Université de Nice-Sophia Antipolis — Vendredi 25 mai 2007
Polytech' CiP1 — Durée: 3h

Création et Manipulation de documents

(Hélène Renard / Sylvain Schmitz)

Travaux Dirigés – Séance n^{o}12

---

1 Objectifs du TD

L'objectif de cette séance est de découvrir les rouages internes du format *OpenDocument*. Les balises de structure et les notions de style sont abordées.

2 Éditer un document `odt`

Un document *OpenDocument* est en fait une archive JAR contenant divers fichiers :

`META-INF/manifest.xml` un fichier recensant le contenu de l'archive,

`mimetype` un fichier donnant le type MIME du document (vide),

`settings.xml` les règlages de l'application pour ce document (emplacement des fenêtres, choix de l'imprimante, ...),

`meta.xml` les meta-informations prtant sur le contenu du document,

`styles.xml` les styles utilisés dans le document,

`content.xml` le contenu du document à proprement parler.

Les fichiers sont dans des formats XML définis par le standard *OpenDocument*. Le contenu XML n'est pas très lisible par défaut, mais l'option « *Size optimization for XML format (no pretty printing)* » peut être désactivée dans *Options—Load/Save—General*.

Exercice n°1 : Utilisation de la commande `jar`.

1. Créez un répertoire `CMDocs/odoc/` et placez-vous à l'intérieur.
2. Lisez le manuel pour la commande `jar`.
3. Extrayez les fichiers d'un de vos documents `.odt`.
4. Modifiez une portion de texte de `content.xml`.
5. Recréez un document `CMDocs/test.odt` à partir du contenu de `CMDocs/odoc/` avec la commande `jar` et ouvrez-le sous OpenOffice.org.

3 Contenu d'un document `.odt`

Les fichiers qui nous intéressent pour ce cours sont `META-INF/manifest.xml` et `content.xml` ; en fait, ce sont les seuls fichiers indispensables. Le fichier `~schmitz/minimal.odt` vous servira de base de travail pour construire des documents `.odt`.

Exercice n°2 : Copiez `~schmitz/minimal.odt` dans votre répertoire `CMDocs`, supprimez le contenu du répertoire `CMDocs/odoc/` et ouvrez-y le fichier `minimal.odt` avec la commande `jar`.

3.1 Structure de `content.xml`

Le fichier `content.xml` a la structure suivante :

```
<?xml version="1.0" encoding="UTF-8"?>

<office:document-content déclaration des espaces de noms
  xmlns:office="urn:oasis:names:tc:opendocument:xmlns:office:1.0"
  xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"
  xmlns:style="urn:oasis:names:tc:opendocument:xmlns:style:1.0"
  ...
  office:version="1.0">

  <office:scripts/> pour des scripts embarqués, falcutatif

  <office:font-face-decls>
    déclaration des polices de caractères, falcutative
  </office:font-face-decls>

  <office:automatic-styles>
    informations de style embarquées, falcutatives
  </office:automatic-styles>

  <office:body>
    <office:text> ou office:drawing, ou office:presentation, ou office:spreadsheet...
      le contenu du document, obligatoire !
    </office:text>
  </office:body>

</office:document-content>
```

3.2 Espaces de nom

XML définit un système permettant de créer des balises modulaires, en mêlant des balises provenant de différents langages à balise au sein d'un même document grâce à la notion d'espace de noms. La définition d'un espace de nom en tête de document se fait par la déclaration

`xmlns:`*nom*`="`URI`"`

dans l'élément racine du document. Cette déclaration associe le préfixe *nom* à un dialecte XML précis via une URI (par exemple `http://www.w3.org/1999/xhtml` pour XHTML). Par la suite, tous les éléments et attributs du document utilisant le préfixe

nom`:element`

seront reconnus comme éléments du dialecte XML associé au préfixe *nom*.

Les espaces de noms que nous utiliserons durant cette séance sont

`office` à l'URI `urn:oasis:names:tc:opendocument:xmlns:office:1.0`, qui sert à décrire des informations globales pour tous les documents *OpenDocument*,

`text` à l'URI `urn:oasis:names:tc:opendocument:xmlns:text:1.0`, qui sert au contenu des documents textuels (avec l'extension `.odt`), et aux fragments de texte dans les autres documents,

`style` à l'URI `urn:oasis:names:tc:opendocument:xmlns:style:1.0`, qui sert à préciser le format des éléments du document,

`table` à l'URI `urn:oasis:names:tc:opendocument:xmlns:table:1.0`, qui sert à décrire des tableaux dans un document tableur, ou dans des documents textuels,

`draw` à l'URI `urn:oasis:names:tc:opendocument:xmlns:drawing:1.0`, pour les graphiques et images,

`fo` à l'URI `urn:oasis:names:tc:opendocument:xmlns:xsl-fo-compatible:1.0`, pour des éléments issus de XSL FO, un dialecte de mise en forme,

`xlink` à l'URI `http://www.w3.org/1999/xlink`, qui sert à réaliser des liens hypertextes,

`svg` à l'URI `urn:oasis:names:tc:opendocument:xmlns:svg-compatible:1.0`, pour des éléments issus de SVG, un dialecte de dessin vectoriel.

