

XML – The Lingua Franca of the Information Age

Abigail F. Cantor, PE, MCS D

Biographical Sketch of Author

Abigail F. Cantor, owner of Process Research in Madison, Wisconsin, is a chemical engineer and a computer programmer.

Registered as a Professional Engineer, Ms. Cantor has been designing water and wastewater treatment processes since 1980. Since 1992, she has been mainly involved in investigations of corrosion in drinking water distribution systems.

In addition, Ms. Cantor is certified as a computer programmer by Microsoft. She develops applications to handle industrial, research, and business data. Her computer projects cover all steps of software development – planning the application's structures, designing the database, setting up the user interface, writing the code, and deploying the application.

Abstract

Just as the common language, Lingua Franca, united the world of trade and commerce in the Middle Ages, a new common language will unite incompatible computer systems to facilitate the exchange of data and information. That language is XML – Extensible Markup Language.

Anyone can make up XML tags and create an XML file in a text editor. These XML files can be opened in commercial XML-aware software such as spreadsheets and databases where the data can then be stored and manipulated. Data can also be saved as an XML file from the commercial software.

There are many advantages to communicating via XML.

- As an XML file, data takes up less computer memory and can be transferred to other computers faster than in other formats.
- Data can be transferred to other computers even if different operating systems or data storage software is used.
- Members of an organization or common interest group can declare standardized sets of XML tags to facilitate communication.
- The display of the XML data can be defined in an infinite number of ways using XSL files.

The full power of XML can be realized by manipulating it through a programming language. If a need arises to move data from one application to another that is outside the scope of commercial software, specialized software – both desktop computer programs and web applications – can be written to read and write XML files and to transfer the data as needed.

It is important to be aware of XML and its advantages. Whether you will be working with XML directly or whether you will be insulated from XML through computer software and web page interfaces, you will be communicating with the world more and more using XML. XML will soon be the Lingua Franca of the Information Age.

XML – The Lingua Franca of the Information Age

Abigail F. Cantor, PE, MCSD

Trade and communication were made possible around the Mediterranean Sea from the Middle Ages through the nineteenth century by the use of Lingua Franca, a common language known to all travelers and traders. In the modern Information Age, our trade in commerce and information easily circles the globe as we sit at our computers. Unfortunately, there has been no lingua franca for computers. As computers were developed over the twentieth century, incompatible operating systems and data storage software evolved. Now, an effort is being made to facilitate the exchange of data and information using a common language. That language is known as “XML.”

This presentation will introduce you to XML and show you how to start using XML immediately. Then, with some additional knowledge you can add even more power to your XML data files.

Understanding XML. XML is short for “extensible markup language.” As a markup language, XML is related to the familiar “hypertext markup language” or HTML used to create Internet web pages. Even with the existence of HTML to commonly display information, we still could not easily send data to others to use in spreadsheets, databases, or alternate displays. XML is the means to sending data and other information using syntax similar to HTML.

Both HTML and XML use tags to demarcate a text document into special divisions. A tag is enclosed in the brackets, < >. A text division has an opening tag (such as <P>) and a closing tag (such as </P>). A division of the text document may look like:

```
<P> Hello, everyone!</P>
```

The difference between XML and HTML is that the XML user can make up any tag desired and does not have to depend on a set of tags understood only by the particular computer browser where the information is displayed.

Figure 1 is an example of XML describing water samples. This is just one possibility for describing water samples with other possibilities limited only by the imagination of the user.

Reading an XML file. Suppose a colleague sent the XML of Figure 1 to you in a text file named, “WaterSample.xml.” It would be useful for viewing both the data and the data’s structure. To see this, write the XML of Figure 1 in a text editor such as Notepad. Name the file with an “xml” file extension, such as “WaterSample.xml”. Open the file in the browser application of the computer, such as Netscape or Internet Explorer. The browser (depending on its version) displays XML files by visually distinguishing between data and structure (partially shown in Figure 2).

Still, this data cannot be used automatically in a versatile manner. One simple addition to the XML text file in a text editor, like Notepad, can transform the file into a source of easily transferable and usable data. Add the following “header” line of text at the beginning of the file:

```
<?xml version="1.0" ?>
```

Now, the XML file can be opened in commercial XML-aware software such as Microsoft Excel 2002¹ spreadsheet or Microsoft Access 2002² database. Figure 3 shows the first columns from the WaterSample.xml file opened in Excel 2002.

