IronXL.Excel Guide

Version 2019.5.2

IronSoftware 1/23/20 Ahmed Aboelmagd

Contents

Introduction

<-- Text in introduction part -->

And you can visit https://ironsoftware.com/csharp/excel/ for more information.

You can download sample project from GitHub (https://github.com/magedo93/IronSoftware.git)

Document Organization

- Chapter 1 Install IronXL.Excel: this part describes How to install IronXL.Excel to existing project.
- Chapter 2 Basic Operations: this part describes basic operation with excel create or Open workbook, select sheet, select cell save workbook
- Chapter 3 Advanced Sheet Operations: this part describes how to different manipulation capabilities like adding headers or footers, mathematical operations files, and other features.
- Summery
  brief conclusion about what we have learned in this document
- About author
  brief about document author

Chapter 1: Install IronXL.Excel

IronXL.Excel can be installed and used on all of .NET projects type like windows application, ASP.NET MVC and .Net Core Application.

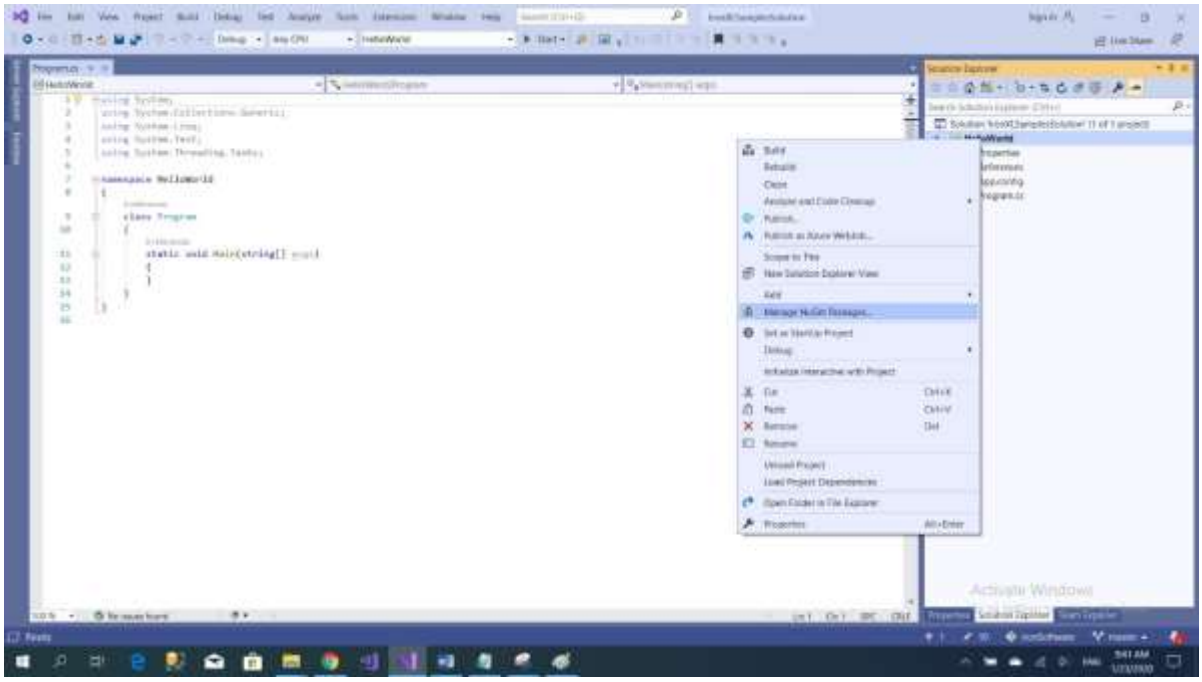
To add IronXL.Excel library to the project we have two ways, from Visual studio editor install using NuGet or command line using package console manager as following: -

