XSLT Exercises

Sebastian Rahtz

January 2001

1. Our practice data

All our exercises will be based on formatting an XML file which describes gravestones in a cemetery, the people commemorated on them, and the inscriptions. It will be helpful to first have a look at the data file on your disk in `H:\XML\CEMSIMPLE.XML`, using Emacs or Wordpad. Do not try to open it in XMetal, as this file has no DTD (you can look at `H:\XML\DATA.XML` if you want a version with a DTD).

In this data set the root element is `<cemetery>` and under that are a series of 23 `<stone>` elements. Within `<stone>` you find one or more `<person>` elements, some `<inscrip>` elements, and then probably `<desc>`, `<photo>` and `<icon>` elements. Thus the entire record for gravestone number 3 looks like this:

```
<stone number="3" zona="z_A" plot="0" height="42" width="50"
   form="PLW" material="ST" cond="c_1" bits="1" faces="E">

<person sex="f">
  <name><fnm>Hilda</fnm><snm>Munthe</snm></name>
  <born><date><day>2</day><mon>10</mon><yr>1882</yr></date></born>
   <died><date><day>28</day><mon>9</mon><yr>1967</yr></date></died>
   <pb><cty>Biarritz</cty><cny>France</cny></pb>
   <pd><cty>Rome</cty><cny>Italy</cny></pd><cause></cause>
   <age status="1">84</age><nat status="1" idref="SW"/>
   <profess></profess>
</person>

<person sex="m">
  <name><fnm>John Axel Viking</fnm><snm>Munthe</snm></name>
  <born><date><day> 3</day><mon> 4</mon><yr>1908</yr></date></born>
  <died><date><day> 8</day><mon>12</mon><yr>1976</yr></date></died>
  <pb><cty>London </cty><cny>England</cny></pb>
  <pd><cty>London </cty><cny>England</cny></pd><cause></cause>
  <age status="1">68 </age><nat status="1" idref="SW"/>
  <profess></profess>
</person>

<inscrip  face="f_E" cond="c_1" manner="m_IF" type="t_P" lang="l_EN">
 <l p="c">IN MEMORIAM</l>
 <l p="c"><ntag>AXEL</ntag> <ntag>MUNTHE</ntag></l>
 <l p="c"><ptag>OSKARSHAMN</ptag> OCTOBER 31, 1857</l>
 <l p="c"><ptag>STOCKHOLM</ptag> FEBRUARY 11, 1949</l>
</inscrip>

<inscrip  face="f_E" cond="c_1" manner="m_IF" type="t_S" lang="l_EN">
 <l p="c"><ntag>HILDA</ntag> <ntag>MUNTHE</ntag></l>
 <l p="c">NEE <ntag>PENNINGTON</ntag>-MELLOR</l>
 <l p="c"><ptag>BIARRITZ</ptag> OCTOBER 2, 1882</l>
 <l p="c"><ptag>ROME</ptag> SEPTEMBER 28, 1967</l></inscrip>
 <inscrip  face="f_E" cond="c_1" manner="m_IF" type="t_S" lang="l_EN">
 <l p="c"><ntag>JOHN</ntag> <ntag>AXEL</ntag> <ntag>VIKING</ntag> <ntag>MUNTHE</ntag></l>

 <l p="c"><ptag>LONDON</ptag> APRIL 3, 1908</l>
 <l p="c"><ptag>LONDON</ptag> DECEMBER 8, 1976</l>
```

```
</inscrip>
<desc>(vol 1, plate 2)</desc>
<photo film="*H" neg="26291"></photo>
</stone>
```

You can ignore a lot of the markup and attributes for the purposes of these exercises. What we want to achieve is a transformation of this file to HTML so that we can view it on the Web.

2. First look at using XSLT

We will first see how the tools work. Firstly, a command-line XSLT processor, making an HTML file.

1. Open an MSDOS box and change directory to `H:\XML`

   ```
   h:
   cd XML
   ```

2. Run the Saxon processor, to make `ex1.html` from `cemsimple.xml` using `ex1.xsl` as your stylesheet:

   ```
   saxon -o ex1.html cemsimple.xml ex1.xsl
   ```

3. Load ex1.html into your Web browser and see what the result looks like. Internet Explorer and Netscape are available - make sure you select the IE5 icon on the desktop to get the most recent version.
4. Load `ex1.xsl` into your editor (Emacs, XED or Wordpad; you cannot use XMetal for this, as the stylesheet has no DTD), and add a new template at the end before `</xsl:stylesheet>` as follows:

   ```
   <xsl:template match="stone">
     <h1>Stone <xsl:value-of select="@number"/></h1>
     <xsl:apply-templates/>
   </xsl:template>
   ```

   What this does is to specify some output for `<stone>` elements, making an HTML top-level heading, with the stone number in the title.
5. Save the XSL file, re-run Saxon, and look at the result again in your Web browser.

Now let us see the transformation happening in the web browser itself. For this you *must* use Internet Explorer 5, in the beta version we have installed on your computer.

