LEARNING epplus

#epplus

Table of Contents

About

Chapter 1: Getting started with epplus

Remarks

EPPlus is a .NET library that reads and writes Excel 2007/2010/2013 files using the Open Office Xml format (xlsx).

EPPlus supports:

- Cell Ranges
- Cell styling (Border, Color, Fill, Font, Number, Alignments)
- Charts
- Pictures
- Shapes
- Comments
- Tables
- Protection
- Encryption
- Pivot tables
- Data validation
- Conditional formatting
- VBA
- Formula calculation

Versions

Version	Release Date
First Release	2009-11-30
2.5.0.1	2010-01-25
2.6.0.1	2010-03-23
2.7.0.1	2010-06-17
2.8.0.2	2010-11-15
2.9.0.1	2011-05-31
3.0.0.2	2012-01-31
3.1	2012-04-11
4.0.5	2016-01-08
4.1	2016-07-14

Examples

Installation

Download the files from CodePlex and add them to the project.

Or install the files with the Package Manager.

```
PM> Install-Package EPPlus
```

Getting started

```
//Create a new ExcelPackage
using (ExcelPackage excelPackage = new ExcelPackage())
{
   //Set some properties of the Excel document
   excelPackage.Workbook.Properties.Author = "VDWWD";
   excelPackage.Workbook.Properties.Title = "Title of Document";
   excelPackage.Workbook.Properties.Subject = "EPPlus demo export data";
   excelPackage.Workbook.Properties.Created = DateTime.Now;

    //Create the WorkSheet
    ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Sheet 1");

    //Add some text to cell A1
    worksheet.Cells["A1"].Value = "My first EPPlus spreadsheet!";
    //You could also use [line, column] notation:
    worksheet.Cells[1,2].Value = "This is cell B1!";

    //Save your file
    FileInfo fi = new FileInfo(@"Path\To\Your\File.xlsx");
    excelPackage.SaveAs(fi);
}

//Opening an existing Excel file
FileInfo fi = new FileInfo(@"Path\To\Your\File.xlsx");
using (ExcelPackage excelPackage = new ExcelPackage(fi))
{
    //Get a WorkSheet by index. Note that EPPlus indexes are base 1, not base 0!
    ExcelWorksheet firstWorksheet = excelPackage.Workbook.Worksheets[1];

    //Get a WorkSheet by name. If the worksheet doesn't exist, throw an exeption
    ExcelWorksheet namedWorksheet = excelPackage.Workbook.Worksheets["SomeWorksheet"];

    //If you don't know if a worksheet exists, you could use LINQ,
    //So it doesn't throw an exception, but return null in case it doesn't find it
    ExcelWorksheet anotherWorksheet =
        excelPackage.Workbook.Worksheets.FirstOrDefault(x=>x.Name=="SomeWorksheet");

    //Get the content from cells A1 and B1 as string, in two different notations
    string valA1 = firstWorksheet.Cells["A1"].Value.ToString();
    string valB1 = firstWorksheet.Cells[1,2].Value.ToString();

    //Save your file
    excelPackage.Save();
}
```

Read Getting started with epplus online: https://riptutorial.com/epplus/topic/8070/getting-started-with-epplus

Chapter 2: Append data to existing document

Introduction

How to append data to an already existing Excel document.

Examples

Appending data

```
//the path of the file
string filePath = "C:\\ExcelDemo.xlsx";

//or if you use asp.net, get the relative path
filePath = Server.MapPath("ExcelDemo.xlsx");

//create a fileinfo object of an excel file on the disk
FileInfo file = new FileInfo(filePath);

//create a new Excel package from the file
using (ExcelPackage excelPackage = new ExcelPackage(file))
{
    //create an instance of the the first sheet in the loaded file
    ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets[1];

    //add some data
    worksheet.Cells[4, 1].Value = "Added data in Cell A4";
    worksheet.Cells[4, 2].Value = "Added data in Cell B4";

    //save the changes
    excelPackage.Save();
}
```

Read Append data to existing document online: https://riptutorial.com/epplus/topic/8596/append-data-to-existing-document

Chapter 3: Columns and Rows

Introduction

This topic contains information about working with columns and rows, like resizing, hiding, autofit

Examples

Autofit columns

```
//Make all text fit the cells
worksheet.Cells[worksheet.Dimension.Address].AutoFitColumns();

//Autofit with minimum size for the column.
double minimumSize = 10;
worksheet.Cells[worksheet.Dimension.Address].AutoFitColumns(minimumSize);

//Autofit with minimum and maximum size for the column.
double maximumSize = 50;
worksheet.Cells[worksheet.Dimension.Address].AutoFitColumns(minimumSize, maximumSize);

//optional use this to make all columms just a bit wider, text would sometimes still overflow
after AutoFitColumns().
for (int col = 1; col <= worksheet.Dimension.End.Column; col++)
{
    worksheet.Column(col).Width = worksheet.Column(col).Width + 1;
}
```

Hide columns and rows

```
//Hide column "A"
worksheet.Column(1).Hidden = true;

//Hide row 1
worksheet.Row(1).Hidden = true;
```

Resizing rows and columns

```
//Set the row "A" height to 15
double rowHeight = 15;
worksheet.Row(1).Height = rowHeight;

//Set the column 1 width to 50
double columnWidth = 50;
worksheet.Column(1).Width = columnWidth;
```

When Bestfit is set to true, the column will grow wider when a user inputs numbers in a cell

```
worksheet.Column(1).BestFit = true;
```

Copy columns or rows

```
workSheet.Cells[1,5,100,5].Copy(workSheet.Cells[1,2,100,2]);
```

Copies column 5 into column 2 Basically Source.Copy(Destination)

This would only copy the first 100 rows.

```
Cells[RowStart, ColumnStart, RowEnd, ColumnEnd ]
is the format so to copy a row into another row you would just switch the indexes accordingly
```

Read Columns and Rows online: https://riptutorial.com/epplus/topic/8766/columns-and-rows

