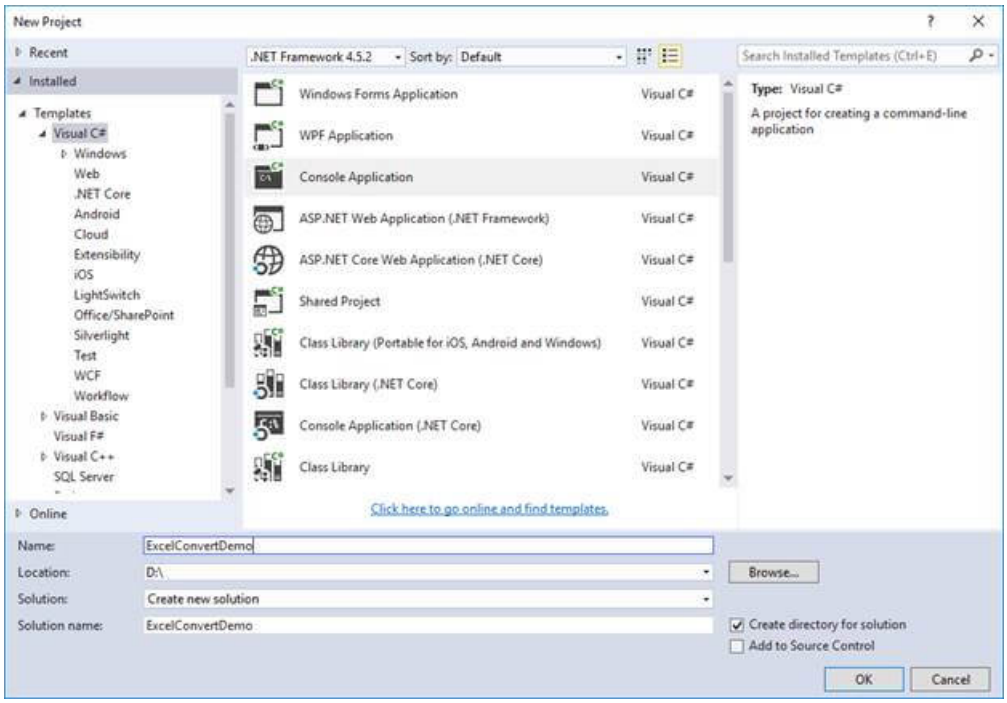
Creating Excel File Using OpenXML

Program.zip

In this example, let's create a demo console project in Visual Studio by selecting File -> New -> Project. From the project window, select Console template and type name and select path for the project, as shown below.



After creating the project, create a Model Class and give a name as TestModel, as shown below.

```
01. public class TestModel
02.  {
03.      public int TestId { get; set; }
04.      public string TestName { get; set; }
05.      public string TestDesc { get; set; }
06.      public DateTime TestDate { get; set; }
07.  }
```

Create one more class and give name as TestModelList.

```
01. public class TestModelList
02. {
03.     public List<TestModel> testData { get; set; }
04. }
```

Let's add some hard coded data into this model. For that, write the below code in the main method of Program.cs file.

```
01. class Program
02. {
03.     static void Main(string[] args)
```

```
        TestModelList tmList = new TestModelList();
        tmList.testData = new List<TestModel>();
        TestModel tm = new TestModel();

        tm.TestDesc = "Tested 1 time";
        tm.TestDate = DateTime.Now.Date;
        tmList.testData.Add(tm);

        TestModel tm1 = new TestModel();
        tm1.TestId = 2;
        tm1.TestName = "Test2";
        tm1.TestDesc = "Tested 2 times";
        tm1.TestDate = DateTime.Now.AddDays(-1);
        tmList.testData.Add(tm1);

        TestModel tm2 = new TestModel();
        tm2.TestId = 3;
        tm2.TestName = "Test3";
        tm2.TestDesc = "Tested 3 times";
        tm2.TestDate = DateTime.Now.AddDays(-2);
        tmList.testData.Add(tm2);

        TestModel tm3 = new TestModel();
        tm3.TestId = 4;
        tm3.TestName = "Test4";
        tm3.TestDesc = "Tested 4 times";
        tm3.TestDate = DateTime.Now.AddDays(-3);
        tmList.testData.Add(tm);

    }
}
```

Now, we have got a Model ready. So, let's start writing functions for creating an Excel file using OpenXml. For this, add OpenXml from NuGet Packages by right-clicking the project and selecting "Manage NuGet Package" and search openxml. From the list, select DocumentFormat.OpenXml as shown below and install it.