Writing to an XML file. Now that we have a means to read and use XML data so easily, we need a means to easily write data as an XML file. An XML file is actually the simplest of file structures, a text file. As has been

shown, XML structure and data are created in text editors, such as Notepad. However, it can be quite tedious and time-consuming to create such a text file for a lot of data. To create an XML file of the data automatically, commercial XML-aware software is available. Using Excel 2002 as an example, create columns of data with headings in a spreadsheet (see Figure 4). Save the spreadsheet as an XML file, an option available in Excel 2002 (see Figure 5). This new XML file can be viewed in a computer browser or can be opened in an XML-aware commercial spreadsheet application.

Exchanging XML data. Even if data is sent to a computer that uses the same spreadsheet application and computer operating system as the sender's computer, there are advantages to sending the data as an XML file over a spreadsheet file. The Excel data shown in Figure 4 uses 14 kilobytes of memory whereas the same data represented in the XML file of Figure 5 uses 5 kilobytes. Therefore, as an XML file, the data is sent faster over the Internet and uses less memory for storage.

If a different operating system and different spreadsheet application is used by the recipient, the possibility exists that the data will be usable in the recipient's spreadsheet application and even be displayed in the same way as in the original spreadsheet. The display in any application is not definite because software applications are in various stages of recognizing XML at this time. However, the data will definitely be transferred to any operating system because it is in the form of a text file, which all operating systems can work with.

Organizing around XML. Notice that there is a major difference between the XML in Figure 1 and Figure 5. There are a number of extra XML tags listed in Figure 5 that refer to standardized tags meaningful in spreadsheet applications. Even non-Microsoft spreadsheet applications can be designed to understand the tags defined by the "XML Spreadsheet Schema," a special document that describes the tags to be used and their associated data characteristics.

This leads to an important realization about XML. That is, any person or group of people can create their own special set of XML tags to fit their needs. A number of organizations have created and accepted sets of XML tags for better communication among the organization's members. For instance, the Chemical Markup Language is a defined set of XML tags for chemists to communicate the structure of complex molecules.³ The Mathematical Markup Language is a set of XML tags for mathematicians and scientists to exchange documents containing mathematical formulae.³ The new markup languages created by any entity are typically described and standardized in either Document Type Definition documents or in XML Schema documents. These documents are beyond the scope of this presentation, but it is important to know that these documents are used for standardizing a set of XML tags.

Displaying XML data. Additional power can be added to an XML file by providing an accompanying XSL (extensible styles language) file. An XSL file describes how to display data within each tag used in the XML file. Best yet, different XSL files can be used to display the same XML data in a different fashion. For example, the XML data of Figure 1 can be displayed as shown in Figure 6 using the XSL file in Figure 7. The same data can be displayed as shown in Figure 8 using the XSL file in Figure 9. The possibilities for display are endless.

To test the display yourself, open the WaterSamples.xml file in Notepad. Replace the previous header line with:

```
<?xml version="1.0"?>  
<?xml-stylesheet type="text/xsl" href="watersamples1.xsl"?>
```

Save and close WaterSamples.xml. Now, create a new file in Notepad called "watersamples1.xsl". Write the text from Figure 7 into the file. Save and close. Open WaterSamples.xml in the browser.

To view the second version shown in Figure 8, open the WaterSamples.xml file in Notepad. Replace the previous header lines with:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="watersamples2.xsl"?>
```

Save and close WaterSamples.xml. Now, create a new file in Notepad called “watersamples2.xsl”. Write the text from Figure 9 into the file. Save and close. Open WaterSamples.xml in the browser.

Summary. Anyone can make up XML tags and create an XML file in a text editor. These XML files can be opened in commercial XML-aware software such as spreadsheets and databases where the data can then be stored and manipulated. Data can also be saved as an XML file from the commercial software.

There are many advantages to communicating via XML.

- As an XML file, data takes up less computer memory and can be transferred to other computers faster than in other formats.
- Data can be transferred to other computers even if different operating systems or data storage software is used.
- Members of an organization or common interest group can declare standardized sets of XML tags to facilitate communication.
- The display of the XML data can be defined in an infinite number of ways using XSL files.

The full power of XML can be realized by manipulating it through a programming language. If a need arises to move data from one application to another that is outside the scope of commercial software, specialized software – both desktop computer programs and web applications – can be written to read and write XML files and to transfer the data as needed.

It is important to be aware of XML and its advantages. Whether you will be working with XML directly or whether you will be insulated from XML through computer software and web page interfaces, you will be communicating with the world more and more using XML. XML will soon be the Lingua Franca of the Information Age.

References:

1. Microsoft Excel support for XML -- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnexcel2k2/html/odc_xlflatnr.asp
2. Microsoft Access support for XML -- <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/modcore/html/deconxmldatainport.asp>
3. Holzner, Steven. Inside XML. Indianapolis: New Riders Publishing, 2001. *(Note: There are many excellent books describing the use of XML. The reader may want to investigate others in order to learn more on the subject. The book, Inside XML, is very informative for use as a learning tool and as a reference book. It gives a non-Microsoft perspective of XML and covers the proper syntax used in the many aspects of XML: Well-Formed XML Documents, Document Type Definitions, XML Schemas, Cascading Style Sheets, and XSL Transformations and Formatting Objects. Interacting with XML using computer programming is also covered for readers who want to learn about the ultimate power of XML.)*

Figure 1. XML describing water samples

