Export a DataSet to Microsoft Excel without the use of COM objects

https://www.codeproject.com/Articles/9380/Export-a-DataSet-to-Microsoft-Excel-without-the-us

Xodiak - 28 May 2005

A simple function that writes a DataSet to a Microsoft Excel document.

Introduction

This function takes in a `DataSet` and file name and writes the `DataSet` to an Excel worksheet. The code is pretty straightforward. Great thing about this function is that, it's technically an XML file that is saved as an XLS file. So it can be used as either file format. No more leading zero truncation on numbers that look like strings. Example, if you made a tab delimited file and put a field such as "00036" (a field that looks like a number but should be regarded as a string), MS Excel would truncate the leading zeros... This problem is solved with this method.

Here is the code:

C#

```
public static void exportToExcel(DataSet source, string fileName)

{

    System.IO.StreamWriter excelDoc;

    excelDoc = new System.IO.StreamWriter(fileName);
    const string startExcelXML = "<xml version>\r\n<Workbook " +
          "xmlns=\"urn:schemas-microsoft-com:office:spreadsheet\"\r\n" +
          " xmlns:o=\"urn:schemas-microsoft-com:office:office\"\r\n " +
          "xmlns:x=\"urn:schemas-    microsoft-com:office:" +
          "excel\"\r\n xmlns:ss=\"urn:schemas-microsoft-com:" +
          "office:spreadsheet\">\r\n <Styles>\r\n " +
          "<Style ss:ID=\"Default\" ss:Name=\"Normal\">\r\n " +
          "<Alignment ss:Vertical=\"Bottom\"/>\r\n <Borders/>" +
          "\r\n <Font/>\r\n <Interior/>\r\n <NumberFormat/>" +
          "\r\n <Protection/>\r\n </Style>\r\n " +
          "<Style ss:ID=\"BoldColumn\">\r\n <Font " +
          "x:Family=\"Swiss\" ss:Bold=\"1\"/>\r\n </Style>\r\n " +
          "<Style     ss:ID=\"StringLiteral\">\r\n <NumberFormat" +
          " ss:Format=\"@\"/>\r\n </Style>\r\n <Style " +
          "ss:ID=\"Decimal\">\r\n <NumberFormat " +
          "ss:Format=\"0.0000\"/>\r\n </Style>\r\n " +
          "<Style ss:ID=\"Integer\">\r\n <NumberFormat " +
          "ss:Format=\"0\"/>\r\n </Style>\r\n <Style " +
```

```
          "ss:ID=\"DateLiteral\">\r\n <NumberFormat " +
          "ss:Format=\"mm/dd/yyyy;@\"/>\r\n </Style>\r\n " +
          "</Styles>\r\n ";
      const string endExcelXML = "</Workbook>";

      int rowCount = 0;
      int sheetCount = 1;
      /*
     <xml version>
     <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">
     <Styles>
     <Style ss:ID="Default" ss:Name="Normal">
       <Alignment ss:Vertical="Bottom"/>
       <Borders/>
       <Font/>
       <Interior/>
       <NumberFormat/>
       <Protection/>
     </Style>
     <Style ss:ID="BoldColumn">
       <Font x:Family="Swiss" ss:Bold="1"/>
     </Style>
     <Style ss:ID="StringLiteral">
       <NumberFormat ss:Format="@"/>
     </Style>
     <Style ss:ID="Decimal">
       <NumberFormat ss:Format="0.0000"/>
     </Style>
     <Style ss:ID="Integer">
       <NumberFormat ss:Format="0"/>
     </Style>
     <Style ss:ID="DateLiteral">
       <NumberFormat ss:Format="mm/dd/yyyy;@"/>
     </Style>
     </Styles>
     <Worksheet ss:Name="Sheet1">
     </Worksheet>
     </Workbook>
     */
     excelDoc.Write(startExcelXML);
     excelDoc.Write("<Worksheet ss:Name=\"Sheet" + sheetCount + "\">");
     excelDoc.Write("<Table>");
     excelDoc.Write("<Row>");
     for(int x = 0; x < source.Tables[0].Columns.Count; x++)
     {
       excelDoc.Write("<Cell ss:StyleID=\"BoldColumn\"><Data
ss:Type=\"String\">");
       excelDoc.Write(source.Tables[0].Columns[x].ColumnName);
       excelDoc.Write("</Data></Cell>");
     }
     excelDoc.Write("</Row>");
     foreach(DataRow x in source.Tables[0].Rows)
     {
```

```
      rowCount++;
      //if the number of rows is > 64000 create a new page to continue output
      if(rowCount==64000)
      {
        rowCount = 0;
        sheetCount++;
        excelDoc.Write("</Table>");
        excelDoc.Write(" </Worksheet>");
        excelDoc.Write("<Worksheet ss:Name=\"Sheet" + sheetCount + "\">");
        excelDoc.Write("<Table>");
      }
      excelDoc.Write("<Row>"); //ID=" + rowCount + "
      for(int y = 0; y < source.Tables[0].Columns.Count; y++)
      {
        System.Type rowType;
        rowType = x[y].GetType();