Next, create functions for creating an Excel package using OpenXml as shown below.

- First, import OpenXml packages as shown below.

```
using DocumentFormat.OpenXml;
using DocumentFormat.OpenXml.Packaging;
using DocumentFormat.OpenXml.Spreadsheet;
using X14 = DocumentFormat.OpenXml.Office2010.Excel;
using X15 = DocumentFormat.OpenXml.Office2013.Excel;
```

  Then, add the below code for creating an Excel file into given path.

```
public void CreateExcelFile(TestModelList data, string OutPutFileDirectory)
{
    var datetime = DateTime.Now.ToString().Replace("/", "_").Replace(":", "_");

    string fileFullname = Path.Combine(OutPutFileDirectory, "Output.xlsx");

    if (File.Exists(fileFullname))
    {
        fileFullname = Path.Combine(OutPutFileDirectory, "Output_" + datetime + ".xlsx");
    }

    using (SpreadsheetDocument package = SpreadsheetDocument.Create(fileFullname, SpreadsheetDocu
    {
        CreatePartsForExcel(package, data);
    }
}
```

- Write functions for creating workbook and worksheet into Excel.

```
    SheetData partSheetData = GenerateSheetdataForDetails(data);

    WorkbookPart workbookPart1 = document.AddWorkbookPart();

    WorkbookStylesPart workbookStylesPart1 = workbookPart1.AddNewPart<WorkbookStylesPart>("rId3");
    GenerateWorkbookStylesPartContent(workbookStylesPart1);

    WorksheetPart worksheetPart1 = workbookPart1.AddNewPart<WorksheetPart>("rId1");
    GenerateWorksheetPartContent(worksheetPart1, partSheetData);
}
```

- Write functions for creating workbook and work sheet content in Excel, as shown below.

```
private void GenerateWorkbookPartContent(WorkbookPart workbookPart1)
{
    Workbook workbook1 = new Workbook();
    Sheets sheets1 = new Sheets();
    Sheet sheet1 = new Sheet() { Name = "Sheet1", SheetId = (UInt32Value)1U, Id = "rId1" };
    sheets1.Append(sheet1);
    workbook1.Append(sheets1);
    workbookPart1.Workbook = workbook1;
}

private void GenerateWorksheetPartContent(WorksheetPart worksheetPart1, SheetData sheetData1)
{

Worksheet worksheet1 = new Worksheet() { MCAttributes = new MarkupCompatibilityAttributes() { Ignc

worksheet1.AddNamespaceDeclaration("r", "http://schemas.openxmlformats.org/officeDocument/2006/rel
    worksheet1.AddNamespaceDeclaration("mc", "http://schemas.openxmlformats.org/markup-compatibility/2006");

worksheet1.AddNamespaceDeclaration("x14ac", "http://schemas.microsoft.com/office/spreadsheetml/200
    SheetDimension sheetDimension1 = new SheetDimension() { Reference = "A1" };

    SheetViews sheetViews1 = new SheetViews();


SheetView sheetView1 = new SheetView() { TabSelected = true, WorkbookViewId = (UInt32Value)0U };

Selection selection1 = new Selection() { ActiveCell = "A1", SequenceOfReferences = new ListValue<S
() { InnerText = "A1" } };

    sheetView1.Append(selection1);

    sheetViews1.Append(sheetView1);

SheetFormatProperties sheetFormatProperties1 = new SheetFormatProperties() { DefaultRowHeight = 15


PageMargins pageMargins1 = new PageMargins() { Left = 0.7D, Right = 0.7D, Top = 0.75D, Bottom = 0
    worksheet1.Append(sheetDimension1);
    worksheet1.Append(sheetViews1);
    worksheet1.Append(sheetFormatProperties1);
    worksheet1.Append(sheetData1);
    worksheet1.Append(pageMargins1);
    worksheetPart1.Worksheet = worksheet1;
}
```