```

<watersamples>
  <watersample>
    <id>
      1000A
    </id>
    <alkalinity>
      <units>
        mg/L as CaCO3
      </units>
      <value>
        127
      </value>
    </alkalinity>
    <pH>
      <units>
        S.U.
      </units>
      <value>
        7.2
      </value>
    </pH>
    <sampler>
      <employeeeno>
        203
      </employeeeno>
      <firstname>
        Veronica
      </firstname>
      <lastname>
        Lake
      </lastname>
      <organization>
        Northern Wisconsin Watershed Association
      </organization>
    </sampler>
  </watersample>
  <watersample>
    <id>
      45
    </id>
    <alkalinity>
      <units>
        mg/L as CaCO3
      </units>
      <value>
        74
      </value>
    </alkalinity>
    <pH>
      <units>
        S.U.
      </units>
      <value>
        7.0
      </value>
    </pH>
    <sampler>
      <employeeeno>
        890
      </employeeeno>
      <firstname>
        Muddy
      </firstname>
    </sampler>
  </watersample>
</watersamples>

```

```

        <lastname>
            Waters
        </lastname>
        <organization>
            Louisiana Water Resources Agency
        </organization>
    </sampler>
</watersample>
<watersample>
    <id>
        1002
    </id>
    <alkalinity>
        <units>
            mg/L as CaCO3
