT / XSL-FO et il est traité avec Cocoon 1
- Le code contient quelques instructions de traitement qu'on ignorera ici

A. Source XML

```
<page>
 <title>Hello Apache/FOP and Apache/Cocoon friends</title>
<content>
Here is some content. It should work with FOP 0.18 (and hopefully above).
It is totally uninteresting. It is totally uninteresting. ... </content>

<content>
Here is some more content.
It is slightly uninteresting.  .... :)
</content>

 <comment>Written by DKS/Tecfa 6/01</comment>
</page>
```

B. Feuille de style XSL/FO

```
<?xml version="1.0"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format" version="1.0" >

<!-- rule for the whole document: root element is page -->

<xsl:template match="page">
 <fo:root>
  <fo:layout-master-set>

  <!-- Definition of a single master page. It is simple (no headers etc.) -->
  <fo:simple-page-master
          master-name="first"
          margin-left="2cm"  margin-right="2cm"
          margin-bottom="0.5cm" margin-top="0.75cm"
          page-width="21cm" page-height="29.7cm"
          >
   <!-- required element body -->
   <fo:region-body/>
  </fo:simple-page-master>
 </fo:layout-master-set>

 <!-- Definition of a page sequence -->
 <fo:page-sequence master-name="first">
  <fo:flow flow-name="xsl-region-body" font-size="14pt" line-height="14pt">
```

```
    <xsl:apply-templates/>
  </fo:flow>
 </fo:page-sequence>
</fo:root>
</xsl:template>

<!-- A series of XSLT rules that produce fo:blocks to be inserted above -->

 <xsl:template match="page/title">
    <fo:block font-size="36pt" text-align="center" line-height="40pt" space-
before="0.5cm" space-after="1.0cm">
    <xsl:apply-templates/></fo:block>
  </xsl:template>

 <xsl:template match="content">
    <fo:block text-align="justify"  space-before="0.5cm">
    <xsl:apply-templates/></fo:block>
  </xsl:template>

 <xsl:template match="comment">
    <fo:block font-size="12pt" text-align="start" space-before="0.7cm" font-
style="italic">
    <xsl:apply-templates/></fo:block>
  </xsl:template>

</xsl:stylesheet>
```

C. Sequelette XSL-T/FO de base

```
<xsl:template match="page">
 <fo:root>
  <fo:layout-master-set>

  <!-- Definition of a single master page. It is simple (no headers etc.) -->
  <fo:simple-page-master
          master-name="first"  >
   <!-- required element body -->
   <fo:region-body/>
  </fo:simple-page-master>
 </fo:layout-master-set>

 <!-- Definition of a page sequence -->
 <fo:page-sequence master-name="first">
  <fo:flow flow-name="xsl-region-body" font-size="14pt" line-height="14pt">
    <xsl:apply-templates/>
  </fo:flow>
 </fo:page-sequence>
</fo:root>
</xsl:template>
```

- Une règle XSLT qui définit la racine FO pour la racine XML

Architecture de la racine FO,

Sous la racine fo:root il y a toujours:

- un **`fo:layout-master-set`** qui définit un ou plusieurs page layouts définis avec **`fo:simple-page-master`**
- des déclarations à option **`fo:declarations`**
- une sequénce d'un ou plusieurs **`fo:page-sequences`** qui contiennent du texte (litéralement ou fournis par XSLT) et des instructions de formattage.