Chapter 4: Creating charts

Introduction

How to create charts with EPPlus

Examples

Pie Chart

```
//create a new ExcelPackage
using (ExcelPackage excelPackage = new ExcelPackage())
{
    //create a WorkSheet
    ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Sheet 1");

    //fill cell data with a loop, note that row and column indexes start at 1
    Random rnd = new Random();
    for (int i = 1; i <= 10; i++)
    {
        worksheet.Cells[1, i].Value = "Value " + i;
        worksheet.Cells[2, i].Value = rnd.Next(5, 15);
    }

    //create a new piechart of type Pie3D
    ExcelPieChart pieChart = worksheet.Drawings.AddChart("pieChart", eChartType.Pie3D) as
ExcelPieChart;

    //set the title
    pieChart.Title.Text = "PieChart Example";

    //select the ranges for the pie. First the values, then the header range
    pieChart.Series.Add(ExcelRange.GetAddress(2, 1, 2, 10), ExcelRange.GetAddress(1, 1, 1,
10));

    //position of the legend
    pieChart.Legend.Position = eLegendPosition.Bottom;

    //show the percentages in the pie
    pieChart.DataLabel.ShowPercent = true;

    //size of the chart
    pieChart.SetSize(500, 400);

    //add the chart at cell C5
    pieChart.SetPosition(4, 0, 2, 0);
}
```

Line Chart

```
//create a new ExcelPackage
using (ExcelPackage excelPackage = new ExcelPackage())
{
    //create a WorkSheet
    ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Sheet 1");

    //fill cell data with a loop, note that row and column indexes start at 1
    Random rnd = new Random();
    for (int i = 2; i <= 11; i++)
    {
        worksheet.Cells[1, i].Value = "Value " + (i - 1);
        worksheet.Cells[2, i].Value = rnd.Next(5, 25);
        worksheet.Cells[3, i].Value = rnd.Next(5, 25);
    }
    worksheet.Cells[2, 1].Value = "Age 1";
    worksheet.Cells[3, 1].Value = "Age 2";

    //create a new piechart of type Line
    ExcelLineChart lineChart = worksheet.Drawings.AddChart("lineChart", eChartType.Line) as
ExcelLineChart;

    //set the title
    lineChart.Title.Text = "LineChart Example";

    //create the ranges for the chart
    var rangeLabel = worksheet.Cells["B1:K1"];
    var range1 = worksheet.Cells["B2:K2"];
    var range2 = worksheet.Cells["B3:K3"];

    //add the ranges to the chart
    lineChart.Series.Add(range1, rangeLabel);
    lineChart.Series.Add(range2, rangeLabel);

    //set the names of the legend
    lineChart.Series[0].Header = worksheet.Cells["A2"].Value.ToString();
    lineChart.Series[1].Header = worksheet.Cells["A3"].Value.ToString();

    //position of the legend
    lineChart.Legend.Position = eLegendPosition.Right;

    //size of the chart
    lineChart.SetSize(600, 300);

    //add the chart at cell B6
    lineChart.SetPosition(5, 0, 1, 0);
}
```

Read Creating charts online: https://riptutorial.com/epplus/topic/8286/creating-charts

Chapter 5: Creating formulas and calculate ranges

Introduction

Basic examples of how to create cells with a formula for calculations within the Excel sheet.

Examples

Add formulas to a cell

```
//set the total value of cells in range A1 - A25 into A27
worksheet.Cells["A27"].Formula = "=SUM(A1:A25)";

//set the number of cells with content in range C1 - C25 into C27
worksheet.Cells["C27"].Formula = "=COUNT(C1:C25)";

//fill column K with the sum of each row, range A - J
for (int i = 1; i <= 25; i++)
{
    var cell = worksheet.Cells[i, 12];
    cell.Formula = "=SUM(" + worksheet.Cells[i, 1].Address + ":" + worksheet.Cells[i,
10].Address + ")";
}

//calculate the quartile of range E1 - E25 into E27
worksheet.Cells[27, 5].Formula = "=QUARTILE(E1:E25,1)";
```

Formula with multiple sheets

```
//set the total value of all cells in Sheet 2 into G27
worksheet.Cells["G27"].Formula = "=SUM('" + worksheet2.Name + "'!" +
worksheet2.Dimension.Start.Address + ":" + worksheet2.Dimension.End.Address + ")";

//set the number of cells with content in Sheet 2, range C1 - C25 into I27
worksheet.Cells["I27"].Formula = "=COUNT('" + excelPackage.Workbook.Worksheets[2].Name + "'!"
+ excelPackage.Workbook.Worksheets[2].Cells["A1:B25"] + ")";
```

Manual calculation

If you use formulas, Excel will ask you to save the file every time, even if there were no changes made. To prevent this behaviour you can set the calculation mode to manual.

```
excelPackage.Workbook.CalcMode = ExcelCalcMode.Manual;

//fill the sheet with data and set the formulas

excelPackage.Workbook.Calculate();
```

Complete example with formulas

```
//create a new ExcelPackage
using (ExcelPackage excelPackage = new ExcelPackage())
{
    //create 2 WorkSheets
    ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Sheet 1");
    ExcelWorksheet worksheet2 = excelPackage.Workbook.Worksheets.Add("Sheet 2");

    //set the calculation mode to manual
    excelPackage.Workbook.CalcMode = ExcelCalcMode.Manual;

    //fill cell data with a loop, note that row and column indexes start at 1
    for (int i = 1; i <= 25; i++)
    {
        for (int j = 1; j <= 10; j++)
        {
            worksheet.Cells[i, j].Value = (i + j) - 1;
            worksheet2.Cells[i, j].Value = (i + j) - 1;
        }
    }

    //set the total value of cells in range A1 - A25 into A27
    worksheet.Cells["A27"].Formula = "=SUM(A1:A25)";

    //set the number of cells with content in range C1 - C25 into C27
    worksheet.Cells["C27"].Formula = "=COUNT(C1:C25)";

    //fill column K with the sum of each row, range A - J
    for (int i = 1; i <= 25; i++)
    {
        var cell = worksheet.Cells[i, 12];
        cell.Formula = "=SUM(" + worksheet.Cells[i, 1].Address + ":" + worksheet.Cells[i,
10].Address + ")";
    }

    //calculate the quartile of range E1 - E25 into E27
    worksheet.Cells[27, 5].Formula = "=QUARTILE(E1:E25,1)";

    //set the total value of all cells in Sheet 2 into G27
    worksheet.Cells["G27"].Formula = "=SUM('" + worksheet2.Name + "'!" +
worksheet2.Dimension.Start.Address + ":" + worksheet2.Dimension.End.Address + ")";

    //set the number of cells with content in Sheet 2, range C1 - C25 into I27
    worksheet.Cells["I27"].Formula = "=COUNT('" + excelPackage.Workbook.Worksheets[2].Name +
"'!" + excelPackage.Workbook.Worksheets[2].Cells["A1:B25"] + ")";

    //calculate all the values of the formulas in the Excel file
    excelPackage.Workbook.Calculate();

    //Save the file
    FileInfo fi = new FileInfo("FormulaExample.xlsx");
    excelPackage.SaveAs(fi);
}
```