        </units>
        <value>
            89
        </value>
    </alkalinity>
    <pH>
        <units>
            S.U.
        </units>
        <value>
            7.3
        </value>
    </pH>
    <sampler>
        <employeeeno>
            203
        </employeeeno>
        <firstname>
            Veronica
        </firstname>
        <lastname>
            Lake
        </lastname>
        <organization>
            Northern Wisconsin Watershed Association
        </organization>
    </sampler>
</watersample>
</watersamples>

```

Figure 2. XML viewed directly in a computer browser (partial view of file)

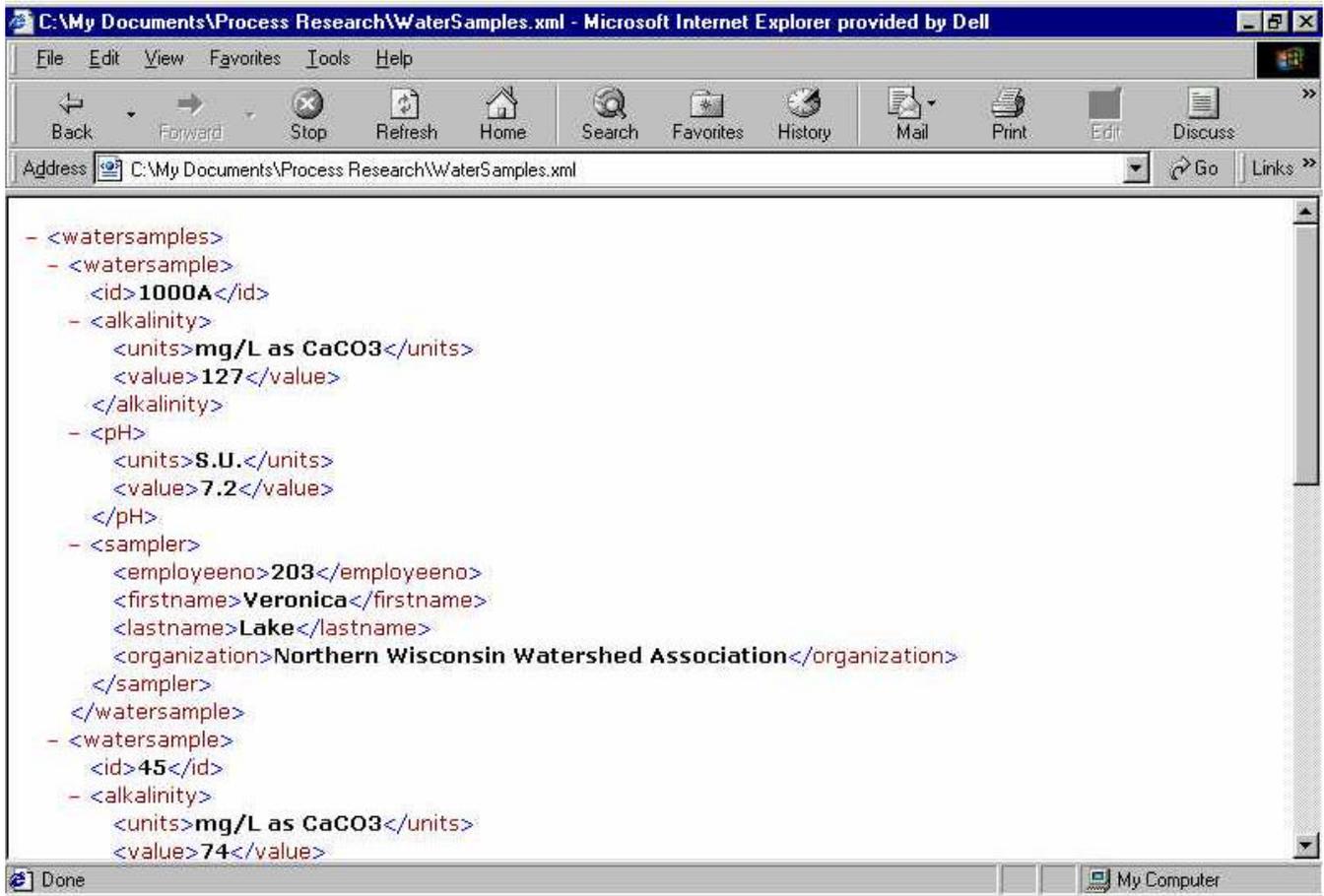


Figure 3. The first columns from the WaterSample.xml file opened in Excel 2002

1	/watersamples					
2	/watersample/alkalinity/units	/watersample/alkalinity/value	/watersample/alkalinity/value/@agg	/watersample/id	/watersample/pH/units	/watersample/
3	mg/L as CaCO3	127	127	1000A	S.U.	
4	mg/L as CaCO3	74	74	45	S.U.	
5	mg/L as CaCO3	89	89	1002	S.U.	
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						
29						
30						
31						
32						
33						
34						
35						

Figure 4. Columns of data with headings in an Excel 2002 spreadsheet

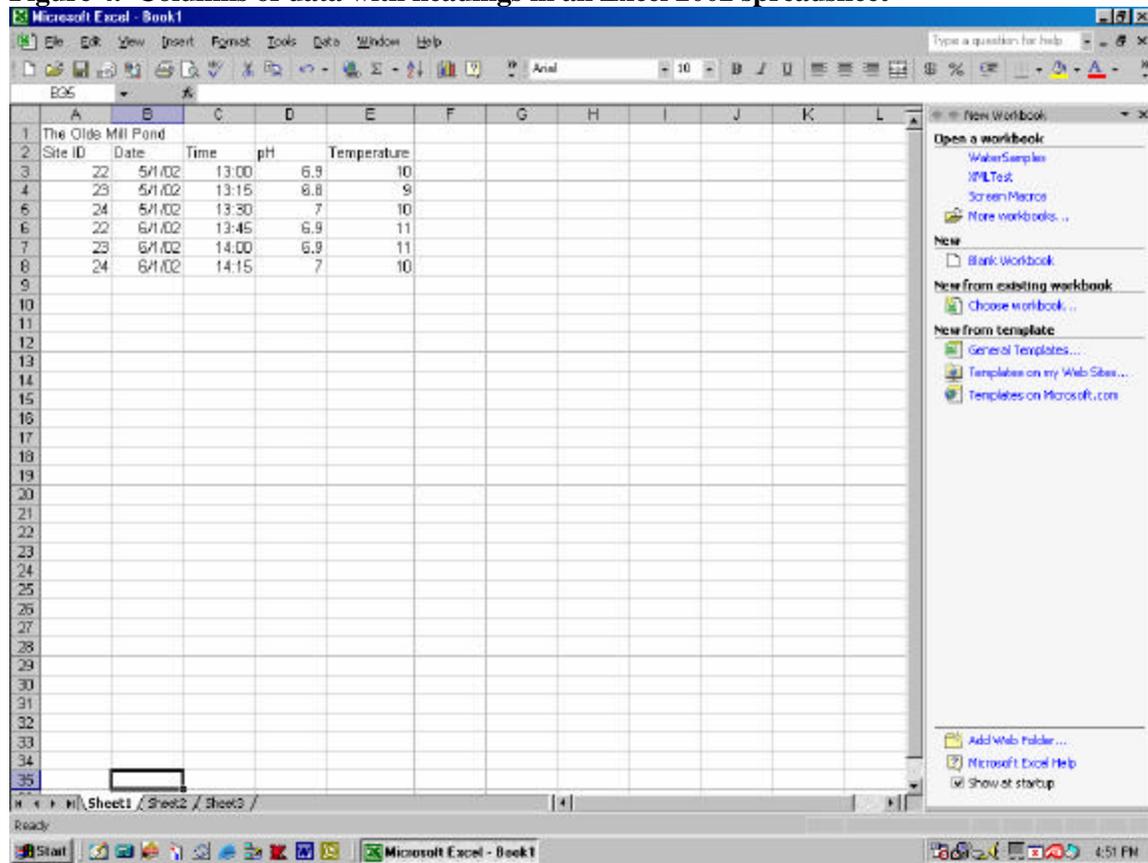


Figure 5. The Excel 2002 spreadsheet of Figure 4 saved as an XML file

