# Manipulating Office Documents

# Project Set Up

## Setup Steps for Visual Studio

From within VS (using point and click):

* Open the WorkingWithDataAndFiles solution
* Add console project called office-demo.csproj to the solution
* Use NuGet to get a reference to the DocumentFormat.OpenXml and CSVHelper libraries.
* Download and add the following two files to the project by clicking on the links and carrying out the associated instructions:
  + [Movie\_data.docx](https://github.com/QACS-TL/DataAndFiles/blob/main/DataAndFilesDemos/movie_data.docx) (download by clicking the download button
  + [Top revenue earning SciFi movies per alphabetic letter.csv](https://github.com/QACS-TL/DataAndFiles/blob/main/DataAndFilesDemos/Top%20revenue%20earning%20SciFi%20movies%20per%20alphabetic%20letter.csv) (download by right clicking the Raw button at the top of the file, select Save Link As… , choose the location on your computer where you want to save the file, and select Save)
* Select the docx file in the Solution Explorer window.
* Locate the “Copy To Output Directory” property in the Properties window and select the “Copy Always” option.
* Follow the above two steps for the csv file.

## Setup Steps for Visual Studio Code

From Command Prompt:

* Add a new console project to the demos solution by typing:

dotnet new console --output office-demo

dotnet sln add office-demo

* Include the OpenXML and CSVHelper packages in the csv-demo project by typing:

dotnet add package DocumentFormat.OpenXml

dotnet add package CsvHelper

* Download and add the following two files to the project by clicking on the links and carrying out the associated instructions:
  + [Movie\_data.docx](https://github.com/QACS-TL/DataAndFiles/blob/main/DataAndFilesDemos/movie_data.docx) (download by clicking the download button
  + [Top revenue earning SciFi movies per alphabetic letter.csv](https://github.com/QACS-TL/DataAndFiles/blob/main/DataAndFilesDemos/Top%20revenue%20earning%20SciFi%20movies%20per%20alphabetic%20letter.csv) (download by right clicking the Raw button at the top of the file, select Save Link As… , choose the location on your computer where you want to save the file, and select Save)
* Open the csv-demo folder in Visual Studio Code
* Use Visual Studio Code to open the office-demo.csproj file.
* Open the project file and add an <ItemGroup> element just before the closing </Project> tag
* Add the following to the newly added <ItemGroup> tag:

<None Update="Movie\_data.docx.csv">

<CopyToOutputDirectory>Always</CopyToOutputDirectory>

</None>

<None Update="Top revenue earning SciFi movies per alphabetic letter.csv">

<CopyToOutputDirectory>Always</CopyToOutputDirectory>

</None>

* Save your changes

## Now do the following regardless of project type:

* Add a class to the project called Movie.cs
* Replace the code in the file with the following:

namespace office\_demo

{

internal class Movie

{

public string Movie\_ID { get; set; }

public string Title { get; set; }

public string Release\_Date { get; set; }

public decimal Revenue { get; set; }

public string Tagline { get; set; }

}

}

* Replace the Program.cs file’s content with the following:

using System.Globalization;

using CsvHelper;

using DocumentFormat.OpenXml.Packaging;

using DocumentFormat.OpenXml.Wordprocessing;

namespace office\_demo

{

internal class Program

{

static void Main(string[] args)

{

//The following code reads movie details in from a csv file and adds the data into a Word table

List<Movie> movieList = null;

using (var sr = new StreamReader("Top revenue earning SciFi movies per alphabetic letter.csv"))

using (var reader = new CsvReader(sr, CultureInfo.InvariantCulture))

{

movieList = reader.GetRecords<Movie>().ToList();

foreach (Movie m in movieList)

{

Console.WriteLine($"{m.Movie\_ID} is {m.Title}");

}

}

// Read a template document, replace some text, and save as another document

File.Copy("movie\_data.docx", "clone.docx", true);

int currentRow = 1;

foreach (Movie movie in movieList)

{

ChangeTextInCell("clone.docx", currentRow, 0, movie.Movie\_ID);

ChangeTextInCell("clone.docx", currentRow, 1, movie.Title);

ChangeTextInCell("clone.docx", currentRow, 2, movie.Release\_Date);

ChangeTextInCell("clone.docx", currentRow, 3, movie.Revenue.ToString("C"));

ChangeTextInCell("clone.docx", currentRow, 4, movie.Tagline);

currentRow++;

}

}

public static void ChangeTextInCell(string filepath, int rowpos, int cellpos, string text)

{

using (WordprocessingDocument doc =

WordprocessingDocument.Open(filepath, true))

{

// Find the first table in the document.

Table table =

doc.MainDocumentPart.Document.Body.Elements<Table>().First();

int maxNumberOfColumns = 5;

//Optional code that determines how many columns are needed in the table

//foreach (TableRow tr in table.Elements<TableRow>())

//{

// if (tr.Elements<TableCell>().Count() > maxNumberOfColumns)

// {

// maxNumberOfColumns = tr.Elements<TableCell>().Count();

// }

//}

// Find the row in the table (add rows and cells if they don't exist).

if (table.Elements<TableRow>().Count() <= rowpos)

{

for (int i = table.Elements<TableRow>().Count() - 1; i < rowpos; i++)

{

var tr = new TableRow();

table.Append(tr);

}

}

TableRow row = table.Elements<TableRow>().ElementAt(rowpos);

for (int j = row.Elements<TableCell>().Count(); j < maxNumberOfColumns; j++)

{

var tc = new TableCell();

row.Append(tc);

}

TableCell cell = row.Elements<TableCell>().ElementAt(cellpos);

// Find the first paragraph in the table cell and add one if necessary.

if (cell.Elements<Paragraph>().Count() == 0)

{

var para = new Paragraph();

cell.Append(para);

}

Paragraph p = cell.Elements<Paragraph>().First();

if (p.InnerText == String.Empty)

{

string newText = text;

p.RemoveAllChildren();

p.AppendChild(new Run(new Text(newText)));

}

// Find the first run in the paragraph.

Run r = p.Elements<Run>().First();

// Set the text for the run.

Text t = r.Elements<Text>().First();

t.Text = text;

}

}

}

}

* Review the code and try to work out what it’s doing.
* Build and run the code and confirm it behaves as you expected.