Read Creating formulas and calculate ranges online:
https://riptutorial.com/epplus/topic/8227/creating-formulas-and-calculate-ranges

Chapter 6: Filling the document with data

Introduction

How to fill your created Excel sheet with data from different sources.

Examples

Fill with a DataTable

```
//create a new ExcelPackage
using (ExcelPackage excelPackage = new ExcelPackage())
{
    //create a datatable
    DataTable dataTable = new DataTable();

    //add three colums to the datatable
    dataTable.Columns.Add("ID", typeof(int));
    dataTable.Columns.Add("Type", typeof(string));
    dataTable.Columns.Add("Name", typeof(string));

    //add some rows
    dataTable.Rows.Add(0, "Country", "Netherlands");
    dataTable.Rows.Add(1, "Country", "Japan");
    dataTable.Rows.Add(2, "Country", "America");
    dataTable.Rows.Add(3, "State", "Gelderland");
    dataTable.Rows.Add(4, "State", "Texas");
    dataTable.Rows.Add(5, "State", "Echizen");
    dataTable.Rows.Add(6, "City", "Amsterdam");
    dataTable.Rows.Add(7, "City", "Tokyo");
    dataTable.Rows.Add(8, "City", "New York");

    //create a WorkSheet
    ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Sheet 1");

    //add all the content from the DataTable, starting at cell A1
    worksheet.Cells["A1"].LoadFromDataTable(dataTable, true);
}
```

Fill with a DataTable from an SQL query or Stored Procedure

```
//create a new ExcelPackage
using (ExcelPackage excelPackage = new ExcelPackage())
{
    //the query or stored procedure name for the database
    string sqlQuery = "SELECT * FROM myTable";

    //create a datatable
    DataTable dataTable = loadExternalDataSet(sqlQuery);

    //create a WorkSheet
    ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Sheet 1");
    //add all the content from the DataTable, starting at cell A1
    worksheet.Cells["A1"].LoadFromDataTable(dataTable, true);
}

//method for retrieving data from the database and return it as a datatable
public static DataTable loadExternalDataSet(string sqlQuery)
{
    DataTable dt = new DataTable();

    using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["myConnStr"].ConnectionString))
    using (SqlDataAdapter adapter = new SqlDataAdapter(sqlQuery, connection))
    {
        try
        {
            adapter.Fill(dt);
        }
        catch
        {
        }
    }

    return dt;
}
```

Manually fill cells

Fill some cells with text.

```
worksheet.Cells["A1"].Value = "Lorem ipsum";
worksheet.Cells["B2"].Value = "dolor sit amet";
worksheet.Cells["C3"].Value = "consectetur adipiscing";
worksheet.Cells["D4"].Value = "elit sed do eiusmod";

worksheet.Cells["E5"].Value = 12345;
worksheet.Cells["F6"].Value = DateTime.Now;
```

Fill cell data with a loop, note that row and column indexes start at 1

```
for (int i = 1; i <= 30; i++)
{
    for (int j = 1; j <= 15; j++)
    {
        worksheet.Cells[i, j].Value = "Row " + i + ", Column " + j;
    }
}
```

Fill from collection

```
//create a new ExcelPackage
using (ExcelPackage excelPackage = new ExcelPackage())
{
    //create a WorkSheet
    ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Sheet 1");

    //create a new list with books
    List<Book> books = new List<Book>();

    //add some books to the list
    for (int i = 0; i < 10; i++)
    {
        Book b = new Book();

        b.id = i;
        b.name = "Name " + i;
        b.category = "Category " + i;
        b.date = DateTime.Now.AddDays(i).AddHours(i);

        books.Add(b);
    }

    //add all the content from the List<Book> collection, starting at cell A1
    worksheet.Cells["A1"].LoadFromCollection(books);
}
```

Read Filling the document with data online: https://riptutorial.com/epplus/topic/8223/filling-the-document-with-data

Chapter 7: Formatting values

Introduction

How to get the desired formatting of DateTime and Numeric values.

Examples

Number formatting

```
//integer (not really needed unless you need to round numbers, Excel with use default cell
properties)
worksheet.Cells["A1:A25"].Style.Numberformat.Format = "0";

//integer without displaying the number 0 in the cell
worksheet.Cells["A1:A25"].Style.Numberformat.Format = "#";

//number with 1 decimal place
worksheet.Cells["A1:A25"].Style.Numberformat.Format = "0.0";

//number with 2 decimal places
worksheet.Cells["A1:A25"].Style.Numberformat.Format = "0.00";

//number with 2 decimal places and thousand separator
worksheet.Cells["A1:A25"].Style.Numberformat.Format = "#,##0.00";

//number with 2 decimal places and thousand separator and money symbol
worksheet.Cells["A1:A25"].Style.Numberformat.Format = "€#,##0.00";

//percentage (1 = 100%, 0.01 = 1%)
worksheet.Cells["A1:A25"].Style.Numberformat.Format = "0%";
```

Date formatting

```
//default DateTime patterns
worksheet.Cells["A1:A25"].Style.Numberformat.Format =
DateTimeFormatInfo.CurrentInfo.ShortDatePattern;

//custom DateTime patters
worksheet.Cells["A1:A25"].Style.Numberformat.Format = "dd-MM-yyyy HH:mm";

//or overwrite the patterns in the CurrentThread with your own
Thread.CurrentThread.CurrentCulture = new CultureInfo("nl-NL")
{
    DateTimeFormat = { YearMonthPattern = "MMM yy" }
};
worksheet.Cells["A1:A25"].Style.Numberformat.Format =
DateTimeFormatInfo.CurrentInfo.YearMonthPattern;
```

Text Format

```
worksheet.Cells["A1:A25"].Style.Numberformat.Format = "@";
```

Read Formatting values online: https://riptutorial.com/epplus/topic/8080/formatting-values

Chapter 8: Importing data from existing file

Introduction

How to import data from an existing Excel or CSV file.