4. Pagination

Exemple 4-1: Un simple stylesheet XSLT + FO

url: **http://tecfa.unige.ch/guides/xml/cocoon/xslfo/formcont.sxml**
url: **http://tecfa.unige.ch/guides/xml/cocoon/xslfo/formcont.sxml.text**
url: **http://tecfa.unige.ch/guides/xml/cocoon/xslfo/formcont-fo.xsl**

```
 <xsl:template match="training-course">
  <xsl:processing-instruction name="cocoon-format">type="text/xslfo"</
xsl:processing-instruction>
  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:layout-master-set>
       <fo:simple-page-master master-name="first"
                 page-height="29.7cm"
                 page-width="21cm"
                 margin-top="2cm"
                 margin-bottom="2cm"
                 margin-left="2.5cm"
                 margin-right="2.5cm">
     <fo:region-body margin-top="3cm"/>
     <fo:region-before extent="3cm"/>
     <fo:region-after extent="1.5cm"/>
    </fo:simple-page-master>

    <fo:simple-page-master master-name="right"
                 page-height="29.7cm"
                 page-width="21cm"
```

```
                margin-top="2cm"
                margin-bottom="2cm"
                margin-left="2.5cm"
                margin-right="2.5cm">
    <fo:region-body margin-top="2.5cm"/>
    <fo:region-before extent="2.5cm"/>
    <fo:region-after extent="1.5cm"/>
</fo:simple-page-master>

<fo:simple-page-master master-name="left"
                page-height="29.7cm"
                page-width="21cm"
                margin-top="2cm"
                margin-bottom="2cm"
                margin-left="2.5cm"
                margin-right="2.5cm">
    <fo:region-body margin-top="2.5cm"/>
    <fo:region-before extent="2.5cm"/>
    <fo:region-after extent="1.5cm"/>
</fo:simple-page-master>

<fo:page-sequence-master master-name="run">
      <fo:repeatable-page-master-alternatives maximum-repeats="no-limit" >
          <fo:conditional-page-master-reference
                master-name="left"
                odd-or-even="even" />
          <fo:conditional-page-master-reference
                master-name="right"
```

```
                    odd-or-even="odd" />
                <fo:conditional-page-master-reference
                    master-name="title"
                     />
            </fo:repeatable-page-master-alternatives>
    </fo:page-sequence-master>

  </fo:layout-master-set>
   <!-- end: defines page layout -->

<!-- actual layout -->
   <fo:page-sequence master-name="run" initial-page-number="1">

    <fo:static-content flow-name="xsl-region-before">
      <fo:block text-align="end" font-size="10pt" font-family="serif" line-
height="14pt" color="red" > Atelier Webmaster <fo:page-number/> </fo:block>
    </fo:static-content>

    <fo:flow flow-name="xsl-region-body" font-size="12pt" line-height="14pt">
       <xsl:apply-templates select="tc-courses/tc-course[position()=1]/course-
module[position()=1]"/>
      </fo:flow>
   </fo:page-sequence>
   </fo:root>

  </xsl:template>
```

5. Formattage des blocs et éléments spéciaux

- Le principe ressemble assez fortement à CSS (sauf qu'on utilise une syntaxe XML)
- Note: certains attributs CSS deviennent des éléments XSL ! (listes par exemple)

Exemple 5-1: Formattage CSS vs. XSL

CSS:

```
page > title {
     display: block;
     text-align: center; line-height: 40pt; ....
}
```

XSLT/FO:

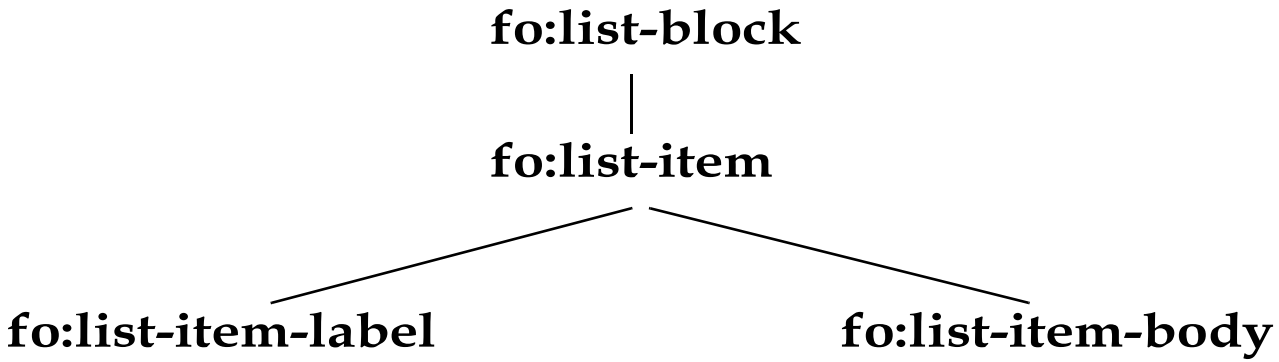
```
 <xsl:template match="page/title">
    <fo:block font-size="36pt" text-align="center"
              line-height="40pt" space-before="0.5cm"
              space-after="1.0cm">
    <xsl:apply-templates/></fo:block>
  </xsl:template>
```

FO simple:

```
 <fo:block font-size="36pt" text-align="center"
              line-height="40pt" space-before="0.5cm"
              space-after="1.0cm">
Hello Apache/FOP and Apache/Cocoon friends
    </fo:block>
```

5.1 Listes

- Mécanisme puissant: listes ordinaires, notes en bas de page, simples tables, etc.



