Using XML/XSLT as an Alternative to Microsoft Excel

Pradeep Padala
Computer & Information Science & Engineering
University of Florida
Gainesville, Florida 32611–6120
Email: ppadala@cise.ufl.edu

***Abstract*— In CISE[1], instructors usually maintain student records in Microsoft Excel. This causes serious interoperability and flexibility issues. Alternatives like storing the data in flat files or comma separated value (CSV) files require glue code to be written. XML/XSLT provide an exciting alternative to representing the data in a platform independent way. In this paper, we design XML documents for representing student records and XSL transformations to apply on the documents. We conclude by showing that XML/XSLT offer a distinct advantage over the alternatives.**

I. Introduction

XML[1] provides a convenient way of representing heterogeneous data. It provides interoperability and an easy of manipulating the data with tools like XSLT[2]. Microsoft Excel is a spreadsheet and analysis program that is used to store data like payrolls, student grades etc. Excel provides various ways to interact with the data using macros. But, Excel can only be run on windows and has serious interoperability problems with other office suites like OpenOffice.

At CISE, instructors maintain student records containing various information like project grades, exam grades and extra credit. Usually, they are stored in Excel causing interoperability problems with UNIX applications.

As a teaching assistant for CEN3031 (*Introduction to Software Engineering*), I have developed various methods to cope up with the platform dependence of Excel. In this paper, first, I explain the basic problem associated with using Excel to store data and basics of data representation. Then, the proposed solution for representing the data in XML is explained. Next, variations of the document and benefits of each representation are discussed. Finally, the XSL transformations applied to the XML document and resulting HTML output is shown.

II. The problem - data representation

In CEN3031, we have student records containing grades. The grades are entered by a TA and are shown to the students on on-line services. The on-line services serve the content using CGI (Common Gateway Interface).

Student records often contain simple fields like name, ssn, exam1 grade etc. Using Excel to store and manipulating the records is straight-forward and easy. But, Excel is dependent on windows platform. The data can also be represented in simple text or csv (comma separated value format) files. Due to simple nature of these files, they require glue code for manipulation. Though the glue code can be developed quickly using scripting languages like Perl, writing the code is still a significant overhead for course staff. On the other extreme, we have databases that can store information in a very structured way. Though this is the best choice for large data, for simple data like student grades it is an overkill.

XML with XSLT is the right choice for representing the data. XSLT can easily be applied to produce HTML output that can be served using CGI.

```
<course name="se">
    <student>
        blah blah blah
        blah blah blah
    </student>
</course>
```

Fig. 1. Sample listing

III. XML records

We can create a simple record structure with XML documents. Theoretically we don't need to have a DTD. Here I took liberty and created a simple XML document whose top level element is `<course>` and contains `<student>` nodes. Think of them as records. Figure 1 shows a sample of how a student record can be represented in XML. Figure 2 shows the listing of a complete XML document containing two student records.

This document contains two student records, whose contents can be included in any order. Meaning, I can have `<firstname> </firstname>` any where in the record. The fields need not be in order. This solves the problem of some one sorting the excel file and messing the master document's structure.

IV. Conversion to HTML

This section describes the way XSL transformations are used to produce HTML output. We can do variety of things using XSLT. The main concept is to separate the data and

This article is the result of a discussion with Dr. Cubert regarding using XML/XSLT for representing student records in CEN3031

[1]Computer & Information Science & Engineering

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

<course name="se">
    <student>
        <firstname>Kevin</firstname>
        <lastname>Ayers</lastname>
        <password>a1234xyz</password>
        <ssn>12345678</ssn>
        <class>7EG</class>
        <major>CEN</major>
        <phone>
            <areacode>352</areacode>
            <number>1234567</number>
        </phone>
        <email>abc@bbc.com</email>
        <username>kma</username>
        <exam name="exam1" max="100">
            <score>61</score>
        </exam>
        <exam name="exam2" max="70">
            <score>50</score>
        </exam>
        <ta name="ppadala"/>
        <sect name="6261"/>
    </student>

