

```

using System;
using System.Data;
using System.Data.SqlClient;
using System.Text;
using CustomersData;

/// <summary>
/// Author: Doug Streitenberger
/// Purpose: Create a convenient wallet sized printable output
/// of customer information. This web form builds a Microsoft Word ML
/// stream of the Customers Table.
/// Output consists of two columns of tables within the page
/// containing customer data.
/// Requires Microsoft Word 2003 or later to display properly. Prior versions
/// will display an XML stream in raw form.

/// Note: Page errors are handled by the InsertDefect method of
/// the CustDefects class within the CustomersData Namespace.
/// This method inserts an entry to table tblDefects and
/// includes the page on which the error occurred along with
/// the error description.
/// </summary>
public partial class DashboardDataXML : System.Web.UI.Page
{
    StringBuilder sb = new StringBuilder();

    protected void Page_Load(object sender, EventArgs e)
    {
        // Set Mime Type to Microsoft Word
        Response.ContentType = "application/msword";

        BindData();
        // Push the Data to the client.
        Response.Write(sb.ToString());
    }

    private void BindData()
    {
        try
        {
            // A DataTable containing all Customer records is returned
            // from CustomersEnum method of the CustData class
            // within the CustomersData Namespace.
            DataTable dt = CustData.CustomersEnum();

            // Initialize XML stream
            // XML Header. Schema is must.

```

```

sb.Append("<?xml version='1.0'?>");
sb.Append("<w:wordDocument xmlns:w='http://schemas" +
    ".microsoft.com/office/word/2003/wordml'>");

// Body of the document and header information
sb.Append("<w:body>");
sb.Append("<w:p><w:r><w:rPr><w:rFonts w:ascii='Arial' w:h-ansi='Arial' w:cs='Arial'/>" +
    "<w:sz w:val='15'/>" +
    "<w:b w:val='on' /><w:t>Customers</w:t></w:rPr></w:r></w:p>");
sb.Append("<w:p><w:r><w:rPr><w:rFonts w:ascii='Arial' w:h-ansi='Arial' w:cs='Arial'/>" +
    "<w:sz w:val='15'/>" +
    "<w:b w:val='on' /><w:t>As Of: " + DateTime.Now.ToShortDateString() +
    "</w:t></w:rPr></w:r></w:p>");

sb.Append("<w:p><w:r><w:t></w:t></w:r></w:p>");
sb.Append("<w:tbl>");

bool bLeftTable = true; // Used to determine if the table currently being built resides on the
// left or right hand side of the document.
int iMaxRows = 18; // The maximum number of rows per individual display table.
int iTableRowNum = 1; // The current row number within a given display table.
int iTotRowCount = dt.Rows.Count; // The number of rows returned from the Customers data table.
int iCurrentRowCount = 1; // The current row selected within the Customers data table

// Loop through the Customers DataTable
foreach (DataRow dr in dt.Rows)
{
    // If this is the first display table row
    // Build a new display table shell
    if (iTableRowNum == 1)
    {
        BuildDataTable(bLeftTable);
    }
    // Fill the Display table with Customer data rows
    FillDataTable(dr["CustomerID"].ToString(), dr["CompanyName"].ToString()
        , dr["ContactName"].ToString()
        , dr["Phone"].ToString(), dr["Fax"].ToString());

    // If the number of display rows have reached the maximum
    // allowed display rows per table
    // Build a stream to close the display table
    if ((iTableRowNum == iMaxRows) || (iCurrentRowCount == iTotRowCount))
    {
        CloseDataTable(bLeftTable);
        // If there are more Customer Records to process
        if (iCurrentRowCount < iTotRowCount)
        {