- fo:list-block: contient la liste et contient quelques définitions pour la géométrie
- fo:list-item: enfants de fo:list-block, c.a.d. des items qui contiennent un label et un body (voir ci-dessous)
- fo: fo:list-item-label: contient le contenu du label (typiquement un fo:block avec un nombre, un caractère dingbat, etc.)
- The fo:list-item-body contient le corps d'un item, un ou plusieurs fo:block

Exemple 5-2: Une simple "bullet-list"

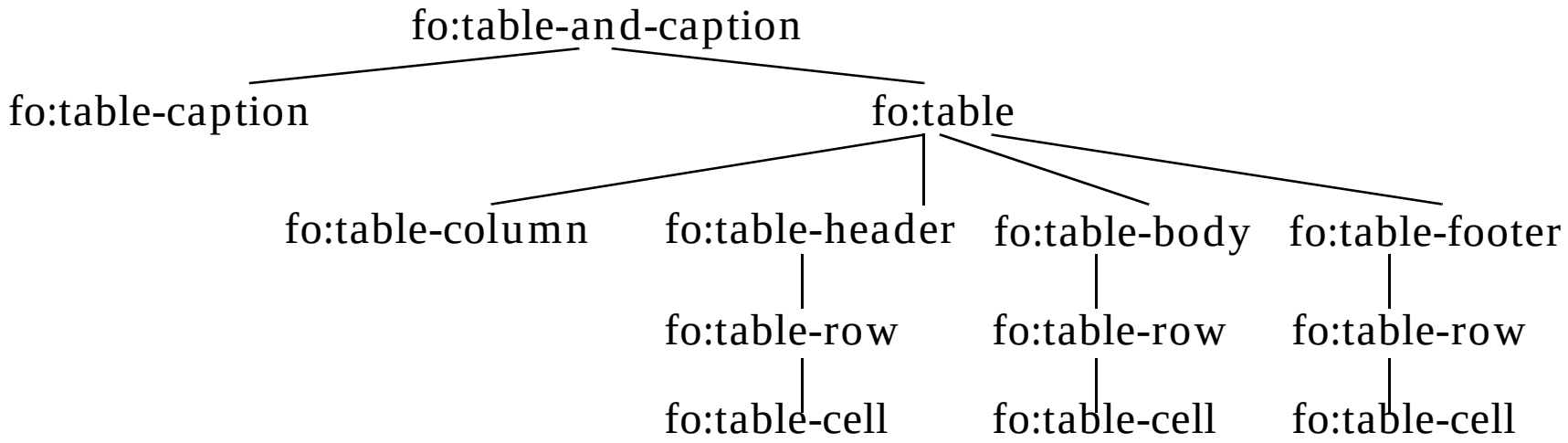
```
<xsl:template match="session-materials">

 Matériaux de cours:
 <fo:list-block space-before.optimum="6pt">
  <xsl:apply-templates select="session-material"/>
 </fo:list-block>
</xsl:template>

<xsl:template match="session-material">
 <fo:list-item space-before.optimum="8pt">
   <fo:list-item-label end-indent="label-end()">
   <fo:block>&#x2022;</fo:block>
  </fo:list-item-label>
  <fo:list-item-body start-indent="body-start()">
   <fo:block text-align="justify">
    <xsl:value-of select="@material-title"/>
   </fo:block>
   <fo:block text-align="justify" font-size="10pt">
    (URL:<xsl:value-of select="@material-url"/>)
    <xsl:value-of select="@material-comment"/>
   </fo:block>
  </fo:list-item-body>
 </fo:list-item>
</xsl:template>
```

5.2 Tables

- ressemblent un peu aux tables HTML



- fo:table-and-caption:
- fo:table-caption: La caption d'une table
- fo:table: la table proprement dite. Elle contien header et footer à option et un body.
- fo:table-column: permet de specifier notamment la longeur d'une colonne
- fo:table-header: Ligne entête, contient des lignes ou cellules
- fo:table-footer: Ligne "footer", contient des lignes ou cellules
- fo:table-body: contient des lignes ou cellules
- fo:table-row: contient des cellules qui contiennent des fo:blocks

Exemple 5-3: Table à deux colonnes fixes

```
 <xsl:template match="session">
  <fo:table space-before.optimum="6pt" text-align="center">
   <fo:table-column column-width="3cm"/> <fo:table-column column-width="13cm"/>
    <fo:table-body>
     <fo:table-row space-before.optimum="6pt">
      <fo:table-cell>
       <fo:block font-size="12pt" text-align="start" space-
before.optimum="5pt"><xsl:value-of select="session-date/@session-day"/>/
<xsl:value-of select="session-date/@session-month"/>/<xsl:value-of
select="session-date/@session-year"/></fo:block>
       <fo:block font-size="12pt" text-align="start" space-
before.optimum="5pt"><xsl:value-of select="session-component[position()=1]/
starthour"/> -
       <xsl:value-of select="session-component[position()=last()]/endhour"/>
</fo:block>
      </fo:table-cell>
      <fo:table-cell>
       <fo:block color="olive" font-size="12pt" text-align="start" space-
before.optimum="5pt"><xsl:value-of select="session-title"/>
       </fo:block>
       <fo:block font-size="12pt" text-align="start" space-
before.optimum="5pt"><xsl:apply-templates select="session-component"/>
       </fo:block>
      </fo:table-cell>
     </fo:table-row>
    </fo:table-body>  </fo:table>
```

