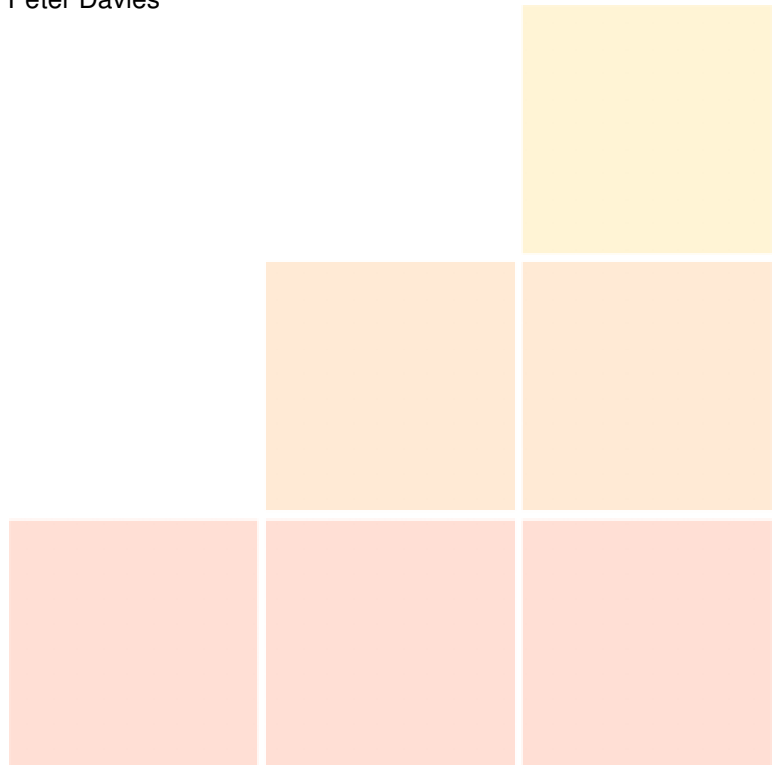


XML Development 2002-3

Assignment 2: XSL Transformations

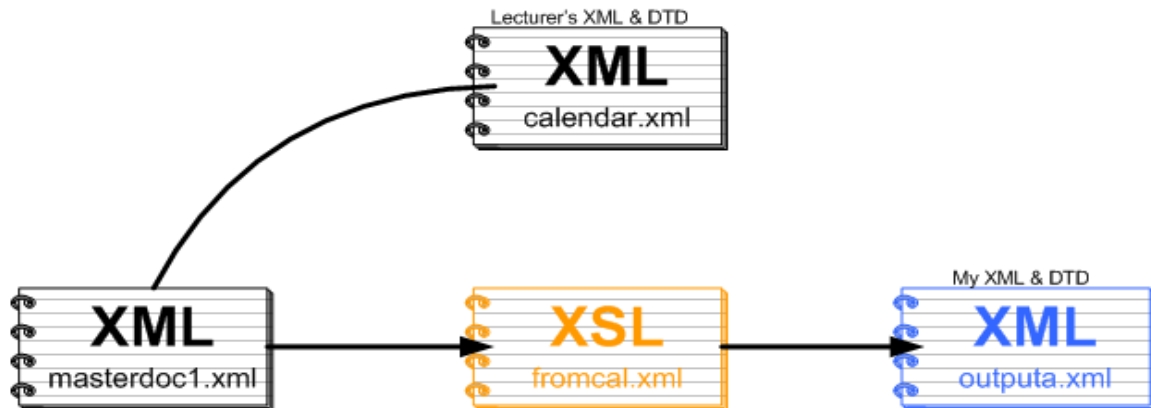
By Peter Davies



Assignment Part 1: XML Transforms

Transformation 1 - Overview

The required transformation was to take the XML file encoded with lecturer's DTD and transform it into a XML file that conformed to the DTD I developed in assignment 1.



The master XML document has an embedded DTD which is very simple and describes the file(s) that could be processed. The stylesheet then matches the Calendar element and loops through each of the Tasks, and the Events processing the data and inserting it into the format of the Agenda XML format and DTD (the file is Agendat.xml).