```

```

        if (bLeftTable)
        {
            // Insert a Spacer cell between the columns
            // Flip To Right Display Table
            SpacerCell();
            bLeftTable = false;
        }
        else
        {
            // Flip To Left Display Table
            bLeftTable = true;
        }
        // Set the Display Table Row Number = 0
        iTableRowNum = 0;
    }

    // Increment the current row number within a given display table.
    // Increment the current row selected within the Customers data table
    iTableRowNum += 1;
    iCurrentRowCount += 1;
}
// After we're finished with data, We need to close out display tables
// properly to conform to XML standards (well formed)
if (bLeftTable)
{
    // If current display table on left hand side of document
    // Insert spacer, and an empty table on right side of document.
    sb.Append("<w:tc><w:tcPr><w:tcW w:w='200' w:type='dxa' /></w:tcPr><w:p><w:r></w:r></w:p></w:tc>");
    sb.Append("<w:tc><w:tcPr><w:tcW w:w='5635' w:type='dxa' /></w:tcPr><w:p><w:r></w:r></w:p></w:tc>");
    sb.Append("</w:tr>");
}
sb.Append("</w:tbl>");
// Finally, set document attributes and close out the document
sb.Append("<w:sectPr><w:pgSz w:w='12240' w:h='15840' w:orient='portrait' w:code='1' />" +
    "<w:pgMar w:top='720' w:right='360' w:bottom='720' w:left='360' w:header='720' w:footer='720' " +
    "<w:gutter='0' /><w:cols w:space='720' /></w:sectPr>");
sb.Append("</w:body>");
sb.Append("</w:wordDocument>");
}
catch (Exception ex)
{
    CustDefects objDefects = new CustDefects();
    objDefects.InsertDefect(Request.ServerVariables["SCRIPT_NAME"], ex.ToString());
    objDefects = null;

    // if error occurred somewhere in the creation of the document,
    // set the stringbuilder stream to '' and send an error message to the screen

```

```

        sb.Remove(0, sb.Length);
        sb.Append("Error processing Compact Personnel Roster");
    }
}

/// <summary>
/// Formats table cell entries
/// XML cannot display the '&','<','>' characters...convert to HTML
/// </summary>
/// <param name="sCellValue">Current data cell being processed</param>
/// <returns></returns>
private string CellValue(string sCellValue)
{
    string sSetCellValue = sCellValue.Trim();
    sSetCellValue = sSetCellValue.Replace("&", "&amp;").Trim();
    sSetCellValue = sSetCellValue.Replace("<", "&lt;").Trim();
    sSetCellValue = sSetCellValue.Replace(">", "&gt;").Trim();
    // If the length of the data = 0, return 'N/A'
    if (sSetCellValue.Length == 0)
    {
        sSetCellValue = "N/A";
    }
    return sSetCellValue;
}

/// <summary>
/// Build an XML stream of data cells for each row being processed.
/// </summary>
/// <param name="sCustomerID">Customer ID</param>
/// <param name="sCompanyName">Company Name</param>
/// <param name="sContactName">Contact Name</param>
/// <param name="sPhone">Phone Number</param>
/// <param name="sFax">Fax Number</param>
private void FillDataTable(string sCustomerID, string sCompanyName, string sContactName,
    string sPhone, string sFax)
{
    sb.Append("<w:tr>");
    sb.Append("<w:tc><w:tcPr><w:tcW w:w='700' w:type='dxa' /><w:vAlign w:val='top' /></w:tcPr><w:p><w:r>" +
        "<w:rPr><w:rFonts w:ascii='Arial' w:h-ansi='Arial' w:cs='Arial' />" +
        "<w:sz w:val='10' /></w:rPr>");
    sb.Append("<w:t>" + CellValue(sCustomerID) + "</w:t>");
    sb.Append("</w:r></w:p></w:tc>");

    sb.Append("<w:tc><w:tcPr><w:tcW w:w='1785' w:type='dxa' /><w:vAlign w:val='top' /></w:tcPr><w:p><w:r>" +
        "<w:rPr><w:rFonts w:ascii='Arial' w:h-ansi='Arial' w:cs='Arial' />" +
        "<w:sz w:val='10' /></w:rPr>");
    sb.Append("<w:t>" + CellValue(sCompanyName) + "</w:t>");
}