- Write code for workbook styles by giving your own font size, color, font name, border properties, cell style formats etc. as shown below.

```
private void GenerateWorkbookStylesPartContent(WorkbookStylesPart workbookStylesPart1)
{

Stylesheet stylesheet1 = new Stylesheet() { MCAttributes = new MarkupCompatibilityAttributes() {
    stylesheet1.AddNamespaceDeclaration("mc", "http://schemas.openxmlformats.org/markup-compatibility/2006");
```

```
    Fonts fonts1 = new Fonts() { Count = (UInt32Value)2U, KnownFonts

    Color color1 = new Color() { Theme = (UInt32Value)1U };
    FontName fontName1 = new FontName() { Val = "Calibri" };
    FontFamilyNumbering fontFamilyNumbering1 = new FontFamilyNumbering() { Val = 2 };
    FontScheme fontScheme1 = new FontScheme() { Val = FontSchemeValues.Minor };

    font1.Append(fontSize1);
    font1.Append(color1);
    font1.Append(fontName1);
    font1.Append(fontFamilyNumbering1);
    font1.Append(fontScheme1);

    Font font2 = new Font();
    Bold bold1 = new Bold();
    FontSize fontSize2 = new FontSize() { Val = 11D };
    Color color2 = new Color() { Theme = (UInt32Value)1U };
    FontName fontName2 = new FontName() { Val = "Calibri" };
    FontFamilyNumbering fontFamilyNumbering2 = new FontFamilyNumbering() { Val = 2 };
    FontScheme fontScheme2 = new FontScheme() { Val = FontSchemeValues.Minor };

    font2.Append(bold1);
    font2.Append(fontSize2);
    font2.Append(color2);
    font2.Append(fontName2);
    font2.Append(fontFamilyNumbering2);
    font2.Append(fontScheme2);

    fonts1.Append(font1);
    fonts1.Append(font2);

    Fills fills1 = new Fills() { Count = (UInt32Value)2U };

    Fill fill1 = new Fill();
    PatternFill patternFill1 = new PatternFill() { PatternType = PatternValues.None };

    fill1.Append(patternFill1);

    Fill fill2 = new Fill();
    PatternFill patternFill2 = new PatternFill() { PatternType = PatternValues.Gray125 };

    fill2.Append(patternFill2);

    fills1.Append(fill1);
    fills1.Append(fill2);

    Borders borders1 = new Borders() { Count = (UInt32Value)2U };

    Border border1 = new Border();
    LeftBorder leftBorder1 = new LeftBorder();
    RightBorder rightBorder1 = new RightBorder();
    TopBorder topBorder1 = new TopBorder();
    BottomBorder bottomBorder1 = new BottomBorder();
    DiagonalBorder diagonalBorder1 = new DiagonalBorder();

    border1.Append(leftBorder1);
    border1.Append(rightBorder1);
    border1.Append(topBorder1);
    border1.Append(bottomBorder1);
    border1.Append(diagonalBorder1);

    Border border2 = new Border();

    LeftBorder leftBorder2 = new LeftBorder() { Style = BorderStyleValues.Thin };
    Color color3 = new Color() { Indexed = (UInt32Value)64U };

    leftBorder2.Append(color3);

    RightBorder rightBorder2 = new RightBorder() { Style = BorderStyleValues.Thin };
    Color color4 = new Color() { Indexed = (UInt32Value)64U };
```

```
    TopBorder topBorder2 = new TopBorder() { Style = BorderStyleValue
    Color color5 = new Color() { Indexed = (UInt32Value)64U };


    BottomBorder bottomBorder2 = new BottomBorder() { Style = BorderStyleValues.Thin };
    Color color6 = new Color() { Indexed = (UInt32Value)64U };

    bottomBorder2.Append(color6);
    DiagonalBorder diagonalBorder2 = new DiagonalBorder();

    border2.Append(leftBorder2);
    border2.Append(rightBorder2);
    border2.Append(topBorder2);
    border2.Append(bottomBorder2);
    border2.Append(diagonalBorder2);

    borders1.Append(border1);
    borders1.Append(border2);

    CellStyleFormats cellStyleFormats1 = new CellStyleFormats() { Count = (UInt32Value)1U };

CellFormat cellFormat1 = new CellFormat() { NumberFormatId = (UInt32Value)0U, FontId = (UInt32Valu

    cellStyleFormats1.Append(cellFormat1);

    CellFormats cellFormats1 = new CellFormats() { Count = (UInt32Value)3U };

CellFormat cellFormat2 = new CellFormat() { NumberFormatId = (UInt32Value)0U, FontId = (UInt32Valu

CellFormat cellFormat3 = new CellFormat() { NumberFormatId = (UInt32Value)0U, FontId = (UInt32Valu
            CellFormat cellFormat4 = new CellFormat() { NumberFormatId = (UInt32Value)0U, FontId =

    cellFormats1.Append(cellFormat2);
    cellFormats1.Append(cellFormat3);
    cellFormats1.Append(cellFormat4);

    CellStyles cellStyles1 = new CellStyles() { Count = (UInt32Value)1U };

CellStyle cellStyle1 = new CellStyle() { Name = "Normal", FormatId = (UInt32Value)0U, BuiltinId =

    cellStyles1.Append(cellStyle1);

DifferentialFormats differentialFormats1 = new DifferentialFormats() { Count = (UInt32Value)0U };

 TableStyles tableStyles1 = new TableStyles() { Count = (UInt32Value)0U, DefaultTableStyle = "Tabl

    StylesheetExtensionList stylesheetExtensionList1 = new StylesheetExtensionList();

    StylesheetExtension stylesheetExtension1 = new StylesheetExtension() { Uri = "{EB79DEF2-
80B8-43e5-95BD-54CBDDF9020C}" };

stylesheetExtension1.AddNamespaceDeclaration("x14", "http://schemas.microsoft.com/office/spreadsh
            X14.SlicerStyles slicerStyles1 = new X14.SlicerStyles() { DefaultSlicerStyle = "Slicer

    stylesheetExtension1.Append(slicerStyles1);

    StylesheetExtension stylesheetExtension2 = new StylesheetExtension() { Uri = "{9260A510-
F301-46a8-8635-F512D64BE5F5}" };

stylesheetExtension2.AddNamespaceDeclaration("x15", "http://schemas.microsoft.com/office/spreadsh

X15.TimelineStyles timelineStyles1 = new X15.TimelineStyles() { DefaultTimelineStyle = "TimeSlicer

    stylesheetExtension2.Append(timelineStyles1);

    stylesheetExtensionList1.Append(stylesheetExtension1);
    stylesheetExtensionList1.Append(stylesheetExtension2);

    stylesheet1.Append(fonts1);
    stylesheet1.Append(fills1);
    stylesheet1.Append(borders1);
```

```
145.     stylesheet1.Append(cellStyles1);
146.     stylesheet1.Append(differentialFormats1);
147.     stylesheet1.Append(tableStyles1);

150.     workbookStylesPart1.Stylesheet = stylesheet1;
151. }
```

