INFO-H-509 : Technologies XML
TP 4 - Corrigé

Professeur : Stijn Vansummeren
Assistant : François Picalausa
http://cs.ulb.ac.be/public/teaching/infoh509

Année académique 2009-2010

Exercice 1.1

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

  <xsl:template match="/">
    <html>
    <head><title>Customers</title></head>
    <body>
      <xsl:apply-templates select="Root/Customers/Customer" />
    </body>
    </html>
  </xsl:template>

  <xsl:template match="Customer">
    <h1><xsl:value-of select="ContactName" /></h1>
    <p><xsl:value-of select="ContactTitle" /> of <xsl:value-of select="CompanyName" /></p>
    <p>Contact:</p>
    <ul>
      <xsl:apply-templates select="Phone | Fax" />
    </ul>
    <address>
      <xsl:value-of select="FullAddress/Address" /><br />
      <xsl:value-of select="FullAddress/City" />, <xsl:value-of
      select="FullAddress/Region" /><xsl:text> </xsl:text><xsl:value-of
      select="FullAddress/PostalCode" /><br />
      <xsl:value-of select="FullAddress/Country" />
    </address>
  </xsl:template>

  <xsl:template match="Phone | Fax">
    <li><xsl:value-of select="name()" />: <xsl:value-of select="." /></li>
  </xsl:template>
</xsl:stylesheet>
```

Exercice 1.2

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="http://www.w3.org/1999/xhtml">
  <xsl:template match="/">
    <html>
      <head><title>Comments</title></head>
      <body>
        <xsl:apply-templates select="//Comment">
          <xsl:sort select="../OrderDate" order="descending" />
        </xsl:apply-templates>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="Comment">
    <h1><xsl:value-of select="../CustomerID" /></h1>
    <!-- <p>On <xsl:value-of select="../OrderDate" /></p> -->
    <xsl:copy-of select="*[namespace-uri() eq 'http://www.w3.org/1999/xhtml']" />
  </xsl:template>
</xsl:stylesheet>
```

Exercice 1.3

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <customers>
      <xsl:apply-templates select="Root/Customers/Customer" />
    </customers>
  </xsl:template>

  <xsl:template match="Customer">
    <xsl:variable name="id" select="@CustomerID" />
    <customer id="{$id}">
      <xsl:attribute name="orders">
        <xsl:value-of select="count(//Order[CustomerID eq $id])" />
      </xsl:attribute>
    </customer>
  </xsl:template>
</xsl:stylesheet>
```

Une autre solution qui utilise le groupement de XSLT 2.0 :

```
<?xml version="1.0"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <customers>
      <xsl:for-each-group select="//Order" group-by="CustomerID">
        <customer id="{current-grouping-key()}">
          <xsl:attribute name="orders">
```

```
            <xsl:value-of select="count(current-group())" />
          </xsl:attribute>
        </customer>
      </xsl:for-each-group>
    </customers>
  </xsl:template>
</xsl:stylesheet>
```

Exercice 1.4

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <customers>
      <xsl:apply-templates select="Root/Customers/Customer" />
    </customers>
  </xsl:template>

  <xsl:template match="Customer">
    <xsl:variable name="id" select="@CustomerID" />
    <customer id="{$id}">
      <xsl:apply-templates select="//Order[CustomerID = $id]">
        <xsl:sort select="OrderDate" order="descending" />
      </xsl:apply-templates>
    </customer>
  </xsl:template>

  <xsl:template match="Order">
    <xsl:copy-of select="." />
  </xsl:template>
</xsl:stylesheet>
```

Exercice 1.5

Pour éviter de dupliquer du code, au lieu de créer un template différent par catégorie, nous utilisons un seul template qui prend pour paramètre le nom d'une catégorie, son poids minimum et son poids maximum. Noter qu'ajouter `value` a un paramètre permet de lui fixer une valeur par défaut.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <weight>
      <xsl:call-template name="weightcategory">
        <xsl:with-param name="min" select="500" />
        <xsl:with-param name="name" select="'heavy'" />
      </xsl:call-template>
      <xsl:call-template name="weightcategory">
        <xsl:with-param name="min" select="10" />
        <xsl:with-param name="max" select="500" />
        <xsl:with-param name="name" select="'medium'" />
      </xsl:call-template>
      <xsl:call-template name="weightcategory">
```

```
      <xsl:with-param name="max" select="10" />
      <xsl:with-param name="name" select="'light'" />
    </xsl:call-template>
  </weight>
</xsl:template>