Examples

Import data from Excel file

```
//create a list to hold all the values
List<string> excelData = new List<string>();

//read the Excel file as byte array
byte[] bin = File.ReadAllBytes("C:\\ExcelDemo.xlsx");

//or if you use asp.net, get the relative path
byte[] bin = File.ReadAllBytes(Server.MapPath("ExcelDemo.xlsx"));

//create a new Excel package in a memorystream
using (MemoryStream stream = new MemoryStream(bin))
using (ExcelPackage excelPackage = new ExcelPackage(stream))
{
    //loop all worksheets
    foreach (ExcelWorksheet worksheet in excelPackage.Workbook.Worksheets)
    {
        //loop all rows
        for (int i = worksheet.Dimension.Start.Row; i <= worksheet.Dimension.End.Row; i++)
        {
            //loop all columns in a row
            for (int j = worksheet.Dimension.Start.Column; j <=
worksheet.Dimension.End.Column; j++)
            {
                //add the cell data to the List
                if (worksheet.Cells[i, j].Value != null)
                {
                    excelData.Add(worksheet.Cells[i, j].Value.ToString());
                }
            }
        }
    }
}
```

Import data from CSV file

```
//set the formatting options
ExcelTextFormat format = new ExcelTextFormat();
format.Delimiter = ';';
format.Culture = new CultureInfo(Thread.CurrentThread.CurrentCulture.ToString());
format.Culture.DateTimeFormat.ShortDatePattern = "dd-mm-yyyy";
format.Encoding = new UTF8Encoding();

//read the CSV file from disk
```

```
FileInfo file = new FileInfo("C:\\CSVDemo.csv");

//or if you use asp.net, get the relative path
FileInfo file = new FileInfo(Server.MapPath("CSVDemo.csv"));

//create a new Excel package
using (ExcelPackage excelPackage = new ExcelPackage())
{
    //create a WorkSheet
    ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Sheet 1");

    //load the CSV data into cell A1
    worksheet.Cells["A1"].LoadFromText(file, format);
}
```

Import data from Excel file with FileUpload Control

```
//check if there is actually a file being uploaded
if (FileUpload1.HasFile)
{
    //load the uploaded file into the memorystream
    using (MemoryStream stream = new MemoryStream(FileUpload1.FileBytes))
    using (ExcelPackage excelPackage = new ExcelPackage(stream))
    {
        //loop all worksheets
        foreach (ExcelWorksheet worksheet in excelPackage.Workbook.Worksheets)
        {
            //loop all rows
            for (int i = worksheet.Dimension.Start.Row; i <= worksheet.Dimension.End.Row; i++)
            {
                //loop all columns in a row
                for (int j = worksheet.Dimension.Start.Column; j <=
worksheet.Dimension.End.Column; j++)
                {
                    //add the cell data to the List
                    if (worksheet.Cells[i, j].Value != null)
                    {
                        excelData.Add(worksheet.Cells[i, j].Value.ToString());
                    }
                }
            }
        }
    }
}
```

Create a DataTable from Excel File

```
public static DataTable ExcelPackageToDataTable(ExcelPackage excelPackage)
{
    DataTable dt = new DataTable();
    ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets[1];

    //check if the worksheet is completely empty
    if (worksheet.Dimension == null)
    {
        return dt;
    }
    //create a list to hold the column names
    List<string> columnNames = new List<string>();

    //needed to keep track of empty column headers
    int currentColumn = 1;

    //loop all columns in the sheet and add them to the datatable
    foreach (var cell in worksheet.Cells[1, 1, 1, worksheet.Dimension.End.Column])
    {
        string columnName = cell.Text.Trim();

        //check if the previous header was empty and add it if it was
        if (cell.Start.Column != currentColumn)
        {
            columnNames.Add("Header_" + currentColumn);
            dt.Columns.Add("Header_" + currentColumn);
            currentColumn++;
        }

        //add the column name to the list to count the duplicates
        columnNames.Add(columnName);

        //count the duplicate column names and make them unique to avoid the exception
        //A column named 'Name' already belongs to this DataTable
        int occurrences = columnNames.Count(x => x.Equals(columnName));
        if (occurrences > 1)
        {
            columnName = columnName + "_" + occurrences;
        }

        //add the column to the datatable
        dt.Columns.Add(columnName);

        currentColumn++;
    }

    //start adding the contents of the excel file to the datatable
    for (int i = 2; i <= worksheet.Dimension.End.Row; i++)
    {
        var row = worksheet.Cells[i, 1, i, worksheet.Dimension.End.Column];
        DataRow newRow = dt.NewRow();

        //loop all cells in the row
        foreach (var cell in row)
        {
            newRow[cell.Start.Column - 1] = cell.Text;
        }

        dt.Rows.Add(newRow);
    }

    return dt;
}
```

Read Importing data from existing file online: https://riptutorial.com/epplus/topic/8290/importing-data-from-existing-file

Chapter 9: Merge Cells

Introduction

How to merge cells

Examples

Merging cells

```
//By range address
worksheet.Cells["A1:B5"].Merge = true;

//By indexes
worksheet.Cells[1,1,5,2].Merge = true;
```

Read Merge Cells online: https://riptutorial.com/epplus/topic/8728/merge-cells

Chapter 10: Pivot Table

Introduction

Pivot table is one kind of interactive table, which can be used to calculate data, such as get sum or count data. Also, users can change pivot table layout for analyzing data with different ways or reassign row/column label. Every time users change layout, data will be recalculated in pivot table.