1. You have a file called `ex1.xml`; open this in your editor and you will see a line

   ```
   <?xml-stylesheet type="text/xsl" href="ex1.xsl"?>
   ```

   This is an instruction to the browser to process the file using `ex1.xsl`
2. Open `ex1.xml` in IE5. You should see the same display as you got with the `ex1.html` you made earlier.
3. Edit `ex1.xsl` a little, and reload `ex1.xml` in the browser. Did your changes take effect?
4. Use the `View Source` option in the browser, and compare `ex1.xml` with `ex1.html`
5. Take out the `xml-stylesheet` line from `ex1.xml` and reload it. You should see the default behaviour of IE5 when confronted with an XML file.

You should now be confident that you can control the display of an XML file using the XSLT stylesheet. Now we can move on to refine the stylesheet.

3. Developing the stylesheet

Below are a series of suggestions for ways in which you can enhance the stylesheet. We will give worked solutions for the first two, then leave you on your own to try the rest.

You can use either the Saxon command-line method, or Internet Explorer, to do this work. While the browser route is more fun, be warned that you do not get very much feedback if you make a mistake in the XSL. And also remember back home that you must update your IE5 with the latest release of the XML and XSL engines (on the CD, of course).

First, let us see how to put people's names in bold, and separate surname and forename.

1. Open the stylesheet, and add a template which applies to the `<name>` element:

```
<xsl:template match="name">
  <b><xsl:apply-templates/></b>
</xsl:template>
```

2. Now can we put a space between forename and surname? The simplest way is to provide a template for `<snm>` which inserts a space before its content:

```
<xsl:template match="snm">
  <xsl:text> </xsl:text><xsl:apply-templates/>
</xsl:template>
```

3. Now apply the stylesheet, or reload the data file, and confirm that it works as you expected.

The second worked exercise is to make the inscriptions look nice.

1. First, a template to wrap each inscription in a block quote

```
<xsl:template match="inscrip">
  <blockquote><xsl:apply-templates/></blockquote>
</xsl:template>
```

2. Now, put a linebreak after each line of the inscriptions. Note the use of `<br/>` instead of just `<br>`; this is necessary to make it valid XML, and the XSLT processors know about it.

```
<xsl:template match="l">
  <xsl:apply-templates/><br/>
</xsl:template>
```

3. Finally, lets put anything in an `<i>` element into italics:

```
<xsl:template match="i">
  <i><xsl:apply-templates/></i>
</xsl:template>
```

Everything going smoothly? OK, now you are on your own to see if you can add some more features to our display. Sample solutions are on your CD-ROM in the `xslexercises` directory.

1. Omit everything from the `<person>` except the contents of the `<name>` (hint: use `<xsl:apply-templates select="name"/>`), and put each person in a separate paragraph
2. Omit the `<deco>` and `<photo>` elements, and format the `<died>` element nicely after each person's name, with / between day and month, and month and year
3. Improve the formatting or information display in whatever way interests you; can you, for instance, centre the lines which have a with a 'p' attribute with a value of 'c'?

4. Starting a new XSLT stylesheet

Remember the cookery book we looked at earlier? Can you construct an XSLT specification to render that to HTML?

5. Advanced Exercises

If you are feeling confident, tackle some more complex tasks. If you prefer, you can just study the solutions on the CD-ROM.

Now that we have the gravestones looking quite respectable, let us consider how we can present them in different ways. Most of these exercises are much simpler than they may seem at first sight!

1. Produce a table of contents for the catalogue as we left it in the final exercise of part 1; give each heading an identifier (the stone number will do, prefixed by "S"), and produce a list of numbers at the start, linked to the right stone.
   Sample solution: `ex10.xsl`, sample output `ex10.html`
2. Produce the catalogue sorted in order of surname and forename of people commemorated
   Sample solution: `ex11.xsl`, sample output `ex11.html`
3. Produce the ordinary catalogue, but append an index of surnames (sorted in alphabetical order), where each name is an HTML links to the right stone number
   Sample solution: `ex12.xsl`, sample output `ex12.html`
4. Produce a summary table listing the stone number, the number of people commemorated, and the number of lines of inscription
   Sample solution: `ex13.xsl`, sample output `ex13.html`

Now it starts getting a little less obvious:

1. Do the same catalogue that we produced earlier, but show only the *first* person and the *first* inscription. If there are other people or inscriptions, put in a paragraph saying eg 'there are *n* other people commemorated'. When you come to print an inscription, *number* the lines.
   Sample solution: `ex20.xsl`, sample output `ex20.html`
2. Make an exact copy of the input file, but put the stones in order of the year of death of the first person commemorated
   Sample solution: `ex21.xsl`, sample output `ex21.xml`
3. Using the file from the previous exercise, produce a catalogue of years in which people died. The year should be a `<h2>` heading, and should be followed by a numbered, list of the people (just surname and forenames) who died in that year, sorted by surname.
   Sample solution: `ex22.xsl`, sample output `ex22.html`
4. Can you do that last exercise *without* an intermediate sorted file?
   Sample solution: `ex23.xsl`, sample output `ex23.html`