D'autres espaces de nom utiles sont `dc` pour les meta données *Dublin Core* et `meta` pour les autres meta données dans le fichier `meta.xml`, et `manifest` pour la liste des fichiers dans `META-INF/manifest.xml`.

Comme vous pouvez le voir, le format *OpenDocument* réutilise des éléments issus de plusieurs standards existants dans un souci d'interopérabilité et de facilité d'utilisation.

4 Éléments XML d'un document `.odt`

4.1 Paragraphes, titres, caractères

L'espace de nom associé au préfixe `text` prévoit plusieurs éléments qui devraient vous être familiers.

`text:p` est l'élément *paragraphe*,

`text:h` est l'élément de *titre*, avec un attribut `text:outline-level` qui donne le niveau d'importance du titre (`1` pour le plus important, puis croissant jusqu'à `10` pour le moins important),

`text:span` est l'élément pour identifier une suite de caractères.

Exercice n°3 : Ajoutez un paragraphe et un titre de second niveau à `odoc/content.xml`. Recompilez `test.odt` et vérifiez le résultat.

4.1.1 Polices de caractères

Les polices de caractères utilisées dans le document sont déclarées dans l'élément `office:font-face-decls`, qui contient une succession d'éléments `style:font-face` :

```
<office:font-face-decls>
  <style:font-face style:name="starbats"
                   svg:font-family="starbats"
                   style:font-charset="x-symbol"/>
  <style:font-face style:name="Times New Roman"
                   svg:font-family="&apos;Times New Roman&apos;"
                   style:font-family-generic="roman"
                   style:font-pitch="variable"/>
  <style:font-face style:name="Arial"
                   svg:font-family="Arial"
                   style:font-family-generic="swiss"
                   style:font-pitch="variable"/>
</office:font-face-decls>
```

4.1.2 Styles de caractères

Le style d'un élément quelconque du document peut être changé par l'attribut `style-name`, avec pour contenu le nom d'un style préalablement déclaré (par exemple dans l'élément `office:automatic-styles`). Un élément de style peut à son tour hériter des attributs d'un autre élément grâce à l'attribut `style:parent-style-name`.

Un élément `text:span` peut recevoir un style particulier par l'attribut `text:style-name`, qui appelle un style de la famille `text` et contient un élément `style:text-properties`. Par exemple, on peut définir un style nommé *Titaliques* par

```
<office:automatic-styles>
  <style:style style:name="Titaliques" style:family="text">
    <style:text-properties
     fo:font-style="italic"
     style:font-style-asian="italic"
     style:font-style-complex="italic"/>
  ...
</office:automatic-styles>
```

et l'utiliser dans le document par

```
<text:p>Un paragraphe avec
  <text:span text:style-name="Titaliques">des italiques</text:span>
</text:p>
```

Les mises en forme de caractères utilisent en général les éléments de l'espace de nom `fo`. Les styles XSL:FO reprennent tout ce que vous avez vu avec CSS et plus encore. En particulier

`fo:font-name` donne le nom d'une police de caractères déclarée précedemment,

`fo:font-style` peut être entre autres `normal` ou `italic`,

`fo:font-weight` peut être `bold`,

`fo:font-size` une longueur en points, par exemple `12pt`,

`fo:font-variant` peut être `small-caps`,

`fo:text-transform` `lowercase`, `uppercase`, ...

`fo:color`, `fo:background-color` une couleur sous forme hexadécimale, par exemple `#ff0000` pour du rouge,

`fo:text-undeline-*` où `*` peut être
- `style`, avec pour valeurs `none`, `dashed`, `dotted`, ...;
- `type` avec pour valeurs `none`, `single`, `double`;
- `width` avec pour valeurs `auto`, `normal`, `bold`, `thin`, ...;
- `color` par exemple `#ff0000`, ou `font-color` pour avoir la même couleur que le texte souligné.

`fo:text-shadow` une valeur de déplacement horizontal et vertical de l'ombre, par exemple `1pt`.

`...`

Exercice n°4 : Mettez une portion de texte de votre paragraphe en gras et une en gras et ombré en suivant le modèle précédent pour les italiques. Vérifiez le résultat sous OpenOffice.org.

4.1.3 Liens hypertextes

Un lien hypertexte est un élément `text:a` avec pour attribut `xlink:type="simple"` et l'URI dans l'attribut `xlink:href`.

```
<text:p>Un paragraphe avec un
  <text:a xlink:type="simple" xlink:href="http://www.exemple.fr/">lien hypertexte</text:a>.
</text:p>
```

Exercice n°5 : Ajoutez un lien hypertexte à votre document.

