C# How To Read .xlsx Excel File With 3 Lines of Code

Dietmar Schoder, 29 Jul 2014 CPOL

4.92 (44 votes)

Rate:

Read rows and cells from an xlsx-file: quick, dirty, effective.

Download ExcelDLL.zip - 6 KB
Download Excel.zip - 12.3 KB

Introduction

Our customers often have to import data from very simple Excel *.xslx-files into our software product: with some relevant rows and cells in rather primitive worksheets of an Excel workbook, and that's it. But we do not want to use large DLL's or third party software for that. Therefore we produced a small solution for our needs. It could be useful for you, too:

Using the code

Download the "Excel.dll" (12 kByte, you need .net 4.5!) and add it as a reference to your project. Or adapt the compact source code before - with only the relevant classes "`Workbook`", "`Worksheet`", "`Row`" and "`Cell`" and 45 C# code lines doing important work in total. Then read the worksheets, rows and cells from any Excel *.xlsx-file in your program like so:

Collapse | Copy Code

```
foreach (var worksheet in Workbook.Worksheets(@"C:\ExcelFile.xlsx")
    foreach (var row in worksheet.Rows)
        foreach (var cell in row.Cells)
            // if (cell != null) // Do something with the cells
```

Here you iterate through the worksheets, the rows (and the cells of each row) of the Excel file within three lines of code.

Points of Interest

This article (written by M I developer) describes all the theoretical background, if you are interested in it. We based our solution on the integrated ZIP-library in .net 4.5 and the standard XML-serializer of .net.

If you want to adapt our solution to your needs: edit the simple source code for the Excel.dll. This is how it works:

Maybe you did not know that xlsx-files are ZIP-files. And the text strings of the Excel cells of all worksheets per workbook are always stored in a file named "`xl/sharedStrings.xml`", while the worksheets are called "`xl/worksheets/sheet[1...n].xml`".

So we have to unzip and deserialize the relevant XML files in the Excel xlsx-file:

Collapse | Copy Code

```
using System.IO.Compression;

public static IEnumerable<worksheet> Worksheets(string ExcelFileName)
```

```
{
    worksheet ws;

    using (ZipArchive zipArchive = ZipFile.Open(ExcelFileName, ZipArchiveMode.Read))
    {
        SharedStrings = DeserializedZipEntry<sst>(GetZipArchiveEntry(zipArchive,
@"xl/sharedStrings.xml"));
        foreach (var worksheetEntry in (WorkSheetFileNames(zipArchive)).OrderBy(x => x.FullName))
        {
            ws = DeserializedZipEntry<worksheet>(worksheetEntry);
            ws.ExpandRows();
            yield return ws;
        }
    }
}
```

As you see, we also have to find all worksheets of the workbook at the beginning. We filter the ZIP-archive entries for that:

Collapse | Copy Code

```
private static IEnumerable<ziparchiveentry> WorkSheetFileNames(ZipArchive ZipArchive)
{
    foreach (var zipEntry in ZipArchive.Entries)
        if (zipEntry.FullName.StartsWith("xl/worksheets/sheet"))
            yield return zipEntry;
}
```

For deserialization of each XML formatted ZIP-entry (see also this article written by Md. Rashim uddin) we use this generic method:

Collapse | Copy Code

```
private static T DeserializedZipEntry<T>(ZipArchiveEntry ZipArchiveEntry)
{
    using (Stream stream = ZipArchiveEntry.Open())
        return (T)new XmlSerializer(typeof(T)).Deserialize(XmlReader.Create(stream));
}
```

Therefore the XML-structures have to be reflected in our classes. Here you see the "`sst`"-class and the "`SharedString`"-class for the XML in the "shared strings table":

Collapse | Copy Code

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<sst xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main" count="72" uniqueCount="6">
  <si>
    <t>Text A</t>
  </si>
  <si>
    <t>Text B</t>
  </si>
</sst>
```

Collapse | Copy Code