```

<?xml version="1.0"?>
<Workbook xmlns="urn:schemas-microsoft-com:office:spreadsheet"
  xmlns:o="urn:schemas-microsoft-com:office:office"
  xmlns:x="urn:schemas-microsoft-com:office:excel"
  xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet"
  xmlns:html="http://www.w3.org/TR/REC-html40">
<DocumentProperties xmlns="urn:schemas-microsoft-com:office:office">
  <Author>Veronica Lake</Author>
  <LastAuthor>Veronica Lake</LastAuthor>
  <Created>2002-04-09T21:45:50Z</Created>
  <Company>Northern Wisconsin Watershed Association</Company>
  <Version>10.2625</Version>
</DocumentProperties>
<OfficeDocumentSettings xmlns="urn:schemas-microsoft-com:office:office">
  <DownloadComponents/>
  <LocationOfComponents HRef="file:///D:\"/>
</OfficeDocumentSettings>
<ExcelWorkbook xmlns="urn:schemas-microsoft-com:office:excel">
  <WindowHeight>9345</WindowHeight>
  <WindowWidth>11340</WindowWidth>
  <WindowTopX>480</WindowTopX>
  <WindowTopY>60</WindowTopY>
  <ProtectStructure>False</ProtectStructure>
  <ProtectWindows>False</ProtectWindows>
</ExcelWorkbook>
<Styles>
  <Style ss:ID="Default" ss:Name="Normal">
    <Alignment ss:Vertical="Bottom"/>
    <Borders/>
    <Font/>
    <Interior/>
    <NumberFormat/>
    <Protection/>
  </Style>
  <Style ss:ID="s21">
    <NumberFormat ss:Format="Short Date"/>
  </Style>
  <Style ss:ID="s22">
    <NumberFormat ss:Format="Short Time"/>
  </Style>
</Styles>
<Worksheet ss:Name="Sheet1">
  <Table ss:ExpandedColumnCount="5" ss:ExpandedRowCount="8" x:FullColumns="1"
  x:FullRows="1">
    <Column ss:Index="5" ss:Width="59.25"/>
    <Row>
      <Cell><Data ss:Type="String">The Olde Mill Pond</Data></Cell>
    </Row>
    <Row>
      <Cell><Data ss:Type="String">Site ID</Data></Cell>
      <Cell><Data ss:Type="String">Date</Data></Cell>
      <Cell><Data ss:Type="String">Time</Data></Cell>
      <Cell><Data ss:Type="String">pH</Data></Cell>
      <Cell><Data ss:Type="String">Temperature</Data></Cell>
    </Row>
    <Row>
      <Cell><Data ss:Type="Number">22</Data></Cell>
      <Cell ss:StyleID="s21"><Data ss:Type="DateTime">2002-05-01T00:00:00.000</Data></Cell>
      <Cell ss:StyleID="s22"><Data ss:Type="DateTime">1899-12-31T13:00:00.000</Data></Cell>
      <Cell><Data ss:Type="Number">6.9</Data></Cell>
      <Cell><Data ss:Type="Number">10</Data></Cell>
    </Row>
    <Row>

```