    <student>
        <firstname>John</firstname>
        <lastname>Buisson</lastname>
        <password>c1234abc</password>
        <ssn>12345678</ssn>
        <class>7EG</class>
        <phone>
            <areacode>352</areacode>
            <number>1234467</number>
        </phone>
        <major>CEN</major>
        <username>jb</username>
        <email>bbc@abc.com</email>
        <exam name="exam1" max="100">
            <score>99</score>
        </exam>
        <exam name="exam2" max="70">
            <score>66</score>
        </exam>
        <ta name="kaumudi"/>
        <sect name="6248"/>
    </student>
</course>
```

Fig. 2. XML student records (student.xml)

presentation. We write an XSL style sheet which can be applied using an XSLT processor like xsltproc. This converts the XML document to something useful like html or comma separated file or even pdf.

A. Student Roster Style sheet

First, I wrote a simple XSL style sheet which just lists the information in plain tabular form. The XSL style sheet is in figure 3 and the resulting HTML is shown in figure 4

If you look at the original document closely, you observe that the phone number is divided into area code and number, which makes it easy to use a stylesheet which outputs a phone number like

```
[areacode]-[number]
```

If one day we decide to get rid of the '-', we can simply delete a character from the stylesheet.

B. Student Interface Style sheet

I wrote another style sheet, which provides a student interface. The XSL style sheet and output HTML are shown in figure 5 and 6 respectively.

This style sheet is much simpler and I could have used sorting as well.

C. Style sheet for producing averages

Another style sheet, just prints the averages of all exams. Interestingly, I figured out that having ¡exam¿ nodes with attribute names like `<exam name="exam1">` is cumbersome to manipulate. So I changed student.xml to 7

This listing has similar content except that ¡exam1¿ ¡exam2¿ nodes replacing the plain ¡exam¿ nodes. This is a bit of hard-coding but makes things simpler. Coming back to averages, the style sheet is pretty simple. The style sheet and resulting HTML are shown in figure 8 and 9 respectively.

V. Summary and Conclusions

We have described a simple method to represent student records in XML. XSL transformations can be used to convert the original XML into various useful forms. The following is a summary of the techniques.

- XML/XSLT solves the problem of multiple people working on the student grades. Different TAs just work with their XML documents and turn it over to main person entering grades. All he has to do is to cat them and apply the style sheet. The style sheets will take care of sorting etc...
- The document need not have the nodes in order. The TA need not worry about having `<lastname>` node before `<firstname>`. He/She can put them in any order he/she wishes. For consistency, a simple DTD can be created.
- It's text based. So people like me can use their favourite text editor to enter the grades. Or We can create a web-based front end.
- We can also create a comma separated fields file and import it into Excel

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

<xsl:template match="/course">
<html>
<head>
<title>
    <xsl:value-of select="@name"/> scores
</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
</head>
<body>
    <table border="2">
    <tr>
        <th>First Name</th> <th>Second Name</th> <th>Password</th>
        <th>SSN</th> <th>Class</th> <th>Major</th> <th>Phone</th>
        <th>E-Mail</th> <th>Username</th> <th>TA</th> <th>Section</th>
        <th>Exam1</th> <th>Exam2</th>
    </tr>
        <xsl:apply-templates/>
    </table>
</body>
</html>
</xsl:template>

<xsl:template match="student">
<tr>
    <xsl:apply-templates select="firstname"/> <xsl:apply-templates select="lastname"/>
    <xsl:apply-templates select="password"/> <xsl:apply-templates select="ssn"/>
    <xsl:apply-templates select="class"/> <xsl:apply-templates select="major"/>
    <xsl:apply-templates select="phone"/> <xsl:apply-templates select="email"/>
    <xsl:apply-templates select="username"/> <xsl:apply-templates select="ta"/>
    <xsl:apply-templates select="sect"/> <xsl:apply-templates select="exam"/>
</tr>
</xsl:template>
<xsl:template match="ta|sect">
<td>
    <xsl:value-of select="@name"/>
</td>
</xsl:template>
<xsl:template match="phone">
<td>
    <xsl:value-of select="areacode"/>-<xsl:value-of select="number"/>
</td>
</xsl:template>
<xsl:template match="exam">
<td>
    <xsl:value-of select="score"/> out of <xsl:value-of select="@max"/>
</td>
</xsl:template>

<!--Default Template -->
<xsl:template match="*">
<td>
    <xsl:value-of select="."/>
</td>
</xsl:template>
<!--Ignore the text nodes (the white spaces)-->
<xsl:template match="text()">
</xsl:template>
</xsl:stylesheet>
```

