college CD

Data processing technologies (TTD)

Class 2

XML (suite)

XLS

Standing for *eXtensible Stylesheet Language*, XLS is a styling language for XML consisting of four parts:

XSLT
A language for transforming XML documents.

XPath
A language for navigating in XML documents.

XSL-FO
A language for formatting XML documents (discontinued in 2013).

XQuery
A language for querying XML documents.

What is XSLT?

Standing for XSL Transformations, XLTS is the most important part of XLS allowing to transform a given XML document into another XML document or into another type of document readable by a browser such as HTML and XHTML usually by transforming each XML element into an XHTML element. In order to navigate within an XML document, XPath will be used.

XSLT makes it possible to add or remove elements and attributes to or from the output file, rearrange and sort elements, perform tests and decide to hide or display specific elements, etc.

Basically, XSLT transforms an XML source-tree into an XML result-tree.

Style Sheet Declaration

After declaring de document type definition (DTD) comes the XLS declaration that will link the XML document to its XLS-stylesheet: <xsl:stylesheet> or <xsl:transform> (both can be used).

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

OR

<xsl:transform version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Styling a XML document with XSL

For the demonstration, here is the XML document saved with the xml extension that will be used :

```
<?xml version="1.0" encoding="UTF-8"?>                    // DTD
<?xml-stylesheet type="text/xsl" href="stylesheet.xsl"?>  // Link to XSL stylesheet

<bookstore>                                               // Root
        <book>
                <title>Poèmes Français</title>
                <author>Réjean Thomas</author>
                <year>2006</year>
                <price>14.99</price>
        </book>
        <book>
                <title>Harry Potter</title>
                <author>J K. Rowling</author>
                <year>2005</year>
                <price>29.99</price>
        </book>
        <book>
                <title>Le petit prince</title>
                <author>Antoine de Saint-Exupéry</author>
                <year>1943</year>
                <price>19.99</price>
        </book>
        <book>
                <title>Learning XML</title>
                <author>Erik T. Ray</author>
                <year>2003</year>
                <price>39.95</price>
        </book>
        <book>
                <title>L'avalée des avalés</title>
                <author>Réjean Ducharme</author>
                <year>1966</year>
                <price>39.95</price>
        </book>
        <book>
                <title>Les fleurs du mal</title>
                <author>Charles Baudelaire</author>
                <year>1857</year>
                <price>49.95</price>
        </book>
</bookstore>
```

Creating the XSL Style sheet

There are two parts in the stylesheet. The first part consists in the style itself, just like CSS, and the second part is the template which will make it possible, just like a loop would allow it, to display every elements of the XML document based on a predefined structure.

The file's structure

First, it is needed to save a document using the XLS extension in which the document type (DTD), the XLS, and the template match will be declared :

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">

<html>
<head>
<style>
        Your styles here
</style>
</head>
<body>
        Your template here
</body>
</html>

</xsl:template>
</xsl:stylesheet>
```

The style

Between <style> </style>, use the same CSS styling as usual. But start with establishing the template first, then use CSS to refine the presentation.

The template

```
<h2>Booklist</h2>
<table>
        <tr>
                <th>Title</th>
                <th>Author</th>
                <th>Year</th>
                <th>Price</th>
        </tr>

        <xsl:for-each select="bookstore/book">   // Loops the database to show all elements
        <xsl:sort select="year"/>                // Sorts the results based on a value
        <tr>
                <td><xsl:value-of select="title"/></td>
                <td><xsl:value-of select="author"/></td>
                <td><xsl:value-of select="year"/></td>
                <td><xsl:value-of select="price"/></td>
        </tr>
        </xsl:for-each>
</table>
```

In the above template, A table is used to structure the data. ***<th>*** is used for the headings and **<td>** for the data itself.

Position before and after *<tr> </tr>* in order to create a new line for each entry, ***xsl:for-each*** uses the *book* element to display the sub-elements (title, author, year and price) of all the book elements contained in the XML document.

xsl:sort (optional), sorts the results to display based of the selected value.

Filtering output

In order to filter the results, we could have specified the select value of *xsl:for-each*:

```
<xsl:for-each select="bookstore/book[year='2005']">
```

Note

Other XSLT options are available such as *if*, *choose*, and *apply*. But, although XSLT works fine, some browsers don't support it. Therefore, a more versatile solution consists into using JavaScript for all the transformations.