- Write a function for generating workbook content, as shown below.

```
01. private void GenerateWorkbookPartContent(WorkbookPart workbookPart1)
02. {
03.     Workbook workbook1 = new Workbook();
04.     Sheets sheets1 = new Sheets();
05.     Sheet sheet1 = new Sheet() { Name = "Sheet1", SheetId = (UInt32Value)1U, Id = "rId1" };
06.     sheets1.Append(sheet1);
07.     workbook1.Append(sheets1);
08.     workbookPart1.Workbook = workbook1;
09. }
```

- Write the below functions to add data into Excel.

```
01. private SheetData GenerateSheetdataForDetails(TestModelList data)
02. {
03.     SheetData sheetData1 = new SheetData();
04.     sheetData1.Append(CreateHeaderRowForExcel());
05.
06.     foreach (TestModel testmodel in data.testData)
07.     {
08.         Row partsRows = GenerateRowForChildPartDetail(testmodel);
09.         sheetData1.Append(partsRows);
10.     }
11.     return sheetData1;
12. }
```

The below function is created for creating Header rows in Excel.

```
01. private Row CreateHeaderRowForExcel()
02. {
03.     Row workRow = new Row();
04.     workRow.Append(CreateCell("Test Id", 2U));
05.     workRow.Append(CreateCell("Test Name", 2U));
06.     workRow.Append(CreateCell("Test Description", 2U));
07.     workRow.Append(CreateCell("Test Date", 2U));
08.     return workRow;
09. }
```

Below function is used for generating child rows.

```
01. private Row GenerateRowForChildPartDetail(TestModel testmodel)
02. {
03.     Row tRow = new Row();
04.     tRow.Append(CreateCell(testmodel.TestId.ToString()));
05.     tRow.Append(CreateCell(testmodel.TestName));
06.     tRow.Append(CreateCell(testmodel.TestDesc));
07.     tRow.Append(CreateCell(testmodel.TestDate.ToShortDateString()));
08.
09.     return tRow;
10. }
```