```

<Cell><Data ss:Type="Number">23</Data></Cell>
<Cell ss:StyleID="s21"><Data ss:Type="DateTime">2002-05-01T00:00:00.000</Data></Cell>
<Cell ss:StyleID="s22"><Data ss:Type="DateTime">1899-12-31T13:15:00.000</Data></Cell>
<Cell><Data ss:Type="Number">6.8</Data></Cell>
<Cell><Data ss:Type="Number">9</Data></Cell>
</Row>
<Row>
<Cell><Data ss:Type="Number">24</Data></Cell>
<Cell ss:StyleID="s21"><Data ss:Type="DateTime">2002-05-01T00:00:00.000</Data></Cell>
<Cell ss:StyleID="s22"><Data ss:Type="DateTime">1899-12-31T13:30:00.000</Data></Cell>
<Cell><Data ss:Type="Number">7</Data></Cell>
<Cell><Data ss:Type="Number">10</Data></Cell>
</Row>
<Row>
<Cell><Data ss:Type="Number">22</Data></Cell>
<Cell ss:StyleID="s21"><Data ss:Type="DateTime">2002-06-01T00:00:00.000</Data></Cell>
<Cell ss:StyleID="s22"><Data ss:Type="DateTime">1899-12-31T13:45:00.000</Data></Cell>
<Cell><Data ss:Type="Number">6.9</Data></Cell>
<Cell><Data ss:Type="Number">11</Data></Cell>
</Row>
<Row>
<Cell><Data ss:Type="Number">23</Data></Cell>
<Cell ss:StyleID="s21"><Data ss:Type="DateTime">2002-06-01T00:00:00.000</Data></Cell>
<Cell ss:StyleID="s22"><Data ss:Type="DateTime">1899-12-31T14:00:00.000</Data></Cell>
<Cell><Data ss:Type="Number">6.9</Data></Cell>
<Cell><Data ss:Type="Number">11</Data></Cell>
