XSL Formatting Objects exercises

Sebastian Rahtz
Date: (revised 18/07/2002)

The aim of this exercise is to familiarize yourself with transforming TEI XML documents to XSL Formatting Objects and displaying the result. To prepare:

1. Open a web browser and visit http://www.tei-c.org.uk/Talks/OUCS/2003-02/samples.zip; save the zip file to disk and unpack it in your working directory (ie H:\).
2. Visit http://users.ox.ac.uk/~rahtz/antenna.exe and open the program directly; this will run a setup program for the Antenna House XSL formatter. Just keep clicking on defaults until it is installed. **Please note that the Antenna House Formatter is an evaluation copy of a commercial product! Contact the company, at http://www.antennahouse.com, if you want to use it outside this workshop!**.

In the `samples` directory, you will find two files `punch.xml` (the familiar page from Punch), and `punch-print.xsl` (an XSLT stylesheet to format it to XSL formatting objects). The file `punch.pdf` is a rendering to PDF by an XSL FO engine. You can create a FO file (which is just XML, if you look at it) with the command

```
xsltproc -o punch.fo punch-print.xsl punch.xml
```

You can then load `punch.fo` into the Antenna House XSL Formatter (icon on your desktop), and see the effect. Load the `.fo` file using the `[File]/[Open]` meny (do *not* specify a style file) and choose 'Run formatter'. If you get errors, choose Formatter Options/File output, and check 'Error logging' to have them written to file instead of popping up. The stylesheet is calling a set of generalized XSLT stylesheets for rendering TEI documents to Formatting Objects. You can alter the effects by editing `punch-print.xsl`, which has a series of high-level variables. Many of these are irrelevant to something as simple as this document, but we suggest you try these changes:

- Replace

  ```
  <xsl:variable name="bodyFont">Times Roman</xsl:variable>
  ```

  with

  ```
  <xsl:variable name="bodyFont">Helvetica</xsl:variable>
  ```

  to change the main font
- Replace

  ```
  <xsl:variable name="bodyMaster">10</xsl:variable>
  ```

  with

  ```
  <xsl:variable name="bodyMaster">14</xsl:variable>
  ```

  to change the main font size
- Replace

  ```
  <xsl:variable name="numberHeadings">true</xsl:variable>
  ```

with

```
<xsl:variable name="numberHEadings"></xsl:variable>
```

to stop sections being numbered.

Now undertake a similar process with `dickens.xml` (Dickens' *A Christmas Carol*) and it's corresponding `dickens-print.xsl`. Look at the how page breaks are represented; you can change this by setting

```
<xsl:variable name="activePagebreaks"></xsl:variable>
```

to

```
<xsl:variable
     name="activePagebreaks">true</xsl:variable>
```

Can you see what it has done?

The `<soCalled>` elements in Dickens are rendering with this template:

```
<xsl:template match="soCalled">
 <xsl:text>‘</xsl:text><xsl:apply-templates/><xsl:text>’</xsl:text>
</xsl:template>
```

which just puts quotes around the text. If you add a new template to `dickens-print.xsl` as follows:

```
<xsl:template match="soCalled">
  <fo:inline color="red" text-decoration="underline">
    <xsl:apply-templates/>
  </fo:inline>
</xsl:template>
```

you should see a more visible rendition.

As a simple exercise, can you render all the `<emph>` elements with underlining?