Examples

Creating a Pivot Table

```
//create a new ExcelPackage
using (ExcelPackage excelPackage = new ExcelPackage())
{
    //create 2 WorkSheets. One for the source data and one for the Pivot table
    ExcelWorksheet worksheetPivot = excelPackage.Workbook.Worksheets.Add("Pivot");
    ExcelWorksheet worksheetData = excelPackage.Workbook.Worksheets.Add("Data");

    //add some source data
    worksheetData.Cells["A1"].Value = "Column A";
    worksheetData.Cells["A2"].Value = "Group A";
    worksheetData.Cells["A3"].Value = "Group B";
    worksheetData.Cells["A4"].Value = "Group C";
    worksheetData.Cells["A5"].Value = "Group A";
    worksheetData.Cells["A6"].Value = "Group B";
    worksheetData.Cells["A7"].Value = "Group C";
    worksheetData.Cells["A8"].Value = "Group A";
    worksheetData.Cells["A9"].Value = "Group B";
    worksheetData.Cells["A10"].Value = "Group C";
    worksheetData.Cells["A11"].Value = "Group D";

    worksheetData.Cells["B1"].Value = "Column B";
    worksheetData.Cells["B2"].Value = "emc";
    worksheetData.Cells["B3"].Value = "fma";
    worksheetData.Cells["B4"].Value = "h2o";
    worksheetData.Cells["B5"].Value = "emc";
    worksheetData.Cells["B6"].Value = "fma";
    worksheetData.Cells["B7"].Value = "h2o";
    worksheetData.Cells["B8"].Value = "emc";
    worksheetData.Cells["B9"].Value = "fma";
    worksheetData.Cells["B10"].Value = "h2o";
    worksheetData.Cells["B11"].Value = "emc";

    worksheetData.Cells["C1"].Value = "Column C";
    worksheetData.Cells["C2"].Value = 299;
    worksheetData.Cells["C3"].Value = 792;
    worksheetData.Cells["C4"].Value = 458;
    worksheetData.Cells["C5"].Value = 299;
    worksheetData.Cells["C6"].Value = 792;
    worksheetData.Cells["C7"].Value = 458;
    worksheetData.Cells["C8"].Value = 299;
    worksheetData.Cells["C9"].Value = 792;
    worksheetData.Cells["C10"].Value = 458;
    worksheetData.Cells["C11"].Value = 299;

    worksheetData.Cells["D1"].Value = "Column D";
    worksheetData.Cells["D2"].Value = 40075;
    worksheetData.Cells["D3"].Value = 31415;
    worksheetData.Cells["D4"].Value = 384400;
    worksheetData.Cells["D5"].Value = 40075;
    worksheetData.Cells["D6"].Value = 31415;
    worksheetData.Cells["D7"].Value = 384400;
    worksheetData.Cells["D8"].Value = 40075;
    worksheetData.Cells["D9"].Value = 31415;
    worksheetData.Cells["D10"].Value = 384400;
    worksheetData.Cells["D11"].Value = 40075;

    //define the data range on the source sheet
    var dataRange = worksheetData.Cells[worksheetData.Dimension.Address];

    //create the pivot table
    var pivotTable = worksheetPivot.PivotTables.Add(worksheetPivot.Cells["B2"], dataRange,
"PivotTable");

    //label field
    pivotTable.RowFields.Add(pivotTable.Fields["Column A"]);
    pivotTable.DataOnRows = false;

    //data fields
    var field = pivotTable.DataFields.Add(pivotTable.Fields["Column B"]);
    field.Name = "Count of Column B";
    field.Function = DataFieldFunctions.Count;

    field = pivotTable.DataFields.Add(pivotTable.Fields["Column C"]);
    field.Name = "Sum of Column C";
    field.Function = DataFieldFunctions.Sum;
    field.Format = "0.00";

    field = pivotTable.DataFields.Add(pivotTable.Fields["Column D"]);
    field.Name = "Sum of Column D";
    field.Function = DataFieldFunctions.Sum;
    field.Format = "€#,##0.00";
}
```

Read Pivot Table online: https://riptutorial.com/epplus/topic/8767/pivot-table

Chapter 11: Rich Text in cells

Introduction

Most of the time, when we create spreadsheets, we just use a Cell's Value property to put content in the cell and the Style property to format it.

Occasionally, however, we may wish to apply multiple styles to a cell - maybe put a bold and underlined title before the rest of the content, or highlight a particular part of the text in Red - this is where the cell's RichText property comes into play.

Examples

Adding RichText to a cell

Each element of text you want to use distinct formatting on should be added separately, by adding to the cell's RichText collection property.

```
var cell = ws.Cells[1,1];
cell.IsRichText = true;     // Cell contains RichText rather than basic values
cell.Style.WrapText = true; // Required to honor new lines

var title = cell.RichText.Add("This is my title");
var text = cell.RichText.Add("\nAnd this is my text");
```

Note that each time you Add() a new string, it will inherit the formatting from the previous section. As such, if you want to change the default formatting you will only need to change it on the first string added.