4.1.4 Styles de paragraphes

Les blocs (paragraphes, titres) sont mis en forme par des styles de la famille `paragraph`. Par exemple,

```
<style:style style:name="Pexemple" style:family="paragraph">
  <style:paragraph-properties fo:text-align="right" fo:margin-right="1cm"/>
  <style:text-properties fo:color="#ff0000"/>
</style:style>
```

Ce style est utilisé sur un paragraphe par

```
<text:p text:style-name="Pexemple">Un paragraphe mis en forme.</text:p>
```

Quelques-unes des possibilités offertes par XSL:FO sont

`fo:text-align` avec pour valeurs possibles `start`, `end`, `center` ou `justify`,

`fo:margin-left`, `fo:margin-right` avec une valeur positive pour les marges,

`fo:text-indent` avec une valeur positive ou négative, et des marges doivent être définies,

`fo:margin-top`, `fo:margin-bottom` pour les espaces avant et après les paragraphes,

`fo:background-color` pour la couleur de fond du paragraphe,

`fo:border` a pour valeur une entrée de la forme *épaisseur style couleur*, par exemple `1mm solid #000000` ou `thick double #0000ff` ; on peut spécifier une bordure individuelle par `fo:border-left`, `fo:border-right`, ... L'espace entre bordure et texte, pour peu qu'une bordure soit définie, est donné par `fo:padding`, ou `fo:padding-left`, `fo:padding-right`, ... pour les côtés individuels.

Exercice n°6 : Mettez en forme vos paragraphes pour être justifiés avec un espacement avant et après le paragraphe de `3mm` et une indentation de `1cm`.

4.1.5 Tabulations

Les tabulations sont gérées par des éléments `style:tab-stop` dans `style:paragraph-properties` comme suit :

```
<style:style style:name="Ptabs" style:family="paragraph">
  <style:paragraph-properties>
    <style:tab-stops>
      <style:tab-stop style:position="1cm" style:type="center"/>
      <style:tab-stop style:position="4cm" style:type="left"/>
    </style:tab-stops>
  </style:paragraph-properties>
</style:style>
```

Chaque tabulation est ensuite marquée par un élément `text:tab` dans le contenu :

```
<text:p text:style-name="Ptabs">
  <text:tab/>style
  <text:tab/>modèles de style
</text:p>
<text:p text:style-name="Ptabs">
  <text:tab/>fo
  <text:tab/><text:span text:style-name="Tsc">xsl:fo</text:span>,
    <text:span text:style-name="Tsc">xsl</text:span> Formating Objects
</text:p>
```

Exercice n°7 : Écrivez les définitions des acronymes XML et XHTML dans votre document à l'aide de tabulations. Vérifiez le résultat sous OpenOffice.org.

4.2 Listes

Syntaxe de base Une liste *OpenDocument* n'est pas très différente d'une liste XHTML :

```
<text:list>
  <text:list-item>
    <text:p>Premier élément d'une liste.</text:p>
  </text:list-item>
  <text:list-item>
    <text:p>Encore un élément.</text:p>
  </text:list-item>
  <text:list-item>
    <text:p>Dernier élément.</text:p>
  </text:list-item>
</text:list>
```

Mise en forme La mise en forme d'une liste nécessite l'ajout d'un attribut `style:list-style-name` au style des paragraphes qui reprend le nom d'un élément `text:list-style`. Ce dernier contient un élément `text:list-level-style-bullet` pour les listes à puce, ou `text:list-level-style-number` pour les listes numérotées, ce pour chaque niveau de liste :

```
<style:style style:name="P1" style:family="paragraph"
                             style:list-style-name="L1"/>
<text:list-style style:name="L1">
  <text:list-level-style-bullet text:level="1"
                                text:bullet-char="*">
    <style:list-level-properties text:space-before="0.25in"
                                 text:min-label-width="0.25in"/>
    <style:text-properties style:font-name="starbats"/>
  </text:list-level-style-bullet>

   <text:list-level-style-number text:level="2"
                                 style:num-suffix=":" style:num-format="1">
     <style:list-level-properties text:space-before="0.5in"
                                  text:min-label-width="0.25in"/>
  </text:list-level-style-number>
  et ainsi de suite...
</text:list-style>
```

Ce style est ensuite appliqué à notre liste comme suit :

```
<text:list text:style-name="L1">
  <text:list-item>
    <text:p text:style-name="P1">Premier élément d'une liste.</text:p>
  </text:list-item>
  <text:list-item>
    <text:p text:style-name="P1">Encore un élément.</text:p>
  </text:list-item>
  <text:list-item>
    <text:p text:style-name="P1">Dernier élément.</text:p>
  </text:list-item>
</text:list>
```

Options de mise en forme Les éléments `text:list-level-style-bullet` et `text:list-level-style-number` peuvent recevoir les attributs :

`text:level` le niveau d'emboîtement de la liste, entre `1` et `10`,

`text:bullet-char` le symbole utilisé en tête de liste pour des entrées à puce,

`text:num-prefix`, `text:num-prefix` les caractères à insérer avant et après le numéro d'une entrée numérotée,

`text:num-format` le format des entrées numérotées, `1` pour des chiffres arabes, `a` et `A` pour des lettres latines, `i` et `I` pour des chiffres romains.

