INFO-H-509 : Technologies XML
TP 3 - Correction

Professeur : Stijn Vansummeren
Assistant : Michaël Waumans
http://cs.ulb.ac.be/public/teaching/infoh509

2015 - 2016

Exercise 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>
```

Exercise 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>
```

Exercise 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>
```

An other solution that uses grouping from 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>
```

Exercise 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>
```

Exercise 1.5

To avoid duplicating code, instead of creating a different template for each category, we use only one template that takes as parameters the name of a category, it's maximum weight and it's minimum weight. Notice that adding `value` to a parameter allows to give it a default value.

```
<?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>
```

Exercise 1.5b Sorting by average weight :

```
<?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>
```

Exercise 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>
```

This solution applies the template `match="@*|node()"` recursively, so that it copies all the nodes and attributes of the document. Comments are removed by adding the rule `match="comment()"`. According to XSLT 2.0, those two rules have the same priority, so it is necessary to specify which one has the priority over the other.

Exercise 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>
```

Exercise 1.8

The XPath function `document` allows to open another document. Writing to different files requires the use of `xsl:result-document` that is only present since 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">
    <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>
```