This behaviour can, however, cause some confusion when formatting your text. Using the example above, the following code will make ***all of the text*** in the cell Bold and Italic - this is not the desired behavior:

```
// Common Mistake
var title = cell.RichText.Add("This is my title");
title.Bold = true;
title.Italic = true;

var text = cell.RichText.Add("\nAnd this is my text"); // Will be Bold and Italic too
```

The preferred approach is to add all your text sections first, then apply section-specific formatting afterwards, as shown here:

```
var title = cell.RichText.Add("This is my title");
title.FontName = "Verdana";    // This will be applied to all subsequent sections as well

var text = cell.RichText.Add("\nAnd this is my text");
```

```
// Format JUST the title
title.Bold = true;
title.Italic = true;
```

Text formatting Properties

There are a number of properties that can be applied to sections of RichText.

```
var title = cell.RichText.Add("This is my title");

// Data Type:     bool
// Default Value: false
title.Bold = true;

// Data Type:     System.Drawing.Color
// Default Value: Color.Black
title.Color = Color.Red;
title.Color = Color.FromArgb(255, 0, 0);
title.Color = ColorTranslator.FromHtml("#FF0000");

// Data Type:     string
// Default Value: "Calibri"
title.FontName = "Verdana";

// Data Type:     bool
// Default Value: false
title.Italic = true;

// Data Type:     bool
// Default Value: true
// If this property is set to false, any whitespace (including new lines)
// is trimmed from the start and end of the Text
title.PreserveSpace = true;

// Data Type:     float
// Default Value: 11
// The font size is specified in Points
title.Size = 16;

// Data Type:     bool
// Default Value: false
// Strikethrough
title.Strike = false;

// Data Type:     string
// Default Value: Whatever was set when the text was added to the RichText collection
title.Text += " (updated)";

// Data Type:     bool
// Default Value: false
title.UnderLine = true;

// Data Type:     OfficeOpenXml.Style.ExcelVerticalAlignmentFont
// Default Value: ExcelVerticalAlignmentFont.None
title.VerticalAlign = ExcelVerticalAlignmentFont.None;
```

Inserting RichText in a cell

EPPlus also supports the ability to insert text in a cell using the Insert() method. For example:

```
var file = new FileInfo(filePath);
using (var p = new ExcelPackage(file))
{
    var wb = p.Workbook;
    var ws = wb.Worksheets.FirstOrDefault() ?? wb.Worksheets.Add("Sheet1");

    var cell = ws.Cells[1, 1];
    cell.IsRichText = true;
    cell.RichText.Clear(); // Remove any RichText that may be in the cell already
    var s1 = cell.RichText.Add("Section 1.");
    var s2 = cell.RichText.Add("Section 2.");

    var s3 = cell.RichText.Insert(1, "Section 3.");

    s3.Bold = true;
    p.Save();
}
```

Note that the Insert() method does NOT insert at a character index, but at a Section index. Because the sections are zero-indexed, the above code will produce the following text in the cell:

Section 1.Section 3.Section 2.

Read Rich Text in cells online: https://riptutorial.com/epplus/topic/10776/rich-text-in-cells

Chapter 12: Saving the Excel document

Introduction

Examples on how to save the created Excel sheet to the Disk or send it to the Browser.

Examples

Save to disk

```
//Using File.WriteAllBytes
using (ExcelPackage excelPackage = new ExcelPackage())
{
    //create a new Worksheet
    ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Sheet 1");

    //add some text to cell A1
    worksheet.Cells["A1"].Value = "My second EPPlus spreadsheet!";

    //convert the excel package to a byte array
    byte[] bin = excelPackage.GetAsByteArray();

    //the path of the file
    string filePath = "C:\\ExcelDemo.xlsx";

    //or if you use asp.net, get the relative path
    filePath = Server.MapPath("ExcelDemo.xlsx");

    //write the file to the disk
    File.WriteAllBytes(filePath, bin);

    //Instead of converting to bytes, you could also use FileInfo
    FileInfo fi = new FileInfo(filePath);
    excelPackage.SaveAs(fi);
}

//Using SaveAs
using (ExcelPackage excelPackage = new ExcelPackage())
{
    //create a new Worksheet
    ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Sheet 1");

    //add some text to cell A1
    worksheet.Cells["A1"].Value = "My second EPPlus spreadsheet!";
    //the path of the file
    string filePath = "C:\\ExcelDemo.xlsx";

    //or if you use asp.net, get the relative path
    filePath = Server.MapPath("ExcelDemo.xlsx");

    //Write the file to the disk
    FileInfo fi = new FileInfo(filePath);
    excelPackage.SaveAs(fi);
}
```

Send to the Browser

```
//create a new ExcelPackage
using (ExcelPackage excelPackage = new ExcelPackage())
{
    //create the WorkSheet
    ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Sheet 1");

    //add some text to cell A1
    worksheet.Cells["A1"].Value = "My second EPPlus spreadsheet!";

    //convert the excel package to a byte array
    byte[] bin = excelPackage.GetAsByteArray();

    //clear the buffer stream
    Response.ClearHeaders();
    Response.Clear();
    Response.Buffer = true;

    //set the correct contenttype
    Response.ContentType = "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet";

    //set the correct length of the data being send
    Response.AddHeader("content-length", bin.Length.ToString());

    //set the filename for the excel package
    Response.AddHeader("content-disposition", "attachment; filename=\"ExcelDemo.xlsx\"");

    //send the byte array to the browser
    Response.OutputStream.Write(bin, 0, bin.Length);

    //cleanup
    Response.Flush();
    HttpContext.Current.ApplicationInstance.CompleteRequest();
}
```

Save to disk with SaveFileDialog

```
//Using File.WriteAllBytes
using (ExcelPackage excelPackage = new ExcelPackage())
{
    //create a new Worksheet
    ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Sheet 1");

    //add some text to cell A1
    worksheet.Cells["A1"].Value = "My fourth EPPlus spreadsheet!";

    //convert the excel package to a byte array
    byte[] bin = excelPackage.GetAsByteArray();

    //create a SaveFileDialog instance with some properties
    SaveFileDialog saveFileDialog1 = new SaveFileDialog();
    saveFileDialog1.Title = "Save Excel sheet";
    saveFileDialog1.Filter = "Excel files|*.xlsx|All files|*.*";
    saveFileDialog1.FileName = "ExcelSheet_" + DateTime.Now.ToString("dd-MM-yyyy") + ".xlsx";

    //check if user clicked the save button
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        //write the file to the disk
        File.WriteAllBytes(saveFileDialog1.FileName, bin);
    }
}

//Using SaveAs
using (ExcelPackage excelPackage = new ExcelPackage())
{
    //create a new Worksheet
    ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Sheet 1");

    //add some text to cell A1
    worksheet.Cells["A1"].Value = "My fourth EPPlus spreadsheet!";

    //create a SaveFileDialog instance with some properties
    SaveFileDialog saveFileDialog1 = new SaveFileDialog();
    saveFileDialog1.Title = "Save Excel sheet";
    saveFileDialog1.Filter = "Excel files|*.xlsx|All files|*.*";
    saveFileDialog1.FileName = "ExcelSheet_" + DateTime.Now.ToString("dd-MM-yyyy") + ".xlsx";

    //check if user clicked the save button
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        //Get the FileInfo
        FileInfo fi = new FileInfo(saveFileDialog1.FileName);
        //write the file to the disk
        excelPackage.SaveAs(fi);
    }
}
```

Read Saving the Excel document online: https://riptutorial.com/epplus/topic/8202/saving-the-excel-document

Chapter 13: Styling the Excel document

Introduction

How to style Cells with font types, background color, border styles etc.