```
public class sst
{
    [XmlElement("si")]
    public SharedString[] si;

    public sst()
    {
    }
}

public class SharedString
{
    public string t;
}
```

The same strategy we also use for the "worksheet" -XML-file in the ZIP-file. There we focus on the XML-elements and -attributes "`row`", "`c`", "`v`", "`r`" and "`t`". All the work is done again by the `XmlSerializer`:

Collapse | Copy Code

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<worksheet xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main">
<dimension ref="A1:F12"/>
<sheetViews>
  <sheetView workbookViewId="0"></sheetView>
</sheetViews>
<sheetFormatPr baseColWidth="10" defaultRowHeight="15"/>
<sheetData>
  <row r="1">
    <c r="A1" t="s">
      <v>0</v>
    </c>
    <c r="B1" t="s">
      <v>1</v>
    </c>
    <c r="C1" t="s">
      <v>2</v>
    </c>
  </row>
</sheetData>
</worksheet>
```

Collapse | Copy Code

```
public class worksheet
{
    [XmlArray("sheetData")]
    [XmlArrayItem("row")]
    public Row[] Rows;

    public class worksheet
    {
    }
}
public class Row
{
    [XmlElement("c")]
    public Cell[] FilledCells;
}
public class Cell
{
    [XmlAttribute("r")]
    public string CellReference;
    [XmlAttribute("t")]
    public string tType = "";
    [XmlElement("v")]
    public string Value;
}
```

Of course we have to do a little bit in order to convert the usual Excel cell references like "A1", "B1" and so on to column indices. That is done via the setter of "CellReference" in the "Cell"-class (here we also derive the maximum column index for the whole worksheet) ...

Collapse | Copy Code

```
[XmlAttribute("r")]
public string CellReference
{
    get
    {
        return ColumnIndex.ToString();
    }
    set
    {
        ColumnIndex = worksheet.GetColumnIndex(value);
        if (ColumnIndex > worksheet.MaxColumnIndex)
            worksheet.MaxColumnIndex = ColumnIndex;
    }
}
```

... and a small method named "GetColumnIndex()":

Collapse | Copy Code

```
private int GetColumnIndex(string CellReference)
{
    string colLetter = new Regex("[A-Za-z]+").Match(CellReference).Value.ToUpper();
    int colIndex = 0;

    for (int i = 0; i < colLetter.Length; i++)
    {
        colIndex *= 26;
        colIndex += (colLetter[i] - 'A' + 1);
    }
    return colIndex - 1;
}
```

The last challenge has to do with the fact, that the Excel file does not contain empty Excel cells. So the tiny methods "ExpandRows()" and "ExpandCells()" handle that problem:

Collapse | Copy Code

```
public void ExpandRows()
{
    foreach (var row in Rows)
        row.ExpandCells(NumberOfColumns);
}

public void ExpandCells(int NumberOfColumns)
{
    Cells = new Cell[NumberOfColumns];
    foreach (var cell in FilledCells)
        Cells[cell.ColumnIndex] = cell;
    FilledCells = null;
}
```

In the end we have an array of all rows and an array of all cells for each row representing all columns of the specific Excel worksheet. Empty cells are null in the array, but the ColumnIndex of each cell in "Row.Cells[]" corresponds with the actual Excel column of each cell.

Unfortunately the XML format is not very clear about how to interpret the value of the Excel cells. We tried to do it like so, but any hint for improvement would be appreciated:

Collapse | Copy Code

```
if (tType.Equals("s"))
{
    Text = Workbook.SharedStrings.si[Convert.ToInt32(_value)].t;
    return;
}
if (tType.Equals("str"))
{
```

```
    Text = _value;
    return;
}
try
{
    Amount = Convert.ToDouble(_value, CultureInfo.InvariantCulture);
    Text = Amount.ToString("#,##0.##");
    IsAmount = true;
}
catch (Exception ex)
{
    Amount = 0;
    Text = String.Format("Cell Value '{0}': {1}", _value, ex.Message);
}
```

Besides, when you know that an Excel cell contains a date as its value, you can use this method for conversion:

Collapse | Copy Code

```
public static DateTime DateFromExcelFormat(string ExcelCellValue)
{
    return DateTime.FromOADate(Convert.ToDouble(ExcelCellValue));
}
```

Let me know how the total Excel.DLL works in your environment - and have fun with it!

