Computer Science 140

**Lab 8**

**Due** Week of Dec 3-6 during your lab period.

**Purpose**

Demonstrate basic understanding of XML and XSLT.

**Overview**

Create an XML file and an XSL transform file to format the XML data in a browser window.
If you are having difficulty making progress with this lab, confirm you have the symbols entered correctly and completely, then contact the instructor.

**Resources**

Notes on XML:
http://www.w3.org/XML
http://www.xml.com
Course web site notes on XML

**Prelab work**

Determine the answers to the following pre-lab questions before attempting the lab work. Answers are not handed in for marks. Refer to the XML online notes.

1. What is the purpose of XML?
2. What are beneficial features of XML?
3. What is the difference between XML and HTML?
4. What does the term 'well-formed' mean?
5. What is a 'broken' XML file?
6. What does the term 'valid' mean?
7. What is a DTD used for?
8. How can the DTD specify the order in which elements must be given?
9. Which two places can a DTD be declared?
10. What does the ELEMENT declaration do?
11. What does the plus sign signify after the element name?

12. What does the asterisk signify after the element name?

13. How do you specify which attribute values are needed for XML data?

14. What is XSL used for?

15. What is the purpose of an XML validator?

**Process**

1. Within your `public_html` folder on your deepblue account create a new folder named `lab8`. For this lab if you use the Microsoft Internet Explorer (IE) browser, you may have to close the browser and reopen it if you are seeing XML or XSLT errors.

2. From the course lab web page download the **ledger.txt** file into the lab8 folder. This file contains a set of nine sample sale transactions for the Pizza Palace restaurant. The amounts in the file are arbitrary. Rename the file to `ledger.xml`

3. Edit the `ledger.xml` file. Provide the first line required for an XML file. See your online XML course notes for the correct XML document declaration.

4. In the XML file define a root element named `<ledger>`. Add the required end tag for this root element.

5. Within the `<ledger>` root element, create the following new `<receipt>` element as the last element in the list of receipt elements.

| | |
|---|---|
| date: | 10/26/2013 |
| customer: | 136 |
| pizza: | cheese |
| quantity: | 2 |
| topping: | pineapple |
| amount: | 9.45 |

6. Validate the `ledger.xml` file by displaying it within the browser. If the browser displays parsing errors in the XML, for example missing an end tag or misspelling of an XML tag, the browser will indicate this information. Fix any parsing validation errors before continuing.

7. Create an internal DTD for your XML document. Recall that the DTD information appears just after the XML declaration. Since we must have at least one receipt in the `<ledger>` element, we place a plus sign next to the receipt element.

```
<!DOCTYPE ledger [
 <!ELEMENT ledger  (receipt+)>
 <!ELEMENT receipt (date, customer, pizza, quantity,
                    topping*, amount)>
 <!ELEMENT date      (#PCDATA)>
 <!ELEMENT customer  (#PCDATA)>
     . . .
```

Define the other <!ELEMENT for pizza, quantity, topping, amount (all use #PCDATA)

```
]>
```

← don't forget this notation to close the ledger element description

8. The next step is to modify the structure of the DTD. We want to define a set of valid attribute values for the topping element so that the only acceptable topping varieties are: green pepper, mushroom, pepperoni, cheese, pineapple (in any order). Any other variety used in the topping element will invalidate the XML. Why would you want to do this? To ensure that unexpected or incorrect data does not get used.

   In order to complete this requirement and make the XML file valid, you will have to add in a validation rule in the DTD section of the XML file for the topping attribute called "variety". Just below the last <!ELEMENT line, add the lines

```
<!ATTLIST topping
          variety ( green_pepper |
                    mushroom     |
                    pepperoni    |
                    cheese       |
                    pineapple )
          "cheese"
>
```