Examples

Background color

```
//fill column A with solid red color from hex
worksheet.Column(1).Style.Fill.PatternType = ExcelFillStyle.Solid;
worksheet.Column(1).Style.Fill.BackgroundColor.SetColor(ColorTranslator.FromHtml("#FF0000"));

//fill row 4 with striped orange background
worksheet.Row(4).Style.Fill.PatternType = ExcelFillStyle.DarkHorizontal;
worksheet.Row(4).Style.Fill.BackgroundColor.SetColor(Color.Orange);
```

Border styles

```
//make the borders of cell F6 thick
worksheet.Cells[6, 6].Style.Border.Top.Style = ExcelBorderStyle.Thick;
worksheet.Cells[6, 6].Style.Border.Right.Style = ExcelBorderStyle.Thick;
worksheet.Cells[6, 6].Style.Border.Bottom.Style = ExcelBorderStyle.Thick;
worksheet.Cells[6, 6].Style.Border.Left.Style = ExcelBorderStyle.Thick;

//make the borders of cells A18 - J18 double and with a purple color
worksheet.Cells["A18:J18"].Style.Border.Top.Style = ExcelBorderStyle.Double;
worksheet.Cells["A18:J18"].Style.Border.Bottom.Style = ExcelBorderStyle.Double;
worksheet.Cells["A18:J18"].Style.Border.Top.Color.SetColor(Color.Purple);
worksheet.Cells["A18:J18"].Style.Border.Bottom.Color.SetColor(Color.Purple);
```

Font styles

```
//set the font type for cells C1 - C30
worksheet.Cells["C1:C30"].Style.Font.Size = 13;
worksheet.Cells["C1:C30"].Style.Font.Name = "Calibri";
worksheet.Cells["C1:C30"].Style.Font.Bold = true;
worksheet.Cells["C1:C30"].Style.Font.Color.SetColor(Color.Blue);

//Multiple Fonts in the same cell
ExcelRange rg = worksheet.Cells["A1"];
rg.IsRichText = true;
//ExcelRichText uses "using OfficeOpenXml.Style;"
ExcelRichText text1 = rg.RichText.Add("Text with Font1");
text1.Bold = true;
text1.Italic = true;
text1.Color = System.Drawing.Color.Blue;
ExcelRichText text2 = rg.RichText.Add("Text with Font2");
text2.UnderLine = true;
text2.Bold = false;
```

```
text2.Color = System.Drawing.Color.Red;
ExcelRichText text3 = rg.RichText.Add("Text with Font3");
text3.UnderLine = false;
text3.Strike = true;
```

Text alignment and word wrap

```
//make column H wider and set the text align to the top and right
worksheet.Column(8).Width = 25;
worksheet.Column(8).Style.HorizontalAlignment = ExcelHorizontalAlignment.Right;
worksheet.Column(8).Style.VerticalAlignment = ExcelVerticalAlignment.Top;

//wrap text in the cells
worksheet.Column(8).Style.WrapText = true;
```

Complete example with all styles

```
//create a new ExcelPackage
using (ExcelPackage excelPackage = new ExcelPackage())
{
    //create the WorkSheet
    ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Sheet 1");

    //add some dummy data, note that row and column indexes start at 1
    for (int i = 1; i <= 30; i++)
    {
        for (int j = 1; j <= 15; j++)
        {
            worksheet.Cells[i, j].Value = "Row " + i + ", Column " + j;
        }
    }

    //fill column A with solid red color
    worksheet.Column(1).Style.Fill.PatternType = ExcelFillStyle.Solid;

worksheet.Column(1).Style.Fill.BackgroundColor.SetColor(ColorTranslator.FromHtml("#FF0000"));

    //set the font type for cells C1 - C30
    worksheet.Cells["C1:C30"].Style.Font.Size = 13;
    worksheet.Cells["C1:C30"].Style.Font.Name = "Calibri";
    worksheet.Cells["C1:C30"].Style.Font.Bold = true;
    worksheet.Cells["C1:C30"].Style.Font.Color.SetColor(Color.Blue);

    //fill row 4 with striped orange background
    worksheet.Row(4).Style.Fill.PatternType = ExcelFillStyle.DarkHorizontal;
    worksheet.Row(4).Style.Fill.BackgroundColor.SetColor(Color.Orange);

    //make the borders of cell F6 thick
    worksheet.Cells[6, 6].Style.Border.Top.Style = ExcelBorderStyle.Thick;
    worksheet.Cells[6, 6].Style.Border.Right.Style = ExcelBorderStyle.Thick;
    worksheet.Cells[6, 6].Style.Border.Bottom.Style = ExcelBorderStyle.Thick;
    worksheet.Cells[6, 6].Style.Border.Left.Style = ExcelBorderStyle.Thick;

    //make the borders of cells A18 - J18 double and with a purple color
    worksheet.Cells["A18:J18"].Style.Border.Top.Style = ExcelBorderStyle.Double;
    worksheet.Cells["A18:J18"].Style.Border.Bottom.Style = ExcelBorderStyle.Double;
    worksheet.Cells["A18:J18"].Style.Border.Top.Color.SetColor(Color.Purple);
    worksheet.Cells["A18:J18"].Style.Border.Bottom.Color.SetColor(Color.Purple);

    //make all text fit the cells
    worksheet.Cells[worksheet.Dimension.Address].AutoFitColumns();

    //i use this to make all columms just a bit wider, text would sometimes still overflow
after AutoFitColumns(). Bug?
    for (int col = 1; col <= worksheet.Dimension.End.Column; col++)
    {
        worksheet.Column(col).Width = worksheet.Column(col).Width + 1;
    }

    //make column H wider and set the text align to the top and right
    worksheet.Column(8).Width = 25;
    worksheet.Column(8).Style.HorizontalAlignment = ExcelHorizontalAlignment.Right;
    worksheet.Column(8).Style.VerticalAlignment = ExcelVerticalAlignment.Top;

    //get the image from disk
    using (System.Drawing.Image image =
System.Drawing.Image.FromFile(HttpContext.Current.Server.MapPath("logo.jpg")))
    {
        var excelImage = worksheet.Drawings.AddPicture("My Logo", image);

        //add the image to row 20, column E
        excelImage.SetPosition(20, 0, 5, 0);
    }
}
```