Fig. 3. roster.xsl

First Name	Second Name	Password	SSN	Class	Major	Phone	E-Mail	Username	TA	Section	Exam1	Exam2
Kevin	Ayers	a1234xyz	12345678	7EG	CEN	352-1234567	abc@bbc.com	kma	ppadala	6261	61 out of 100	50 out of 70
John	Buisson	c1234abc	12345678	7EG	CEN	352-1234467	bbc@abc.com	jb	kaumudi	6248	99 out of 100	66 out of 70

Fig. 4. HTML produced by roster.xsl stylesheet

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

<xsl:template match="/course">
<html>
<head>
<title>
    <xsl:value-of select="@name"/> Student Interface
</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
</head>
<body>
    <table border="2">
    <tr><th>Student Name</th> <th>Exam1</th> <th>Exam2</th>
    </tr>
    <xsl:apply-templates select="student"/>
    </table>
</body>
</html>
</xsl:template>

<xsl:template match="student">
    <tr>
    <td><xsl:value-of select="firstname"/>
            <xsl:text> </xsl:text>
        <xsl:value-of select="lastname"/>
    </td>
    <xsl:for-each select="exam">
    <xsl:if test="contains(@name, 'exam1')">
        <td>
        <xsl:value-of select="score"/>
        </td>
    </xsl:if>
    <xsl:if test="contains(@name, 'exam2')">
        <td>
        <xsl:value-of select="score"/>
        </td>
    </xsl:if>
    </xsl:for-each>
    </tr>
</xsl:template>

<!--Ignore the text nodes (the white spaces)-->
<xsl:template match="text()">
</xsl:template>

</xsl:stylesheet>
```

Fig. 5. stdinterface.xsl

Student Name	Exam1	Exam2
Kevin Ayers	61	50
John Buisson	99	66

Fig. 6. HTML produced by stdinterface.xsl stylesheet

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

<xsl:variable name="nstds">
    <xsl:value-of select="count(course/student)"/>
</xsl:variable>

<xsl:template match="/course">
<html>
<head>
<title>
    <xsl:value-of select="@name"/> Exam Scores
</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"/>
</head>
<body>
    <table border="2">
    <tr><th>Exam Name</th> <th>Average</th>
    </tr>
    <tr>
        <td>Exam 1</td>
        <td><xsl:value-of select="sum(student/exam1/score)
        div $nstds"/> out of <xsl:value-of select="student/exam1/@max"/></td>
    </tr>
    <tr>
        <td>Exam 2</td>
        <td><xsl:value-of select="sum(student/exam2/score)
        div $nstds"/> out of <xsl:value-of select="student/exam2/@max"/></td>
    </tr>
    </table>
</body>
</html>
</xsl:template>

<!--Ignore the text nodes (the white spaces)-->
<xsl:template match="text()">
</xsl:template>

</xsl:stylesheet>
```

Fig. 8. averages.xsl

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

<course name="se">
    <student>
        <firstname>Kevin</firstname>
        <lastname>Ayers</lastname>
        <password>a1234xyz</password>
        <ssn>12345678</ssn>
        <class>7EG</class>
        <major>CEN</major>
        <phone>
            <areacode>352</areacode>
            <number>1234567</number>
        </phone>
        <email>abc@bbc.com</email>
        <username>kma</username>
        <exam1 max="100">
            <score>61</score>
        </exam1>
        <exam2 max="70">
            <score>50</score>
        </exam2>
        <ta name="ppadala"/>
        <sect name="6261"/>
    </student>

    <student>
        <firstname>John</firstname>
        <lastname>Buisson</lastname>
        <password>c1234abc</password>
        <ssn>12345678</ssn>
        <class>7EG</class>
        <phone>
            <areacode>352</areacode>
            <number>1234467</number>
        </phone>
        <major>CEN</major>
        <username>jb</username>
        <email>bbc@abc.com</email>
        <exam1 max="100">
            <score>99</score>
        </exam1>
        <exam2 max="70">
            <score>66</score>
        </exam2>
        <ta name="kaumudi"/>
        <sect name="6248"/>
    </student>
</course>
```

Fig. 7. Modified student record XML document

Exam Name	Average
Exam 1	80 out of 100
Exam 2	58 out of 70

Fig. 9. HTML produced by averages.xsl stylesheet

REFERENCES

[1] T. Bray, J. Paoli, and C. M. Sperberg-McQueen (Eds), ""Extensible Markup Language (XML) 1.0 (2nd Edition)"," W3C Recommendation, 2000.

[2] "XSL Transformations (XSLT)," 1999, http://www.w3.org/TR/xslt.