docs.microsoft.com

Blog Tales: Introduction to Excel XML

Archiveddocs

13-16 минут

- 09/08/2016
- 8 minutes to read

In this article

Blog Tales Introduction to Excel XML

Brian Jones

With the soon-to-be released next version of Microsoft® Office (currently code-named "Office 12"), there will be new default file formats for Microsoft Word, PowerPoint®, and Excel®. These new formats, called the Microsoft Office Open XML Formats, will open up a whole new world to Office developers. By default, Office documents will be open and accessible, as they will use standard ZIP and XML technologies with full documentation made available under a royalty-free license. These technologies are an improvement on the existing XML formats that shipped with Microsoft Office 2003 Editions, but those existing Office 2003 XML Reference Schemas can be used today to implement solutions that work with the document data and they provide a great way to gain an understanding of what developing with the new default formats will entail.

The SpreadsheetML format in Microsoft Excel is fairly easy to work with, as it was designed especially to be human readable and editable. But many of you probably haven’t had a chance to take a look at the XML support in Excel. Once you get a handle on how it works, though, you’ll realize you have plenty of uses for the XML features, from converting data between databases and Web pages to sharing files among disparate applications.

To get you started, I’ll build a sample in XML that will illustrate how it all works. As you follow along, you can use Office XP or Office 2003 for this example since both support SpreadsheetML in their versions of Excel. Using a text editor, I’m going to create a very simple table that looks like **Figure 1**, outlining seven steps to create an XML file that represents an Excel worksheet.

Figure 1 The Table Example

1. Create the XML File

To begin, create a new file in Notepad, and call it test.xml. Then follow the steps outlined here. First type the following:

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

This declares that the file is an XML document adhering to the 1.0 version of the XML spec. It should always be found at the top of all your XML files. Next add the root element for the document. XML files always have one and only one root element that contains the rest of the document. For SpreadsheetML, the root element is <Workbook>. After the XML declaration, add that element so that your file now looks like this:

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

2. Declare the Namespace

Now you'll declare the namespace and add a prefix to the root element. Most XML documents have a namespace associated with them. Declaring the namespace of an XML file makes it a lot easier for users parsing your XML to know what type of XML they are dealing with. Even in Office there are a number of different uses for XML. One way to know when you are parsing a Word XML file as opposed to an Excel XML file, for example, is to look at the namespace. With Office XP, when the product group created the SpreadsheetML schema, we were still using namespaces in the form "urn:schemas-microsoft-com:office". Going forward, we'll use URL namespaces, as we did with WordML in Office 2003 (//schemas.microsoft.com/office, for example). By adding the namespace declaration to the spreadsheet, your file should look like this:

```
<?xml version="1.0"?>
<Workbook xmlns="urn:schemas-microsoft-
com:office:spreadsheet">
</Workbook>
```

The last thing you'll do for the namespace is use a prefix, rather than the default. Since the attributes are qualified for the SpreadsheetML schema, you need to do this if you are going to use any attributes. Let's use "ss" (for spreadsheet) as the prefix. You'll add "ss:" in front of all of your elements, and you'll update your namespace declaration to say that the namespace applies to everything with an "ss:" in front of it, instead of just applying to the default XML elements, as shown here:

```
<?xml version="1.0"?>
<ss:Workbook xmlns:ss="urn:schemas-microsoft-
com:office:spreadsheet">
</ss:Workbook>
```

Notice that the namespace declaration says xmlns:ss= instead of just xmlns=. This means that anything with an "ss:" in front of it applies to the spreadsheet namespace.