Below function is used for creating cell by passing only cell data and it adds default style.

```
01. private Cell CreateCell(string text)
02. {
03.     Cell cell = new Cell();
04.     cell.StyleIndex = 1U;
05.     cell.DataType = ResolveCellDataTypeOnValue(text);
06.     cell.CellValue = new CellValue(text);
07.     return cell;
```

Below function is used for creating a cell by passing cell data and cell style.

```
01. private Cell CreateCell(string text, uint styleIndex)
04.     cell.StyleIndex = styleIndex;
05.     cell.DataType = ResolveCellDataTypeOnValue(text);
06.     cell.CellValue = new CellValue(text);
07.     return cell;
08. }
```

Below function is created for resolving the data type of numeric value in a cell.

```
01. private EnumValue<CellValues> ResolveCellDataTypeOnValue(string text)
02. {
03.     int intVal;
04.     double doubleVal;
05.     if (int.TryParse(text, out intVal) || double.TryParse(text, out doubleVal))
06.     {
07.         return CellValues.Number;
08.     }
09.     else
10.     {
11.         return CellValues.String;
12.     }
13. }
```

Now, let's call the main function for generating Excel file into main method with passing our model into it.

```
01. static void Main(string[] args)
02.  {
03.
04.      TestModelList tmList = new TestModelList();
05.      tmList.testData = new List<TestModel>();
06.      TestModel tm = new TestModel();
07.      tm.TestId = 1;
08.      tm.TestName = "Test1";
09.      tm.TestDesc = "Tested 1 time";
10.      tm.TestDate = DateTime.Now.Date;
11.      tmList.testData.Add(tm);
12.
13.      TestModel tm1 = new TestModel();
14.      tm1.TestId = 2;
15.      tm1.TestName = "Test2";
16.      tm1.TestDesc = "Tested 2 times";
17.      tm1.TestDate = DateTime.Now.AddDays(-1);
18.      tmList.testData.Add(tm1);
19.
20.      TestModel tm2 = new TestModel();
21.      tm2.TestId = 3;
22.      tm2.TestName = "Test3";
23.      tm2.TestDesc = "Tested 3 times";
24.      tm2.TestDate = DateTime.Now.AddDays(-2);
25.      tmList.testData.Add(tm2);
26.
27.      TestModel tm3 = new TestModel();
28.      tm3.TestId = 4;
29.      tm3.TestName = "Test4";
30.      tm3.TestDesc = "Tested 4 times";
31.      tm3.TestDate = DateTime.Now.AddDays(-3);
32.      tmList.testData.Add(tm);
33.
34.      Program p = new ExelConvertDemo.Program();
35.      p.CreateTaktExcelFile(tmList, "d:\\");
36.  }
```

Output of the Excel file would be similar to the below image.

Creating Excel File | Excel File Using OpenXML | OpenXML

Rafnas T P TOP 500

A crazy and an enthusiastic developer who loves to develop on Microsoft Technologies including desktop, web, mobile applications and is passionate about writing C# code and SQL queries.

https://about.me/rafnas

157 2m 2

6 1

Type your comment here and press Enter Key (Minimum 10 characters)

Hi, I was trying to create excel with multiple sheets, but ending up in creating corrupted file or file without any formatting. Could you guide me in this?

Girish Mahida

1918 2 0

Dec 19, 2019

0 0 Reply

FEATURED ARTICLES

JWT Token Routing From Gateway To Multiple Micro-Services In .NET Core

CRUD Operations In PostgreSQL With EF Core And ASP.NET Core Web API

Views In Snowflake

Covid-19 Tracker Website With React, Material.UI And Chart.js

TRENDING UP

01 Getting Started With Microsoft Fluent UI React
02 What is Software-Defined Networking and Virtual Networks in Physical Networks
03 What are Virtualized Data Centers and VMware's SDDC Approach
04 What are Data Center Building Blocks and Network Virtualization Services?
05 Introduction to Network Virtualization
06 What is Virtual Networking?
07 Insert Data Into Azure Table Storage Using ASP.NET Core Application
08 What are Virtual Switches and Standard Switches?
09 What Is The Software-Defined Data Center (SDDC) And Physical Data Centers
10 Build A Simple Todo App Using Microsoft Fluent UI React

View All

/