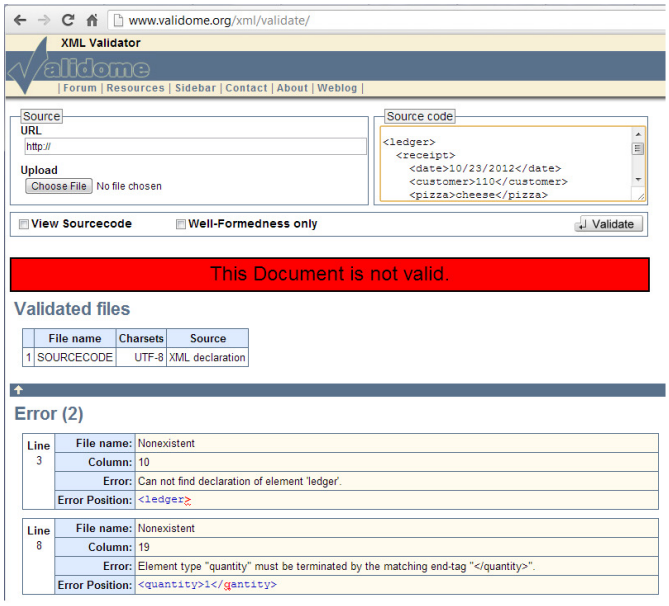
Use the vertical bar (pipe) to separate the options

   There should be line already containing just ]> following this ATTLIST element. This tells the DTD that you have an attribute named "variety" associated with the topping element and its only acceptable value is one of the above list of toppings. The final entry "cheese" is the default.

9. Point your browser to http://www.validome.org/xml/validate and validate your ledger.xml file (either as a local file , URL, or as text). If you receive a validation error or warning, you must fix your ledger.xml file. Confirm you have the symbols entered properly. XML does not care about the amount of white space between tokens, just so long as there is no space within the token. (e.g.

`<!ELEMENT date` is ok 

`< ! ELEMENT date)` is not ok 




← incorrect end tag found

10. Edit your `ledger.xml` file and, for a test, remove any line containing one of the elements inside a `<receipt>` element. Save the file and then revalidate the file to confirm the process catches your intentional XML error. Restore the `ledger.xml` file back to its valid form.

11. Create in the lab8 folder a new file called `ledger.xsl` file which will contain information about how we want to display your XML data in a browser. The first four lines of the `ledger.xsl` file are:

```
<?xml version='1.0'?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html"/>
<xsl:template match="/">
```

The second line indicates that this XML file is a stylesheet – the `xmlns` (xml namespace) attribute provides namespace information (the prefix is xsl).
The third line indicates that the stylesheet output is intended to be in HTML format.
The fourth line shows a template element. This element matches specific XML document nodes (elements) in the attribute match. Since the match is "/", we want to match all the elements in the xmls file.

12. The next part of the stylesheet file is the start of the HTML we want to display. The first HTML lines in the stylesheet file which set up a table are shown as follows. The CSS style for the `<tr>` element is the inline type.

```
<html>
   <body>
     <table border="2" cellpadding="4">
       <tr style="font-variant:small-caps;
                  font-weight:bold;
                  text-align:center;
                  background-color:#faa;">
         <td>  Date     </td>
         <td>  Customer </td>
         <td>  Topping  </td>
       </tr>
```

13. Following that HTML code we need to provide XSLT code to loop through each receipt element in our ledger.

```
<xsl:for-each select="ledger/receipt">
         <tr style="background-color:#fdd;">

           <td align="right">
             <xsl:value-of select="date"/>
           </td>

           <td>
             <xsl:value-of select="customer"/>
           </td>

           <td>
               <xsl:for-each select="topping">
                   <xsl:value-of select="@variety"/>  
               </xsl:for-each>
           </td>
         </tr>
</xsl:for-each>
```

This is extracting the values of the "date", "customer" and "topping" elements from the list of receipt elements. The `<xsl:` you see is the prefix that was defined in the second line from above.

The ` ` is an XML entity representing the non-breaking space. In XML you cannot use the usual set of HTML entities as ` ` or `&lt;` etc.

The `<xsl:for-each` is somewhat similar to a loop structure in which each receipt element is acted upon.

The `<xsl:value-of select` is a means in the xslt to retrieve the value of an element of that name.

This structure is building a row of lines in an HTML table.