L'élément `style:list-level-properties` inclus permet de déterminer l'espace occupé par le numéro ou la puce par l'attribut `text:min-label-width`, et le niveau d'indentation par l'attribut `text:space-before`. Enfin, les listes à puce contiennent aussi un élément `style:text-properties` qui donne la police de caractères utilisée pour la puce.

Exercice n°8 : Créez deux listes emboîtées numérotées qui ressemblent à la liste suivante :

1. Chats
   (a) noirs
   (b) blancs
2. Chiens
3. Poissons

4.3 Tableaux

Les tableaux embarqués dans du texte utilisent néanmoins l'espace de nom désigné par le préfixe `table`.

Encore une fois, la syntaxe de base est très proche de celle d'XHTML :

```
<table:table table:name="tableau">

  les caractéristiques des colonnes
  <table:table-column table:number-columns-repeated="3"/>

  <table:table-row> première ligne
      <table:table-cell office:value-type="string">
        <text:p>Première cellule</text:p>
      </table:table-cell>
      <table:table-cell office:value-type="string">
        <text:p>Deuxième cellule</text:p>
      </table:table-cell>
      <table:table-cell office:value-type="string">
        <text:p>Troisième cellule</text:p>
      </table:table-cell>
  </table:table-row>

  <table:table-row> deuxième ligne
    <table:table-cell office:value-type="string">
      <table:table-cell office:value-type="string">
        <text:p>Première cellule</text:p>
      </table:table-cell>
      <table:table-cell office:value-type="string">
        <text:p>Deuxième cellule</text:p>
      </table:table-cell>
      <table:table-cell office:value-type="string">
        <text:p>Troisième cellule</text:p>
      </table:table-cell>
  </table:table-row>

</table:table>
```

L'élément `table:table-column` permet de spécifier le style de chaque colonne du tableau et de répéter ce style pour plusieurs colonnes avec l'attribut `table:number-columns-repeated`. Même en l'absence de mise en forme, il faut donner ici le nombre de colonnes comme dans l'exemple.

Chaque ligne du tableau est ensuite représentée par un élément `table:table-row`. Les rangées qui servent de titres aux colonnes peuvent être mises dans un élément `table:table-header-rows`.

Chaque ligne contient une suite de cellules `table:table-cell`. Une cellule qui s'étend sur plusieurs colonnes possède un attribut `table:number-columns-spanned`, et est suivie d'autant de cellules vides `table:covered-table-cell` que nécessaire.

Exercice n°9 : Insérez le tableau de l'exemple dans votre document. Changez-le pour que la deuxième ligne contienne une seule cellule à la place des deux premières. Vérifiez le résultat sous OpenOffice.org.

4.4 Images

Les images utilisent des éléments de l'espace de nom préfixé par `draw`. Un fichier image peut être ajouté à votre archive dans le répertoire `Pictures`, par exemple l'image `Pictures/pomme.gif`.

Lien vers l'image

```
<text:p>
  <draw:frame draw:name="pomme" text:anchor-type="paragraph"
              svg:width="1.0417in" svg:height="0.7811in"
              draw:z-index="0">
    <draw:image xlink:href="Pictures/pomme.gif" xlink:type="simple"
                xlink:show="embed" xlink:actuate="onLoad"/>
  </draw:frame>
</text:p>
```

L'image est ajoutée en deux étapes :

1. d'abord, l'insertion d'un cadre `draw:frame`, avec pour attributs le nom de l'image et sa taille,
2. ensuite avec un élément `draw:image` qui référence l'image par un lien `xlink:href`.

Mise à jour de `manifest.xml` Puisque nous changeons le contenu de l'archive JAR, il faut mettre à jour la liste des fichiers dans `manifest.xml` et ajouter les deux lignes :

```
<manifest:file-entry
 manifest:media-type="image/gif"
 manifest:full-path="Pictures/pomme.gif"/>

<manifest:file-entry
 manifest:media-type=""
 manifest:full-path="Pictures/"/>
```

Exercice n°10 : Ajoutez une image à votre document.

5 Traduction vers XHTML

Exercice n°11 : Utilisez l'export XHTML sous OpenOffice.org. Que dire du résultat ?