```

```

sb.Append("</w:r></w:p></w:tc>");

sb.Append("<w:tc><w:tcPr><w:tcW w:w='1350' w:type='dxa' /><w:vAlign w:val='top' /></w:tcPr><w:p><w:r>" +
    "<w:rPr><w:rFonts w:ascii='Arial' w:h-ansi='Arial' w:cs='Arial' />" +
    "<w:sz w:val='10' /></w:rPr>");
sb.Append("<w:t>" + CellValue(sContactName) + "</w:t>");
sb.Append("</w:r></w:p></w:tc>");

sb.Append("<w:tc><w:tcPr><w:tcW w:w='900' w:type='dxa' /><w:vAlign w:val='top' /></w:tcPr><w:p><w:r>" +
    "<w:rPr><w:rFonts w:ascii='Arial' w:h-ansi='Arial' w:cs='Arial' />" +
    "<w:sz w:val='10' /></w:rPr>");
sb.Append("<w:t>" + CellValue(sPhone) + "</w:t>");
sb.Append("</w:r></w:p></w:tc>");

sb.Append("<w:tc><w:tcPr><w:tcW w:w='900' w:type='dxa' /><w:vAlign w:val='top' /></w:tcPr><w:p><w:r>" +
    "<w:rPr><w:rFonts w:ascii='Arial' w:h-ansi='Arial' w:cs='Arial' />" +
    "<w:sz w:val='10' /></w:rPr>");
sb.Append("<w:t>" + CellValue(sFax) + "</w:t>");
sb.Append("</w:r></w:p></w:tc>");
sb.Append("</w:tr>");
}

/// <summary>
/// Build an XML stream initializing an empty display table, columns and column headings
/// </summary>
/// <param name="bIsLeftTable">Boolean - Determines if the current table layout is on the left
/// or right side of the document.</param>
private void BuildDataTable(bool bIsLeftTable)
{
    if (bIsLeftTable)
    {
        sb.Append("<w:tr>");
    }
    sb.Append("<w:tc><w:p><w:r>");
    sb.Append("<w:tbl>");
    sb.Append("<w:tblW w:w='5635' w:type='dxa' />");
    sb.Append("<w:tblPr>");
    sb.Append("<w:tblBorders>");
    sb.Append("<w:top w:val='single' w:sz='4' w:bdrwidth='10' w:space='0' w:color='000000' />");
    sb.Append("<w:left w:val='single' w:sz='4' w:bdrwidth='10' w:space='0' w:color='000000' />");
    sb.Append("<w:bottom w:val='single' w:sz='4' w:bdrwidth='10' w:space='0' w:color='000000' />");
    sb.Append("<w:right w:val='single' w:sz='4' w:bdrwidth='10' w:space='0' w:color='000000' />");
    sb.Append("<w:insideH w:val='single' w:sz='4' w:bdrwidth='10' w:space='0' w:color='000000' />");
    sb.Append("<w:insideV w:val='single' w:sz='4' w:bdrwidth='10' w:space='0' w:color='000000' />");
    sb.Append("</w:tblBorders>");
    sb.Append("</w:tblPr>");
    sb.Append("<w:tr><w:trPr><w:trHeight w:val='140' /></w:trPr>");
}