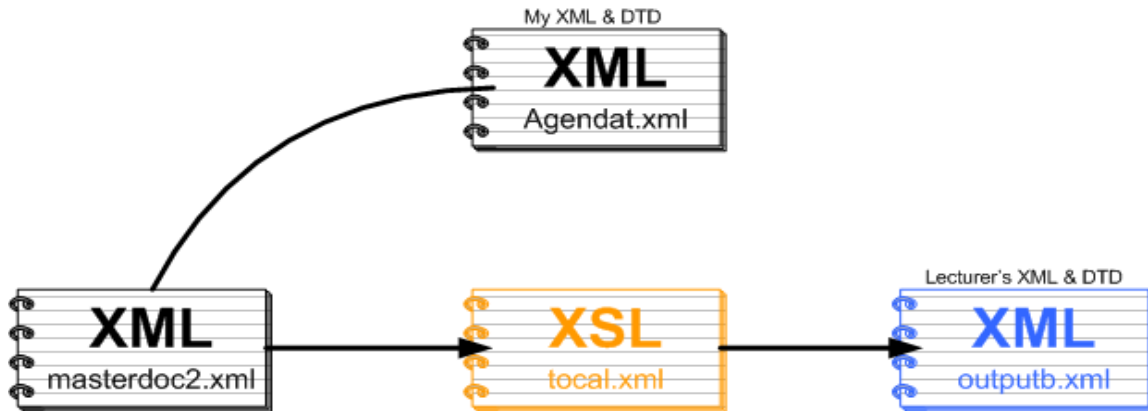
The output created fills the document with all of the tasks and events from the lecturer's file but inserting the Events into the Notes field. This was because the data stored in the Notes element resembled the structure of the data stored in the Calendar Events element.

The Agenda XML document was constructed with projects with identifiers so that a user could link a ToDo with a specific project. The XML output for this transition reflects this by converting all events and tasks into their specific project:

```
<Project ProjectID="pda10">XML Development Workshop</Project>  
<Project ProjectID="pda11">Kelly's Birthday</Project>  
<Project ProjectID="pda13">Web-based Email Demo</Project>
```

Transformation 2 - Overview

The required transformation was to take the XML file encoded with my DTD and transform it into a XML file that conformed to the DTD I developed by the Lecturer.



Due to the Lecturer's DTD being that much different from that of the one I developed as my first XML document it meant that my DTD did not contain much information that could fill the translated document.

Also due to time constraints the original Agendat XML document did not contain much test data. This was solved by using the round-trip data generated from the first transformation.

On creating the Calendar data the "modified" attribute was set to 'true' so that the output could be used on the 3rd part of this assignment:

```
<Event id="pda1" modified="true" deleted="false" duplicated="false">
```

Transformations – Data Loss

Transformation 1

The following bullet list shows the data that is not retained during the 1st transformation. Where data has not been retained in an Agenda element it is converted to text and stored in the Notes message element within the Agenda.

Events Element

- The specific time (hours and mins) – but is displayed in message element
- Start & End dates – but is displayed in message element
- Reminders
- Repeats

All task information is retained and transformed except the category element.

The resulting file was tested in a round-trip approach with the second part (1b) to this assignment. This resulting file was included in the code listings as "outputa rndtrip.xml"

Transformation 2

Due to the nature of the transformation it was possible to insert all of the data stored in my Agenda into that of the Lecturer's.

Assignment Part 2: Synchronisation

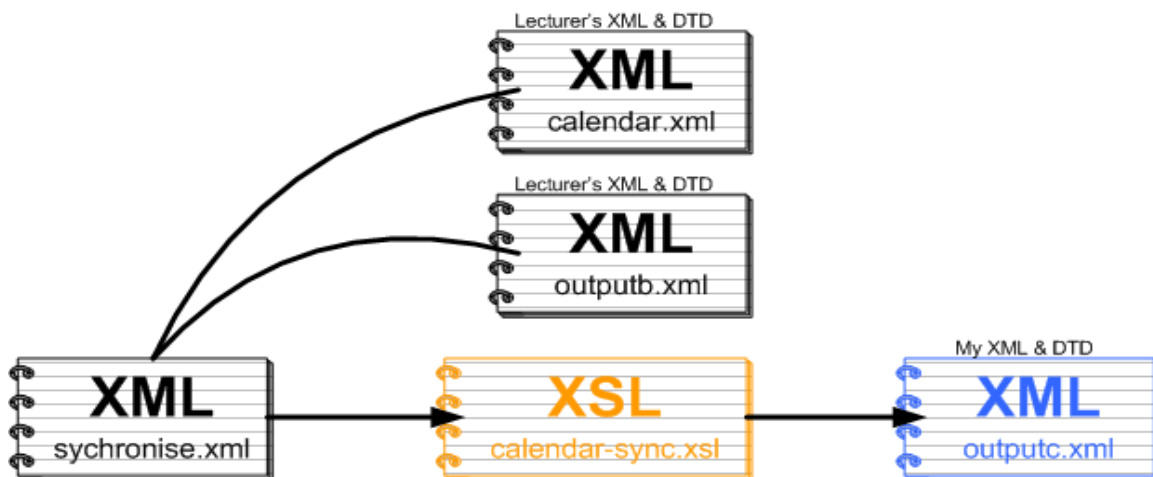
The Agenda XML file I developed for assignment 1 contained a Task, ToDo and Note which all included an ID identifier that is used for establishing a unique identifier for each of the created elements.