        switch(rowType.ToString())
        {
          case "System.String":
             string XMLstring = x[y].ToString();
             XMLstring = XMLstring.Trim();
             XMLstring = XMLstring.Replace("&","&");
             XMLstring = XMLstring.Replace(">",">");
             XMLstring = XMLstring.Replace("<","<");
             excelDoc.Write("<Cell ss:StyleID=\"StringLiteral\">" +
                            "<Data ss:Type=\"String\">");
             excelDoc.Write(XMLstring);
             excelDoc.Write("</Data></Cell>");
             break;
           case "System.DateTime":
             //Excel has a specific Date Format of YYYY-MM-DD followed by
             //the letter 'T' then hh:mm:sss.lll Example 2005-01-
31T24:01:21.000
             //The Following Code puts the date stored in XMLDate
             //to the format above
             DateTime XMLDate = (DateTime)x[y];
             string XMLDatetoString = ""; //Excel Converted Date
             XMLDatetoString = XMLDate.Year.ToString() +
                  "-" +
                  (XMLDate.Month < 10 ? "0" +
                  XMLDate.Month.ToString() : XMLDate.Month.ToString()) +
                  "-" +
                  (XMLDate.Day < 10 ? "0" +
                  XMLDate.Day.ToString() : XMLDate.Day.ToString()) +
                  "T" +
                  (XMLDate.Hour < 10 ? "0" +
                  XMLDate.Hour.ToString() : XMLDate.Hour.ToString()) +
                  ":" +
                  (XMLDate.Minute < 10 ? "0" +
                  XMLDate.Minute.ToString() : XMLDate.Minute.ToString()) +
                  ":" +
                  (XMLDate.Second < 10 ? "0" +
                  XMLDate.Second.ToString() : XMLDate.Second.ToString()) +
                  ".000";
                excelDoc.Write("<Cell ss:StyleID=\"DateLiteral\">" +
                               "<Data ss:Type=\"DateTime\">");
```

```
                excelDoc.Write(XMLDatetoString);
                excelDoc.Write("</Data></Cell>");
                break;
              case "System.Boolean":
                excelDoc.Write("<Cell ss:StyleID=\"StringLiteral\">" +
                               "<Data ss:Type=\"String\">");
                excelDoc.Write(x[y].ToString());
                excelDoc.Write("</Data></Cell>");
                break;
              case "System.Int16":
              case "System.Int32":
              case "System.Int64":
              case "System.Byte":
                excelDoc.Write("<Cell ss:StyleID=\"Integer\">" +
                       "<Data ss:Type=\"Number\">");
                excelDoc.Write(x[y].ToString());
                excelDoc.Write("</Data></Cell>");
                break;
              case "System.Decimal":
              case "System.Double":
                excelDoc.Write("<Cell ss:StyleID=\"Decimal\">" +
                     "<Data ss:Type=\"Number\">");
                excelDoc.Write(x[y].ToString());
                excelDoc.Write("</Data></Cell>");
                break;
              case "System.DBNull":
                excelDoc.Write("<Cell ss:StyleID=\"StringLiteral\">" +
                     "<Data ss:Type=\"String\">");
                excelDoc.Write("");
                excelDoc.Write("</Data></Cell>");
                break;
              default:
                throw(new Exception(rowType.ToString() + " not handled."));
            }
          }
          excelDoc.Write("</Row>");
        }
        excelDoc.Write("</Table>");
        excelDoc.Write(" </Worksheet>");
        excelDoc.Write(endExcelXML);
        excelDoc.Close();
    }
```

Note

To see what generated, just pass the file name with a *.txt* extension. For Excel format, the file name will be *.xls*. For XML format, the file name will be *.xml*.

The Export Routine does have one side effect! (if anyone can figure out a solution to this, it would be greatly appreciated). The file is saved as an *.XLS* file, but it technically is still an XML file. This little nuance makes the file size larger then it really should be. A quick fix to this is to just do File Save As.... after the file has been exported. When

you do the Save As in Excel, it will reconstruct it as a "real" Excel file, and it will bring the file size down to what it should be.