6. Le processeur Apache / FOP

- FOP marche avec Cocoon (pas toujours la dernière version)
- On peut créer des servlets FOP avec n'importe quel serveur Java
- On peut générer des fichiers PDF (et autres formats) en mode "batch"
- FOP est une classe Java 2, et une distribution binaire est disponible

  url: **http://xml.apache.org/**

6.1 FOP avec Cocoon1

- Le processeur FOP s'utilise le plus souvent avec le processeur XSLT
- Le fichier XSL contient une processing instruction pour l'élément racine et une définition du name-space

Exemple 6-1: Un simple stylesheet XSLT + FO

url: **http://tecfa.unige.ch/guides/xml/cocoon/xslfo/hello-page.sxml**
url: **http://tecfa.unige.ch/guides/xml/cocoon/xslfo/hello-page.sxml.text**
url: **http://tecfa.unige.ch/guides/xml/cocoon/xslfo/hello-page-xslfo.xsl**

Fichier XML

```
<?xml version="1.0"?>
<?xml-stylesheet href="hello-page-xslfo.xsl" type="text/xsl"?>
<?cocoon-process type="xslt"?>
```

Fichier XSL / Cocoon

```
<?xml version="1.0"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
  version="1.0" >

<!-- rule for the whole document: root element is page -->

<xsl:template match="page">
    <xsl:processing-instruction name="cocoon-format">type="text/xslfo"
    </xsl:processing-instruction>

 <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
.......
 <!-- Definition of a page sequence -->
 <fo:page-sequence master-name="first">
  <fo:flow flow-name="xsl-region-body" font-size="14pt" line-height="14pt">
    <xsl:apply-templates/>
  </fo:flow>

 </fo:page-sequence>
</fo:root>

</xsl:template>
```

6.2 FOP en batch

A. Installation

- Il faut installer un Java 2 (si ce n'est pas déjà fait).
- Décomprimer l'archive FOP binaire qq part dans votre système.
- Alternativement vous pouvez installer Cocoon (Cocoon 2 pour un FOP récent)
- Ensuite faire / installer un fichier shell script pour lance l'application FOP qui se trouve dans la classe Java `org.apache.fop.apps.Fop`

Exemple 6-2: Fichier fop.bat à placer dans c:\bin de Windos

```
@echo off
set JAVA_HOME=C:\soft\jdk1.3
set path=%JAVA_HOME%\bin;%path%
REM echo path : %path%
echo Traitement FOP avec Java dans %java_home%
REM TOUT LE RESTE DOIT SE TROUVER SUR UNE SEULE LIGNE !!!
java -cp c:\soft\cocoon2\lib\fop-0_18_1.jar; c:\soft\cocoon2\lib\batik-libs.jar;
c:\soft\cocoon2\lib\xalan-2.1.0.jar; c:\soft\cocoon2\lib\xerces_1_4_0.jar;
c:\soft\cocoon2\lib\jimi-1.0.jar; c:\soft\cocoon2\lib\dom2.jar org.apache.fop.apps.Fop %1 %2 %3
%4 %5 %6 %7 %8
```

Note: c\bin doit se trouver dans le path !

Utilisation sur Solaris à TECFA

- Taper "fop" dans un terminal (sur tecfasun5)

(le script se trouve dans /local/go/)

B. Arguments de l'application FOP

- tapez "fop" dans un terminal

```
Syntaxe: Fop [options] [-fo|-xml] infile [-xsl file] [-awt|-pdf|-
   mif|-pcl|-txt|-print] <outfile>
```

```
[INPUT]
 infile            xsl:fo input file (the same as the next)
 -fo  infile       xsl:fo input file
 -xml infile       xml input file, must be used together with -xsl
 -xsl stylesheet   xslt stylesheet

[OUTPUT]
 outfile           input will be rendered as pdf file into outfile
 -pdf outfile      input will be rendered as pdf file (outfile req'd)
 -awt              input will be displayed on screen
 -mif outfile      input will be rendered as mif file (outfile req'd)
 -pcl outfile      input will be rendered as pcl file (outfile req'd)
 -txt outfile      input will be rendered as text file (outfile req'd)
 -print            input file will be rendered and sent to the printer
                   see options with "-print help"

[Examples]
 Fop foo.fo foo.pdf
 Fop -fo foo.fo -pdf foo.pdf (does the same as the previous line)
 Fop -xsl foo.xsl -xml foo.xml -pdf foo.pdf
 Fop foo.fo -mif foo.mif
 Fop foo.fo -print or Fop -print foo.fo
 Fop foo.fo -awt
```