```

```

sb.Append("<w:tc><w:tcPr><w:tcW w:w='700' w:type='dxa' /><w:shd w:val='solid' w:color='E0E0E0' />" +
    "</w:tcPr><w:p><w:r><w:rPr><w:rFonts w:ascii='Arial' w:h-ansi='Arial' w:cs='Arial' />" +
    "<w:sz w:val='10' /><w:b w:val='on' /></w:rPr>");
sb.Append("<w:t>Customer ID</w:t>");
sb.Append("</w:r></w:p></w:tc>");

sb.Append("<w:tc><w:tcPr><w:tcW w:w='1785' w:type='dxa' /><w:shd w:val='solid' w:color='E0E0E0' />" +
    "</w:tcPr><w:p><w:r><w:rPr><w:rFonts w:ascii='Arial' w:h-ansi='Arial' w:cs='Arial' />" +
    "<w:sz w:val='10' /><w:b w:val='on' /></w:rPr>");
sb.Append("<w:t>Company Name</w:t>");
sb.Append("</w:r></w:p></w:tc>");

sb.Append("<w:tc><w:tcPr><w:tcW w:w='1350' w:type='dxa' /><w:shd w:val='solid' w:color='E0E0E0' />" +
    "</w:tcPr><w:p><w:r><w:rPr><w:rFonts w:ascii='Arial' w:h-ansi='Arial' w:cs='Arial' />" +
    "<w:sz w:val='10' /><w:b w:val='on' /></w:rPr>");
sb.Append("<w:t>Contact Name</w:t>");
sb.Append("</w:r></w:p></w:tc>");

sb.Append("<w:tc><w:tcPr><w:tcW w:w='900' w:type='dxa' /><w:shd w:val='solid' w:color='E0E0E0' />" +
    "</w:tcPr><w:p><w:r><w:rPr><w:rFonts w:ascii='Arial' w:h-ansi='Arial' w:cs='Arial' />" +
    "<w:sz w:val='10' /><w:b w:val='on' /></w:rPr>");
sb.Append("<w:t>Phone</w:t>");
sb.Append("</w:r></w:p></w:tc>");

sb.Append("<w:tc><w:tcPr><w:tcW w:w='900' w:type='dxa' /><w:shd w:val='solid' w:color='E0E0E0' />" +
    "</w:tcPr><w:p><w:r><w:rPr><w:rFonts w:ascii='Arial' w:h-ansi='Arial' w:cs='Arial' />" +
    "<w:sz w:val='10' /><w:b w:val='on' /></w:rPr>");
sb.Append("<w:t>Fax</w:t>");
sb.Append("</w:r></w:p></w:tc>");
sb.Append("</w:tr>");
}

/// <summary>
/// Build an XML stream containing a single table cell to provide spacing between display tables.
/// </summary>
private void SpacerCell()
{
    sb.Append("<w:tc><w:tcPr><w:tcW w:w='200' w:type='dxa' /></w:tcPr><w:p><w:r></w:r></w:p></w:tc>");
}

/// <summary>
/// Close out the current display table to conform with XML standards (well formed).
/// </summary>
/// <param name="bIsLeftTable">Boolean - Determines if the current table layout is on the left
/// or right side of the document. If current table is on the right side of the document,
/// close out inner and outer tables; insert blank line; start a new table for customer
/// data on left side of document.</param>

```

```

private void CloseDataTable(bool bIsLeftTable)
{
    sb.Append("<w:tr>");
    sb.Append("<w:tc><w:tcPr><w:gridSpan w:val='5' /></w:tcPr><w:p><w:r>");
    sb.Append("<w:tbl>");
    sb.Append("<w:tr><w:trPr><w:trHeight w:val='140' /></w:trPr>");
    sb.Append("<w:tc><w:tcPr><w:tcW w:w='5635' w:type='dxa' /><w:vAlign w:val='center' />" +
        "<w:shd w:val='solid' w:color='E0E0E0' /></w:tcPr><w:p><w:pPr><w:jc w:val='center' /></w:pPr><w:r>" +
        "<w:rPr><w:rFonts w:ascii='Arial' w:h-ansi='Arial' w:cs='Arial' />" +
        "<w:sz w:val='10' /><w:b w:val='on' /></w:rPr>");
    sb.Append("<w:t>Sensitive Information. Protect accordingly</w:t>");
    sb.Append("</w:r></w:p></w:tc>");
    sb.Append("</w:tr>");
    sb.Append("</w:tbl>");
    sb.Append("</w:r></w:p></w:tc>");
    sb.Append("</w:tr>");

    if (bIsLeftTable)
    {
        sb.Append("</w:tbl>");
        sb.Append("</w:r></w:p></w:tc>");
    }
    else
    {
        sb.Append("</w:tbl>");
        sb.Append("</w:r></w:p></w:tc></w:tr>");
        sb.Append("</w:tbl><w:p><w:r><w:t> </w:t></w:r></w:p><w:p><w:r><w:t> </w:t></w:r></w:p><w:tbl>");
    }
}
}

```