3. Add a Worksheet

Next you'll add a worksheet. Since you have an empty workbook, you need to declare the spreadsheet grid within the workbook. As you may know, workbooks can have multiple worksheets, but here you'll just declare one. In addition, let's declare a table inside the worksheet. The table is where all the grid data will go, and the file will now look like this:

```
<?xml version="1.0"?>
<ss:Workbook xmlns:ss="urn:schemas-microsoft-
com:office:spreadsheet">
    <ss:Worksheet ss:
     Name="Sheet1">
        <ss:Table>
        </ss:Table>
    </ss:Worksheet>
</ss:Workbook>
```

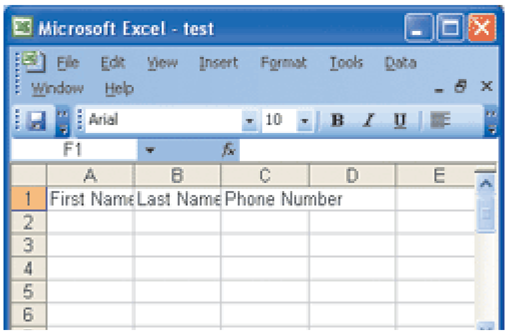
4. Add the Header Row

The first row in the table you want to generate has "First Name", "Last Name", and "Phone Number" in the three columns. Let's add a <Row> tag as well as three <Cell> tags. The actual content of the cell is contained within a <Data> tag, so let's add that as well. The file now looks like **Figure 2**.

Figure 2 XML Worksheet Takes Form

```
<?xml version="1.0"?>
<ss:Workbook xmlns:ss="urn:schemas-microsoft-
com:office:spreadsheet">
    <ss:Worksheet ss:Name="Sheet1">
        <ss:Table>
            <ss:Row>
                <ss:Cell>
                    <ss:Data
ss:Type="String">First Name</ss:Data>
                </ss:Cell>
                <ss:Cell>
                    <ss:Data
ss:Type="String">Last Name</ss:Data>
                </ss:Cell>
                <ss:Cell><ss:Data
ss:Type="String">Phone Number</ss:Data>
                </ss:Cell>
            </ss:Row>
        </ss:Table>
    </ss:Worksheet>
</ss:Workbook>
```

You now have a template for the table that you can open directly in Excel. It will look like **Figure 3**. Not too exciting, but it's a start.



	A	B	C	D	E
1	First Name	Last Name	Phone Number		
2					
3					
4					
5					
6					

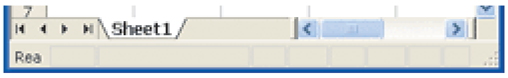


Figure 3 **Rudimentary Worksheet**

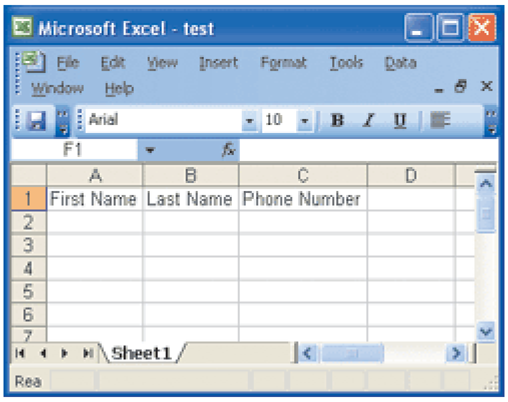
5. Adjust the Column Widths

Notice that the widths of the columns are too narrow for the content. Let’s add some XML to the file to specify the width you want for the columns. The resulting code is shown in **Figure 4**.

Figure 4 Resizing the Columns

```
<?xml version="1.0"?>
<ss:Workbook xmlns:ss="urn:schemas-microsoft-
com:office:spreadsheet">
    <ss:Worksheet ss:Name="Sheet1">
        <ss:Table>
            <ss:Row>
                <ss:Cell>
                    <ss:Data
ss:Type="String">First Name</ss:Data>
                </ss:Cell>
                <ss:Cell>
                    <ss:Data
ss:Type="String">Last Name</ss:Data>
                </ss:Cell>
                <ss:Cell><ss:Data
ss:Type="String">Phone Number</ss:Data>
                </ss:Cell>
            </ss:Row>
        </ss:Table>
    </ss:Worksheet>
</ss:Workbook>
```

Now open the file again in Excel. Notice that the columns are wider and that the text now fits (see **Figure 5**). There is another attribute you can set on the column element that tells it to use autofit for the widths. This only works for numbers and dates though. Since your cells are strings, you need to explicitly set the width.