History

26.7.2014 - Posted initially.

26.7.2014 - Files uploaded here.

26.7.2014 - Explanations added, formatting improved.

28.7.2014 - More explanation added. Deserialization slightly improved.

29.7.2014 - Date conversion from Excel format to DateTime via DateTime.FromOADate()

29.7.2014 - Essence of class "worksheet" added.

31.7.2014 - Comments added in the source code, XML-documentation added to the dll

31.7.2014 - The program now reads all worksheets from a workbook, not only the first one

License

Share

About the Author

Dietmar Schoder
Founder http://www.wbstool8.com
Austria

Dietmar Schoder is the architect and developer of http://www.wbstool8.com. He is a convinced user of Visual Studio, C#, .net, WPF, MVVM, the project management standards of PMI and IPMA - and Microsoft Surface Pro 3. He lives in Vienna (Austria) and Eastbourne (UK).

Follow on Twitter LinkedIn

Comments and Discussions

Add a Comment or Question | Search Comments | Go

Profile popups | Spacing: Relaxed | Noise: Medium | Layout: Normal | Per page: 25 | Update

First Prev Next

Exception when loading file, not handling empty rows new	**ThomasD3**	**18-Jan-15 18:01**
Re: Exception when loading file, not handling empty rows new	Paul C. Rhodes	19-Jan-15 1:30
Can write Excel file ??	**Member 10039732**	**10-Jan-15 9:54**
Re: Can write Excel file ??	Paul C. Rhodes	10-Jan-15 10:22
Roughly half the rows have at least one cell returning null for the cell.Text	**Member 11341566**	**6-Jan-15 11:21**
System.NullReferenceException: Object reference not set to an instance of an object.	**Member 11341566**	**6-Jan-15 11:04**
End of Worksheet	**Member 10039732**	**3-Jan-15 10:30**
Re: End of Worksheet	Paul C. Rhodes	3-Jan-15 10:39
Re: End of Worksheet	Member 10039732	4-Jan-15 10:47
Re: End of Worksheet	Paul C. Rhodes	4-Jan-15 10:55
Re: End of Worksheet	Member 10039732	7-Jan-15 8:21
Sequence contains no matching element.	**Logan Apple**	**24-Dec-14 14:01**
example of use	**bombin**	**18-Dec-14 9:38**
Re: example of use	Dietmar Schoder	18-Dec-14 10:36
My Vote of Excellent	**alberto saint**	**11-Dec-14 14:23**
Using null cells	**Member 11250873**	**20-Nov-14 15:23**
Re: Using null cells	Dietmar Schoder	21-Nov-14 1:44
Re: Using null cells	Member 11250873	24-Nov-14 13:23
Re: Using null cells	Dietmar Schoder	24-Nov-14 13:44
Works well!	**Member 11243678**	**19-Nov-14 6:10**
Cells with Date are not reading properly??	**sukeshchand**	**17-Nov-14 2:09**
Re: Cells with Date are not reading properly??	Dietmar Schoder	17-Nov-14 2:40
CellReference?	**jggregory**	**11-Nov-14 20:39**
Re: CellReference?	Dietmar Schoder	12-Nov-14 2:29
Re: CellReference? [modified]	jggregory	12-Nov-14 7:25

Last Visit: 16-Jan-15 1:13 Last Update: 1-Feb-15 20:53 Refresh 1 2 3 4 5 6 Next »

General News Suggestion Question Bug Answer Joke Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.