This approach means that the ID must be an XML name and therefore cannot start with a number. The adopted solution is as follows:

| | | |
|------|----------------------------------|---------------------|
| Task | <!ATTLIST Task kID ID #REQUIRED> | kID such as "k0001" |
| ToDo | <!ATTLIST ToDo tID ID #REQUIRED> | tID such as "t0001" |
| Note | <!ATTLIST Note nID ID #REQUIRED> | nID such as "n0001" |

This means that each inserted element will have a unique identifier so that during synchronisation a simple comparison can be made to identify new data. It would also be sensible to include a "modified" identifier on each element to determine whether the data has been changed – but on this first assignment I did not.

The assignment specification does not indicate that the two (master and slave) XML calendar files must conform to my Agenda DTD. It simply states that "two versions of the same calendar" must be combined. As my original specification did not cater for the added *created*, *updated* or *deleted* I decided that the best approach would be to use two of the lecturer's calendar files to do the comparison which would then create the required XML output conforming to the DTD I created for assignment 1 (see diagram).



Transformation Problems

Sync Version 1

I initially developed a simple procedure for extracting all of the tasks and events and then transforming them into a single XML document encoded with my DTD. This document was of course very simple and would require some more XSLT processing.

Sync Version 2

Using the previous version as a starting point I developed a series of ideas based around the idea of XSLT "lookup tables":

The following code was originally designed to loop through the events and tasks of the MASTER document and create a lookup entry in the root document:

```
<xsl:template match="/">
  <xsl:for-each select="/Sync/File">
    <xsl:for-each select="document(@filename)/Calendar/Event">
      <calid:name ids="{@id}">
        <xsl:value-of select="@id"/>
      </calid:name>
    </xsl:for-each>
    <xsl:for-each select="document(@filename)/Calendar/Task">
      <calid:name ids="{@id}">
        <xsl:value-of select="@id"/>
      </calid:name>
    </xsl:for-each>
  </xsl:for-each>
  <xsl:apply-templates select="Sync"/>
</xsl:template>
```

Using a for-each to loop through the events and tasks you could then do a simple check to see if when looping through the SLAVE file if the entry existed:

```
<xsl:for-each select="Event">
  <xsl:variable name="cid">
    <xsl:value-of select="current()/@id"/>
  </xsl:variable>
  <xsl:variable name="stateid">
    <xsl:value-of select="document('')/*/*calid:name[@ids=current()/@id]"/>
  </xsl:variable>
  <xsl:choose>
    <xsl:when test="$cid[.= $stateid]">
      <xsl:comment> Event id: <xsl:value-of select="$cid"/> -match</xsl:comment>
      <xsl:call-template name="calEvent">
        <xsl:with-param name="Event" select="current()"/>
      </xsl:call-template>
    </xsl:when>
  </xsl:choose>
</xsl:for-each>
```

This procedure did not work very well and resulted in blank variable \$stateid. This result is due to my lack of understanding of the structure of XSLT stylesheet documents.

Code Listing

Assignment 2: Part 1a

Contains: **masterdoc1.xml (embedded DTD)**
 fromcal.xsl
 outputa.xml
 outputa rndtrip.xml

Assignment 2: Part 1b

Contains: **masterdoc2.xml (embedded DTD)**
 tocal.xsl
 outputb.xml
 outputb rndtrip.xml

Assignment 2: Part 2

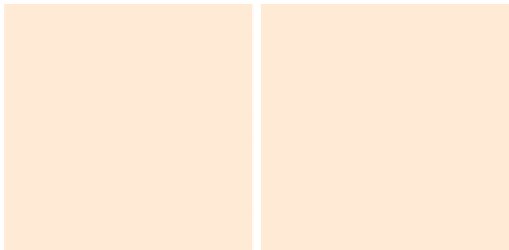
Contains: **synchronise.xml**
 calendar-sync.xsl
 calendar-sync.dtd
 outputb.xml (already printed)
 outputc.xml

Diskette

Contains all code and this report.

**Code Listing
Assignment 2: Part 1a**

**Code Listing
Assignment 2: Part 1b**



**Code Listing
Assignment 2: Part 2**