14. End the stylesheet with the HTML end tags.

```
          </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

15. Save your `ledger.xsl` file. Open your `ledger.xml` file and add this as line 2:

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

This declaration tells the browser how to format this XML file using an associated xsl stylesheet file.

16. Open the `ledger.xml` file in your browser and you should see a table of receipt data containing just the date, customer and topping. You may alter the colour shading specified to something else if you prefer.

| Date | Customer | Topping | Pizza | Amount |
|---|---|---|---|---|
| 10/23/2012 | 110 | mushroom green_pepper | cheese | 12.23 |
| 10/24/2012 | 112 | | Hawaiian | 6.32 |
| 10/24/2012 | 152 | mushroom | pepperoni | 16.30 |
| 10/24/2012 | 122 | pineapple mushroom | cheese | 10.22 |
| 10/24/2012 | 130 | | Hawaiian | 26.60 |
| 10/25/2012 | 137 | | pepperoni | 12.64 |
| 10/25/2012 | 132 | mushroom | pepperoni | 8.32 |
| 10/25/2012 | 133 | green_pepper cheese mushroom | cheese | 11.22 |
| 10/25/2012 | 131 | pineapple | Hawaiian | 13.60 |
| 10/26/2012 | 136 | pineapple | cheese | 9.45 |
| 11/30/2012 | 300 | green_pepper mushroom | Hawaiian | 90.00 |

The issue regarding how browsers display empty table cells appears in this example if you are using the Internet Explorer browser.

The Internet Explorer browser will show the blank topping entries as cells having no borders. The Firefox browser shows blank topping entries as cells having borders. To make the Internet Explorer browser treat empty table cells properly, you can place a non-breaking space character (the HTML entity ` ` ) inside that empty table cell. XML does not permit use of HTML entities coded that way, so enter its decimal equivalent, ` ` between the tags `<td>` and

```
<xsl:for-each select="topping">
```

This will make the appearance of the table consistent for both browsers even if some of the table cells are empty.

17. Edit the `ledger.xsl` file so that your table shows the missing receipt information "pizza" and "amount" in the last two columns.

18. Validate this modified version of your `ledger.xml`.

19. XSLT provides running totals. After the definition of the `<td> </td>` element for the amount column, add the following lines of XSLT:

```
<xsl:variable name="total"
      select="sum(preceding-sibling::receipt/amount) + amount"/>
<td>
    <xsl:value-of  select="$total" />
</td>
```

This XSLT defines a variable named `total`, which will show the running total amounts in the rightmost column. Provide an appropriate column heading name for this new column. When you preview the table in the browser, some of the total values will show more than two decimal places. This amount can be formatted using XSLT's `format-number` function as in

```
select='format-number($total,"#,###.00")'
```

20. Modify one of the receipt element's `topping` value so that it will not validate to confirm the DTD will check the topping lists. Then, restore the XML file back to its valid state.

21. The last two columns would look better aligned right so that the decimal values all line up neatly. Add the required CSS `style` attribute to the two `<td>` tags to do this. (Hint: `text-align` property).

22. Complete the HTML display to show a heading (`<h2>`) containing appropriate text such as "Receipt Display by xx" with xx replaced by your name.

**Hand In**

1. Send me (langs@camosun.ca ) an email message with "Comp 140 Lab 8" as the email subject. Within the mail body, enter the deepblue URL to your `ledger.xml` file. You may include the answers in the email body as text or as an attachment. Do not send me MS Word documents.

   5 marks – proper display of the original xml data using xsl
   5 marks – display of the amount column

2. The answers for the following questions can be found doing this lab or by reviewing the XML class notes.

    a. What are two differences between the definition of an XML document and an XHTML document?
    b. What is the purpose of the DTD in XML?
    c. Answer True or False: All XML elements must contain at least one attribute.
    d. Answer True or False: XML elements can contain other, different elements.
    e. Can an XML data be well-formed but not valid? Explain.
    f. If the element `<topping>` is missing from a `<receipt>` element, given the lab DTD what will the value of `<topping> variety` attribute be set to?
    g. Do the XML elements within a `<receipt>` element need to be defined in the same order as specified by the DTD?
    h. What do you need to do in the XML file to connect to the XSL stylesheet?
    i. Which XSL element is used to process an XML document by looping instructions for a set of elements?
    j. In XSL how do you extract a specific attribute from an element in an XML document?