

Excel to DataTable Sample

Intro

Extract Excel data into a DataTable

The source Workbook contains a listing of Employees for the fictional company AdventureWorks. The actual data is marked with two Named Ranges, "EmployeesHeadings" and "EmployeesData", which will make it easier to import using ExcelWriter. This data is then used to populate a DataGridView

Code

```
public class ExcelToDataTable
{
    // The indexes of the columns which contain dates
    protected int[] dateColumnIdxs;

    /// <summary>
    /// Open an existing Excel workbook, navigate to the "Employees"
    /// named range, and extract the named range data into a DataTable.
    /// </summary>
    public void PopulateDataTable()
    {
        // Create an instance of ExcelApplication
        ExcelApplication xlw = new ExcelApplication();

        // Open the Workbook containing the data
        string workbookPath = @"..\..\ExcelData\ExcelToDataTable.xlsx";
        Workbook wb = xlw.Open(workbookPath);

        //Get the first Area of the Named Range "employees", which contains the
        //column headings and data

        Range empRange = wb.GetNamedRange("Employees");
        Area empArea = empRange.Areas[0];

        // Populate the array of columns which are dates, in this case the 5th
column
        this.dateColumnIdxs = new int[] {4};

        // Call conversion method below to convert Area to DataTable
        DataTable empTable =
            AreaToDataTable(empArea, true, this.dateColumnIdxs, "Employees");
    }

    /// <summary>
    /// ExcelApplication has the ability to read data from Excel workbooks.
    /// This method Extracts data from an ExcelApplication Area object into
    /// a System.Data.DataTable object.
    /// </summary>
    /// <param name="a">Area from which to extract data</param>
    /// <param name="hasHeaderRow">True if first row of area contains column
headers</param>
```

```

    /// <param name="dateColumnIndexes">Columns that should be handled as Date
values</param>
    /// <param name="tableName">Name for the new DataTable</param>
    /// <returns>DataTable containing data from the specified Area</returns>
    private DataTable AreaToDataTable(Area a, bool hasHeaderRow, int[]
dateColumnIndexes, string tableName)
    {
        // Set count and index variables
        int dataRowCount;
        int firstDataRowIdx;
        int columnCount = a.ColumnCount;

        if (hasHeaderRow == true)
        {
            firstDataRowIdx = 1;
            dataRowCount = a.RowCount - 1;
        }
        else
        {
            firstDataRowIdx = 0;
            dataRowCount = a.RowCount;
        }
        // Create DataTable
        DataTable table = new DataTable(tableName);

        // Add columns to DataTable
        for (int colIdx = 0; colIdx < columnCount; colIdx++)
        {
            string columnName;

            if (hasHeaderRow == true)
            {
                // Get column name from cell value
                columnName = (string)a[0, colIdx].Value;
            }
            else
            {
                // Use a generic column name
                columnName = "Column" + colIdx;
            }

            // Insert column into DataTable
            table.Columns.Add(new DataColumn(columnName));
        }

        //Populate rows of DataTable
        // For each row in Area
        for (int rowIdx = 0; rowIdx < dataRowCount; rowIdx++)
        {
            // Add row to DataTable
            DataRow row = table.NewRow();
            table.Rows.Add(row);

            // For each column
            for (int colIdx = 0; colIdx < columnCount; colIdx++)
            {
                // Get value from cell
                object val = a[firstDataRowIdx + rowIdx, colIdx].Value;

```

```

.NET date.
convert.)

        // If current column is in the array of date columns, convert to a
        //(Excel dates are stored in a serial format which we must

        if (Array.IndexOf(dateColumnIndexes, colIdx) != -1)
        {
            // Convert to a .NET date using conversion method below
            DateTime dt = ExcelSerialDateToDateTime(Convert.ToInt32(val));

            // Add date to DataTable
            row[colIdx] = dt;
        }
        else
        {
            // Otherwise, just add cell value to DataTable
            row[colIdx] = val;
        }
    }

    // Return DataTable
    return table;
}

/// <summary>
/// Internally, Microsoft Excel stores dates as the number of days
/// that have passed since the epoch. For example, November 1, 2005
/// is stored as 38657. Use this method to convert
/// an Excel serial date to a .NET DateTime object.
/// </summary>
/// <param name="serialDate">The Excel serial date</param>
/// <returns>DateTime object for the specified serial date</returns>
private DateTime ExcelSerialDateToDateTime(int serialDate)
{
    if(serialDate == 60)
    {
        return new DateTime(1900, 2, 29);
    }
    else if(serialDate < 60)
    {
        serialDate++;
    }

    return new DateTime(1900, 1, 1).AddDays(serialDate - 2);
}

```

```
}
```

Downloads

- Template file: [ExcelToDataTable.xlsx](#)