<xsl:template name="weightcategory">
  <xsl:param name="min" select="-1" />
  <xsl:param name="max" select="-1" />
  <xsl:param name="name" />
  <xsl:element name="{$name}">
    <xsl:for-each select="distinct-values(//ShipInfo[
      ($max lt 0 or number(Freight) lt $max) and
      ($min lt 0 or number(Freight) ge $min)]/../CustomerID)">

      <customer id="{.}"/>
    </xsl:for-each>
  </xsl:element>
</xsl:template>
</xsl:stylesheet>
```

Exercice 1.5b Classement par poids moyen :

```
<?xml version="1.0"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <weight>
      <!-- Group orders by customer ID and compute average weight for each customer -->
      <xsl:variable name="customers">
        <xsl:for-each-group select="//Order" group-by="CustomerID">
          <customer id="{current-grouping-key()}"
                weight="{avg(current-group()/ShipInfo/Freight)}" />
        </xsl:for-each-group>
      </xsl:variable>

      <xsl:call-template name="weightcategory">
        <xsl:with-param name="min" select="500" />
        <xsl:with-param name="name" select="'heavy'" />
        <xsl:with-param name="customers" select="$customers" />
      </xsl:call-template>
      <xsl:call-template name="weightcategory">
        <xsl:with-param name="min" select="10" />
        <xsl:with-param name="max" select="500" />
        <xsl:with-param name="name" select="'medium'" />
        <xsl:with-param name="customers" select="$customers" />
      </xsl:call-template>
      <xsl:call-template name="weightcategory">
        <xsl:with-param name="max" select="10" />
        <xsl:with-param name="name" select="'light'" />
        <xsl:with-param name="customers" select="$customers" />
      </xsl:call-template>
```

```
    </weight>
  </xsl:template>

  <xsl:template name="weightcategory">
    <xsl:param name="min" select="-1" />
    <xsl:param name="max" select="-1" />
    <xsl:param name="name" />
    <xsl:param name="customers" />

    <xsl:element name="{$name}">
      <xsl:for-each select="$customers/customer[
        ($max lt 0 or number(@weight) lt $max) and
        ($min lt 0 or number(@weight) ge $min)]">
        <customer id="{@id}"/>
      </xsl:for-each>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

Exercice 1.6

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="comment()" priority="1" />

  <xsl:template match="@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

Cette solution applique le template `match="@*|node()"` récursivement, de sorte à copier tous les noeuds et attributs du document. Les commentaires sont supprimés en ajoutant la règle `match="comment()"`. Selon XSLT 2.0, ces deux règles ont la même priorité et il est donc nécessaire de les départager en spécifiant laquelle des deux règles a priorité sur l'autre.

Exercice 1.7

Il importe d'utiliser `xsl:output` pour obtenir du texte. Noter aussi que la solution est présentée de sorte à faciliter la lecture. Conserver cette indentation introduit des espaces et des retours à la ligne dans le fichier en sortie.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xsl:output method="text" />

  <xsl:template match="Customer">
    <xsl:variable name="id" select="@CustomerID" />
    <xsl:value-of select="$id" />,
    <xsl:value-of select="FullAddress/Country" />,
```

```
    <xsl:choose>
      <xsl:when test="Fax">
        <xsl:value-of select="Fax[1]" />
      </xsl:when>
      <xsl:otherwise>NULL</xsl:otherwise>
    </xsl:choose>,
    <xsl:value-of select="max(//Order[CustomerID eq $id]/OrderDate/xs:dateTime(text()))" />
    <xsl:text>
</xsl:text></xsl:template>

  <xsl:template match="*">
    <xsl:apply-templates select="*" />
  </xsl:template>
</xsl:stylesheet>
```

Exercice 1.8

La fonction XPath `document` permet d'ouvrir un second document. L'écriture vers différents fichiers demande l'utilisation de `xsl:result-document` qui n'est présent qu'à partir d'XSLT 2.0.

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

  <xsl:template match="/">
    <xsl:for-each select="(document('orders.xml') | document('customers.xml'))//Customer">
      <xsl:result-document href="{concat(@CustomerID, '.html')}">
        <html>
        <head><title><xsl:value-of select="CompanyName" /></title></head>
        <body>
          <xsl:apply-templates select="." />
        </body>
        </html>
      </xsl:result-document>
    </xsl:for-each>
  </xsl:template>

  <!-- la suite est identique a l'exercice 1 -->
  <xsl:template match="Customer">
    ...
  </xsl:template>
  <xsl:template match="Phone | Fax">
    ...
  </xsl:template>
</xsl:stylesheet>
```