</Row>
<Row>
<Cell><Data ss:Type="Number">24</Data></Cell>
<Cell ss:StyleID="s21"><Data ss:Type="DateTime">2002-06-01T00:00:00.000</Data></Cell>
<Cell ss:StyleID="s22"><Data ss:Type="DateTime">1899-12-31T14:15:00.000</Data></Cell>
<Cell><Data ss:Type="Number">7</Data></Cell>
<Cell><Data ss:Type="Number">10</Data></Cell>
</Row>
</Table>
<WorksheetOptions xmlns="urn:schemas-microsoft-com:office:excel">
<Selected/>
<Panes>
<Pane>
<Number>3</Number>
<ActiveRow>34</ActiveRow>
<ActiveCol>1</ActiveCol>
</Pane>
</Panes>
<ProtectObjects>False</ProtectObjects>
<ProtectScenarios>False</ProtectScenarios>
</WorksheetOptions>
</Worksheet>
<Worksheet ss:Name="Sheet2">
<WorksheetOptions xmlns="urn:schemas-microsoft-com:office:excel">
<ProtectObjects>False</ProtectObjects>
<ProtectScenarios>False</ProtectScenarios>
</WorksheetOptions>
</Worksheet>
<Worksheet ss:Name="Sheet3">
<WorksheetOptions xmlns="urn:schemas-microsoft-com:office:excel">
<ProtectObjects>False</ProtectObjects>
<ProtectScenarios>False</ProtectScenarios>
</WorksheetOptions>
</Worksheet>
</Workbook>

```

Figure 6. XML data of Figure 1 transformed by the XSL file of Figure 7

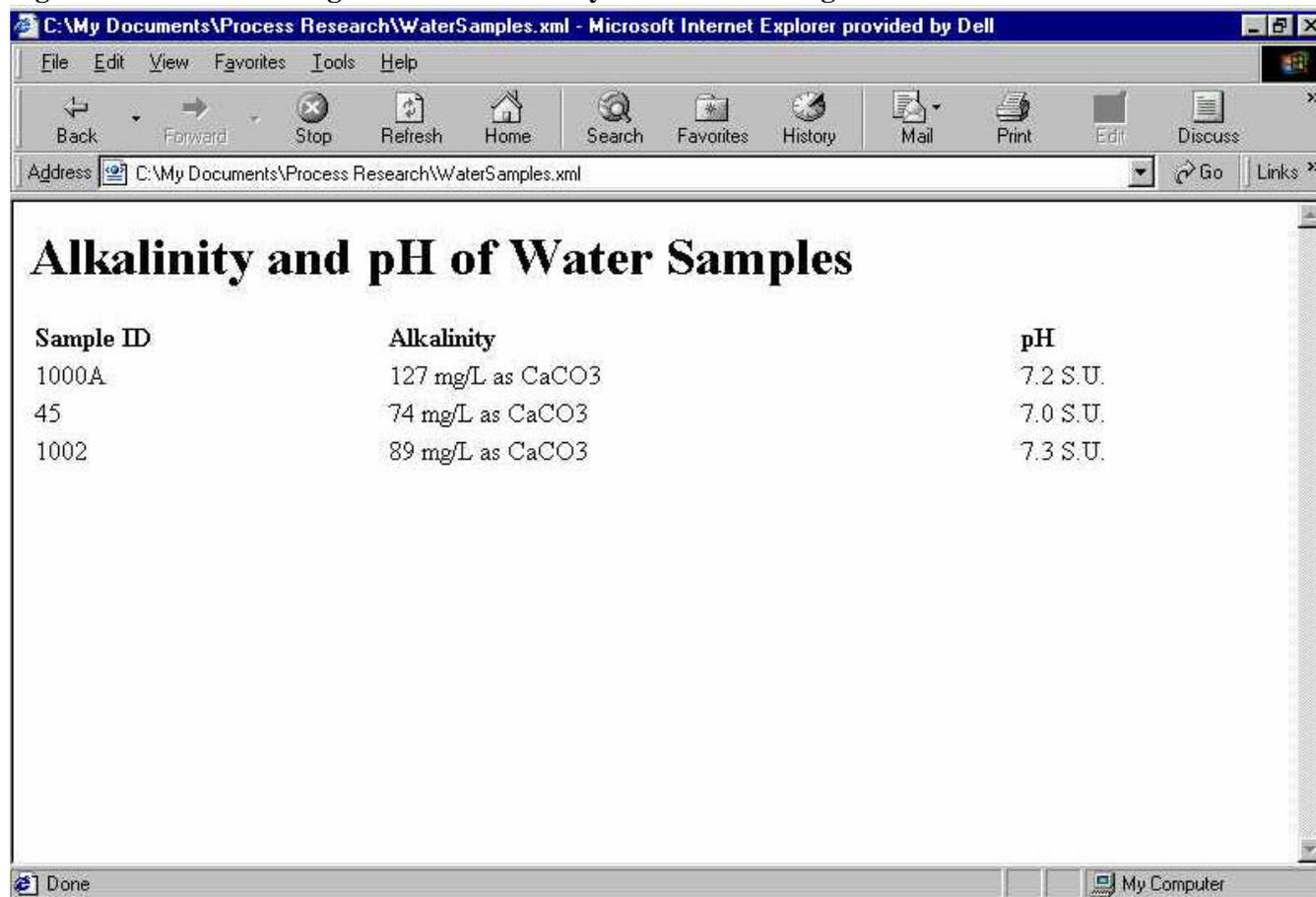


Figure 7. First XSL file transforming the XML data of Figure 1