A	B	C	D
First Name	Last Name	Phone Number	

Figure 5 **Resized Cells**

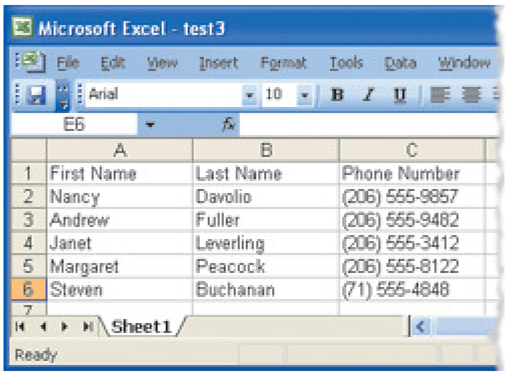
6. Add the Remaining Data

Now add those additional rows of data. This should be pretty easy. Just select that first "row" element and copy it. Then paste it five more times so you have six total rows. Now go through and update the values of the rows. If you are familiar with Extensible Stylesheet Language Transform (XSLT), you'll see how you could easily generate an XSLT that could be applied to a DataSet to transform it into SpreadsheetML. Just repeat the Row tag for each row in your DataSet and add the values in each cell's Data tag. After applying all the data, your XML should look like **Figure 6**, which has been abbreviated for space. **Figure 7** shows the full table in Excel.

Figure 6 XML Table with all Data

```
<?xml version="1.0"?>
<ss:Workbook xmlns:ss="urn:schemas-microsoft-
com:office:spreadsheet">
    <ss:Worksheet ss:Name="Sheet1">
```

```
        <ss:Table>
            <ss:Column ss:Width="80"/>
            <ss:Column ss:Width="80"/>
            <ss:Column ss:Width="80"/>
            <ss:Row>
                <ss:Cell>
                   <ss:Data
ss:Type="String">First Name</ss:Data>
                </ss:Cell>
                <ss:Cell>
                   <ss:Data
ss:Type="String">Last Name</ss:Data>
                </ss:Cell>
                <ss:Cell>
                   <ss:Data
ss:Type="String">Phone Number</ss:Data>
                </ss:Cell>
            </ss:Row>
            <ss:Row>
                <ss:Cell>
                   <ss:Data
ss:Type="String">Nancy</ss:Data>
                </ss:Cell>
                <ss:Cell>
                   <ss:Data
ss:Type="String">Davolio</ss:Data>
                </ss:Cell>
                <ss:Cell>
                   <ss:Data ss:Type="String">
(206)555 9857</ss:Data>
                </ss:Cell>
            </ss:Row>
            <ss:Row>
            ...
            </ss:Row>
        </ss:Table>
    </ss:Worksheet>
</ss:Workbook>
```



	A	B	C
1	First Name	Last Name	Phone Number
2	Nancy	Davolio	(206) 555-9857
3	Andrew	Fuller	(206) 555-9482
4	Janet	Leverling	(206) 555-3412
5	Margaret	Peacock	(206) 555-8122
6	Steven	Buchanan	(71) 555-4848
7			

Figure 7 **Worksheet with Data**

7. Add Header Formatting

As you can see, the first row does not look like a column header, so let’s format it with bold text so that it’s clearly the header. All you need to do is generate a style that has bold text, and then reference that style with the first row. First, add the following XML in front of the Worksheet tag:

```
<ss:Styles>
    <ss:Style ss:ID="1">
        <ss:Font ss:Bold="1"/>
    </ss:Style>
</ss:Styles>
```

This creates a style whose ID is "1" and has bold applied to it. Next, update the first row element to reference StyleID 1. The row code should now look like this:

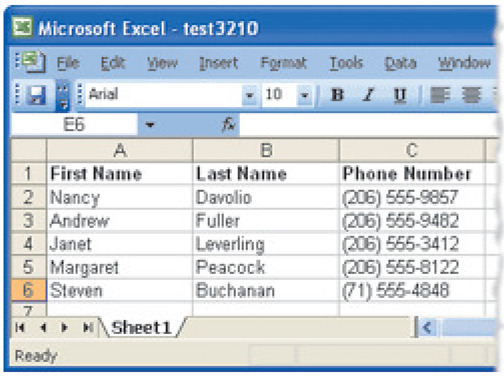
```
<ss:Row ss:SyleID="1">
```

Your XML should now look like **Figure 8**, and **Figure 9** shows how it looks in Excel.

