XML Practical Assignment

T4 and TF, Spring 2019

To be submitted by 3rd April 2019

WARNING: you should use the FireFox browser for viewing XML files with DTD and XSLT links; recent versions of Chrome and Safari contain a bug (or security feature, depends who you believe!) that won't allow the importing of XSLT links from the same location from which the XML was served.

# Basic XML, DTD, and XML-HTML processing

Aim: Load a simple "people" file, check the relationships between the data structure and a DTD, and see how a simple XSLT processing instruction can display it in a browser.

Go to the FireFox browser and look at this file:

<http://www.cs.stir.ac.uk/~rcc/xml_egs/people/people.html>

then compare it with this one:

<http://www.cs.stir.ac.uk/~rcc/xml_egs/people/people.xml>

All being well, they will appear to be identical. Now go to "developer tools" in the browser, or otherwise load the xml source, along with the linked DTD and XLST files. Make sure you understand exactly what is going on to cause the display you see.

If you copy the files into your local filespace (including the xslt one linked to from the XML!) and change the internal references appropriately you should be able to view them on your own computer.

# XLST and namespaces

Download the XML file

<http://www.cs.stir.ac.uk/~rcc/xml_egs/mergedPeople/people2.xml>

and make a copy. You will see that this file derives from the original example used in lectures, where two different data sources have been integrated. It is quite easy to do this simply, but now we have a document that is difficult to query as there are different metadata uses. The best way to control this is via namespaces.

1. Write an XSLT document which will display a list of the names in this document. This can be done without namespaces.
2. Add the two processing instructions to the xml file which will cause a browser to apply the XSLT to the XML, and open the XML in a browser (FireFox!) to view it. You will find, probably, this is harder than you expected!
   1. Hint 1: try the people.xsl file from above as a starting point; that will do something although not what you want!
   2. Hint 2: if an <xsl:..> element doesn't match anything, it just has no output, so you can put in multiple elements to match different input patterns.
3. Add xmlns attributes to the XML file so that the different person data types are distinguished. You should also ensure that the hamster's name is never confused with a person's name. If you do this properly, it should stop the previous XSLT from working, so …
4. Change the XLST to that it works again, by adding namespaces to the xsl elements.

(You should submit all four files so that each XML file can be opened in a browser.)

# Using SAX for more serious applications

Here we will use SAX to do some processing on some larger data instances in XML, including the use of namespaces.

Look at the Java SAX code here

<http://www.cs.stir.ac.uk/~rcc/xml_egs/sax/nameReader.txt>

and understand what it should do; install it in your own environment and check that it executes correctly.

Look at the example of a newsML article here:

<http://www.cs.stir.ac.uk/~rcc/xml_egs/news/2286newsML.xml>

From this specific example, write a (partial) DTD description that would allow a programmer to understand how to navigate to the byline field. Then, from this description, write a Java SAX program that prints out the byline of this file. Note that this XML is not namespace-aware, and so neither should your SAX program be.

Now write a program that will work over a large document containing many such articles, and will print to the output the number of articles written by the author Henry Tricks. You will find such a large document in the file reutersAll.xml in the same directory.

# Write a short report describing what you have done.

"Short" means about two pages of text using normal formatting, ie similar to this document. It should not include code, but it might well contain individual lines or constructs where these are important for your text.

"What you have done" is not to be taken too literally, but should explain how you have used the various standards to produce the desired results at a high level. This would probably include a little, but not a textbook-length explanation, of what each standard does.

You should submit, all in a zipped folder:

All your XML and XSLT code

Your Java source code, and a screenshot of its output

Your report, as a PDF file.

This should be submitted electronically via Canvas by the end of 3rd April 2019.