```
<?xml version = "1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">

<xsl:template match="/">
<html><body>
  <h1>Alkalinity and pH of Water Samples</h1>
  <table width="100%">
    <tr>
      <td><b>Sample ID</b></td>
      <td><b>Alkalinity</b></td>
      <td><b>pH</b></td>
    </tr>
    <xsl:apply-templates/>
  </table>
</body></html>
</xsl:template>

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

<xsl:template match="watersample">
  <tr>
    <td><xsl:value-of select="id"/></td>
    <td><xsl:value-of select="alkalinity/value"/>
      <xsl:value-of select="alkalinity/units"/></td>
    <td><xsl:value-of select="pH/value"/>
      <xsl:value-of select="pH/units"/></td>
    </tr>
  </xsl:template>

</xsl:stylesheet>
```

Figure 8. XML data of Figure 1 transformed by the XSL file of Figure 9

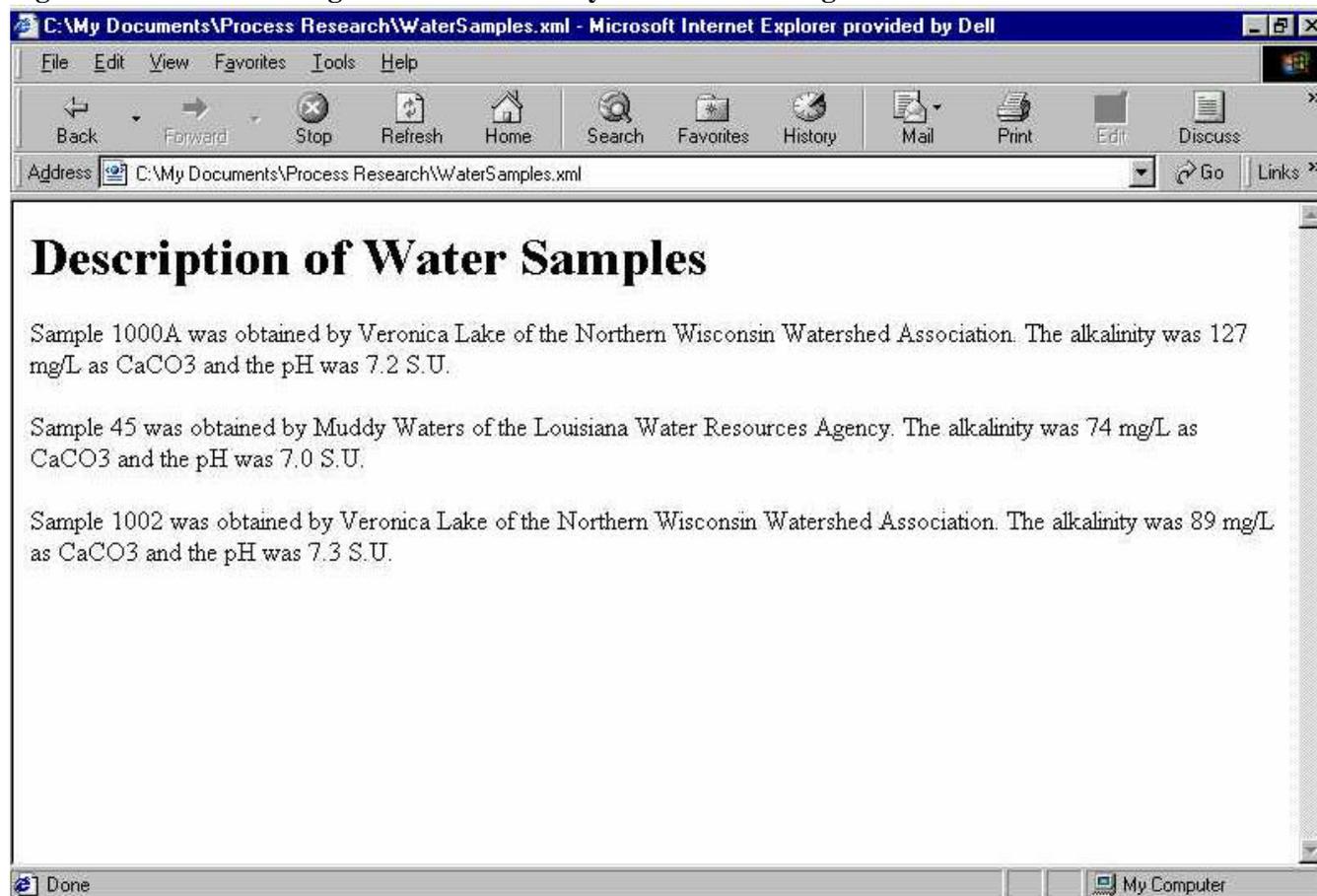


Figure 9. Second XSL file transforming the XML Data of Figure 1

```
<?xml version = "1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">

<xsl:template match="/">
<html><body>
                <h1>Description of Water Samples</h1>
                <xsl:apply-templates/>
</body></html>
</xsl:template>

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

<xsl:template match="watersample">
                <p>
                Sample <xsl:value-of select="id"/> was obtained by
                <xsl:value-of select="sampler/firstname"/>
                <xsl:value-of select="sampler/lastname"/>
                of the <xsl:value-of select="sampler/organization"/>.
                The alkalinity was
                <xsl:value-of select="alkalinity/value"/>
                <xsl:value-of select="alkalinity/units"/>
                and the pH was
                <xsl:value-of select="pH/value"/>
                <xsl:value-of select="pH/units"/>
                </p>
</xsl:template>

</xsl:stylesheet>
```