Add an image to a sheet

```
//get the image from disk
using (System.Drawing.Image image =
System.Drawing.Image.FromFile(HttpContext.Current.Server.MapPath("logo.jpg")))
{
    var excelImage = worksheet.Drawings.AddPicture("My Logo", image);

    //add the image to row 20, column E
    excelImage.SetPosition(20, 0, 5, 0);
}
```

Read Styling the Excel document online: https://riptutorial.com/epplus/topic/8219/styling-the-excel-document

Chapter 14: Tables

Introduction

This topic describe how to add and style tables

Examples

Adding and formating a table

```
//Using statement for ExcelTable and TableStyles
using OfficeOpenXml.Table;


//Defining the tables parameters
int firstRow =1;
int lastRow = worksheet.Dimension.End.Row;
int firstColumn = 1;
int lastColumn = worksheet.Dimension.End.Column;
ExcelRange rg = worksheet.Cells[firstRow, firstColumn, lastRow, LastColumn];
string tableName = "Table1";

//Ading a table to a Range
ExcelTable tab = worksheet.Tables.Add(rg, tableName);

//Formating the table style
tab.TableStyle = TableStyles.Light8;
```

Read Tables online: https://riptutorial.com/epplus/topic/8720/tables

Chapter 15: User Input Validation

Introduction

How to validade user inputs. Validation constrains the values a user can input into a cell, and/or set a combobox for the user select the value for the cell. Optionally, a message can be displayed when the user clicks in a cell and a message error, when the validation fails.

Examples

List Validation

```
//Add a List validation to B column. Values should be in a list
var val = worksheet.DataValidations.AddListValidation("B:B");
//Shows error message when the input doesn't match the accepted values
val.ShowErrorMessage = true;
//Style of warning. "information" and "warning" allow users to ignore the validation,
//while "stop" and "undefined" doesn't
val.ErrorStyle = OfficeOpenXml.DataValidation.ExcelDataValidationWarningStyle.information;
//Title of the error mesage box
val.ErrorTitle = "This is the title";
//Message of the error
val.Error = "This is the message";
//Set to true to show a prompt when user clics on the cell
val.ShowInputMessage = true;
//Set the message for the prompt
val.Prompt = "This is a input message";
//Set the title for the prompt
val.PromptTitle = "This is the title from the input message";
//Define the accepted values
val.Formula.Values.Add("This is accepted");
val.Formula.Values.Add("This is also accepted");
val.Formula.Values.Add("Any other thing is rejected");
//Set to true if blank value is accepted
val.AllowBlank = false;

//Add a List validation to the C column
var val2 = worksheet.DataValidations.AddListValidation("C:C");
//Define the Cells with the accepted values
val2.Formula.ExcelFormula = "=$D$3:$D$5";
//Fill the cells with the accepted values
worksheet.Cells["D3"].Value = "Val1";
worksheet.Cells["D4"].Value = "Val2";
worksheet.Cells["D5"].Value = "Val3";
```

Integer Validation

```
//Add a List validation to the C column
var val3 = worksheet.DataValidations.AddIntegerValidation("E:E");
//For Integer Validation, you have to set error message to true
val3.ShowErrorMessage = true;
val3.Error = "The value must be an integer between 0 and 10";
```

```
//Minimum allowed Value
val3.Formula.Value = 0;
//Maximum allowed Value
val3.Formula2.Value = 10;
//If the cells are not filled, allow blanks or fill with a valid value,
//otherwise it could generate a error when saving
val3.AllowBlank = true;
```

DateTime Validation

```
//Add a DateTime Validation to column F
var val4 = worksheet.DataValidations.AddDateTimeValidation("F:F");
//For DateTime Validation, you have to set error message to true
val4.ShowErrorMessage = true;
//Minimum allowed date
val4.Formula.Value = new DateTime(2017,03,15, 01, 0,0);
//Maximum allowed date
val4.Formula2.Value= new DateTime(2017, 03, 16, 12, 0, 0);
val4.AllowBlank = true;
```

Text Length Validation

```
//Add a TextLength Validation to column G
var val5 = worksheet.DataValidations.AddTextLengthValidation("G:G");
//For TextLenght Validation, you have to set error message to true
val5.ShowErrorMessage = true;
//Minimum allowed text lenght
val5.Formula.Value = 3;
//Maximum allowed text lenght
val5.Formula2.Value = 5;
val5.AllowBlank = true;
```

Read User Input Validation online: https://riptutorial.com/epplus/topic/8739/user-input-validation

Credits

S. No	Chapters	Contributors
1	Getting started with epplus	Community, Magnetron, VDWWD
2	Append data to existing document	VDWWD
3	Columns and Rows	hellyale, Magnetron, VDWWD
4	Creating charts	VDWWD
5	Creating formulas and calculate ranges	Magnetron, VDWWD
6	Filling the document with data	VDWWD
7	Formatting values	Magnetron, VDWWD
8	Importing data from existing file	VDWWD
9	Merge Cells	Magnetron
10	Pivot Table	VDWWD
11	Rich Text in cells	Pete
12	Saving the Excel document	Magnetron, VDWWD
13	Styling the Excel document	Magnetron, VDWWD
14	Tables	Magnetron
15	User Input Validation	Magnetron