Figure 8 Bolding the Header Row

```
<?xml version="1.0"?>
<ss:Workbook xmlns:ss="urn:schemas-microsoft-
com:office:spreadsheet">
    <ss:Styles>
        <ss:Style ss:ID="1">
            <ss:Font ss:Bold="1"/>
        </ss:Style>
    </ss:Styles>
    <ss:Worksheet ss:Name="Sheet1">
        <ss:Table>
            <ss:Column ss:Width="80"/>
            <ss:Column ss:Width="80"/>
            <ss:Column ss:Width="80"/>
            <ss:Row ss:StyleID="1">
                <ss:Cell>
                   <ss:Data
ss:Type="String">First Name</ss:Data>
                </ss:Cell>
                <ss:Cell>
                   <ss:Data
ss:Type="String">Last Name</ss:Data>
                </ss:Cell>
                <ss:Cell>
                   <ss:Data
ss:Type="String">Phone Number</ss:Data>
                </ss:Cell>
            </ss:Row>
            <ss:Row>
```

```
                <ss:Cell>
                   <ss:Data
ss:Type="String">Nancy</ss:Data>
                </ss:Cell>
                <ss:Cell>
                   <ss:Data
ss:Type="String">Davolio</ss:Data>
                </ss:Cell>
                <ss:Cell>
                   <ss:Data ss:Type="String">
(206)555-9857</ss:Data>
                </ss:Cell>
            </ss:Row>
            ...
            </ss:Row>
        </ss:Table>
    </ss:Worksheet>
</ss:Workbook>
```


Microsoft Excel - test3210

	A	B	C
1	First Name	Last Name	Phone Number
2	Nancy	Davolio	(206) 555-9857
3	Andrew	Fuller	(206) 555-9482
4	Janet	Leverling	(206) 555-3412
5	Margaret	Peacock	(206) 555-8122
6	Steven	Buchanan	(71) 555-4848
7			

Figure 9 **The Completed Worksheet**

Wrap-Up

That was a pretty simple example, but it’s a good introduction if you’re new to Office XML (or even new to XML in general). The new XML formats for future versions of Excel will look different than what I’ve shown you with SpreadsheetML, but there will also be some similarities. It’s good to become familiar with the existing schemas, and I'll start posting a lot more about the new schemas on my blog at blogs.msdn.com/brian_jones.

Blog Tales Resources

Are you looking to get a jump on becoming an Office Open XML Formats expert? Or are you just curious about how the changes in the next version of Microsoft Office will impact your environment? Check out these sites for more information.

XML in Office Development

Whether you're just getting up to speed with XML or you've been making use of XML support in Microsoft Office since the release of Office 2003, there is always more to learn. The Office Developer Center has an excellent section dedicated to XML in Office Development. Here you'll find tutorials, code samples, reference documentation, and useful tools.

XML in Office Technical Articles

Looking for some not-so-light reading about XML? Check out this collection of in-depth articles on using XML to improve how your organization works.

Jensen Harris: An Office User Interface Blog

While changes to XML in Microsoft Office are buried beneath the surface, the new user interfaces found in familiar apps like Microsoft Excel and Word are a bit more readily evident. Jensen Harris, a member of the Microsoft Office user experience team, keeps a blog where he discuss the new UI designs in Office applications and explains why these changes were made.

Brian Jones is a program manager at Microsoft working on XML functionality and file formats in Office. Most recently, Brian has worked on the Microsoft Office Open XML Formats that will be introduced in Office 12. This column was adapted from his blog, which can be found at blogs.msdn.com/brian_jones.