Install using NuGet

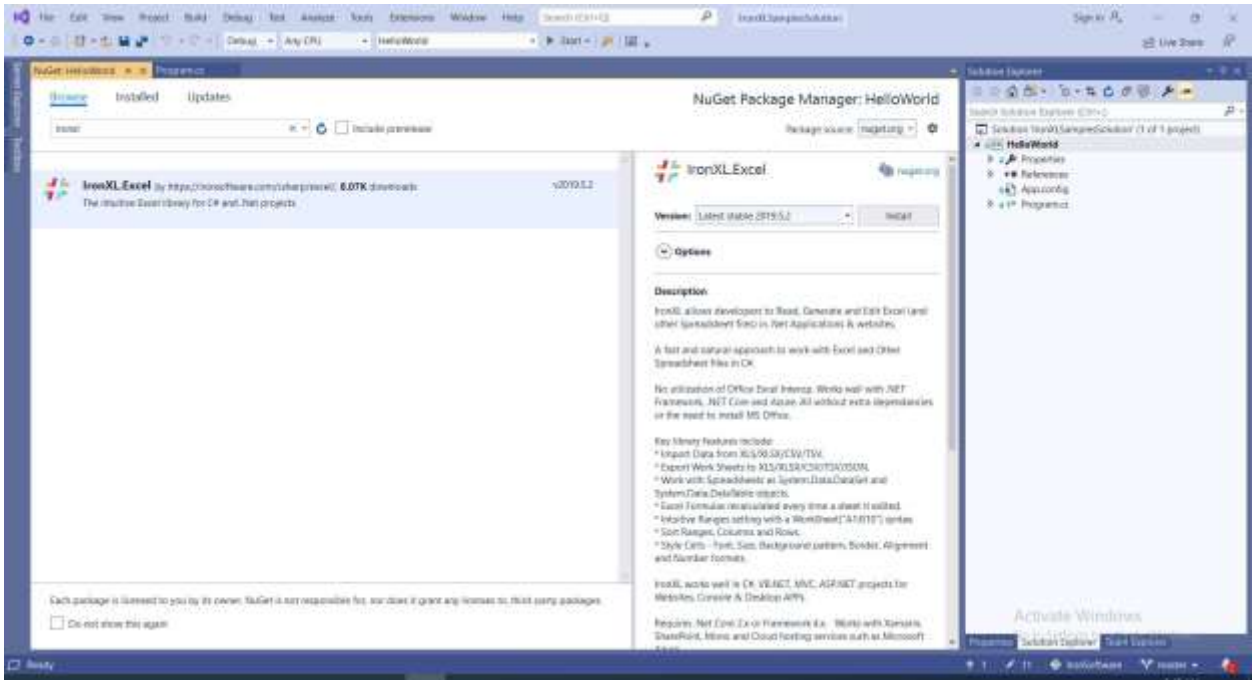
To add IronXL.Excel library to our project using NuGet we can do it using visualized interface (NuGet Package Manager) or by command using Package Manager Console as following: -

Using NuGet Package Manager

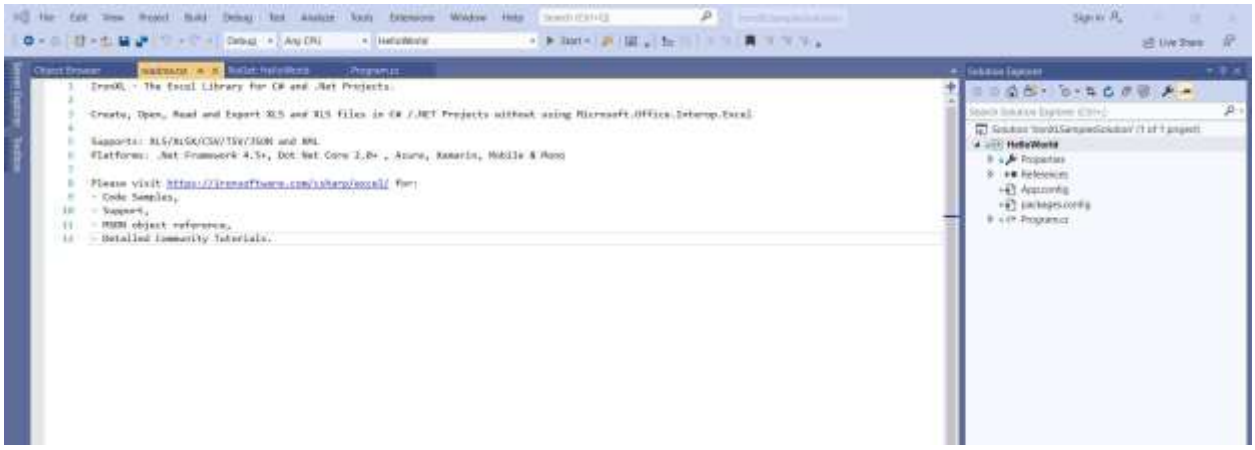
1- Using mouse -> right click on project name -> Select manage NuGet Package



2- From brows tab -> search for IronXL.Excel -> Install

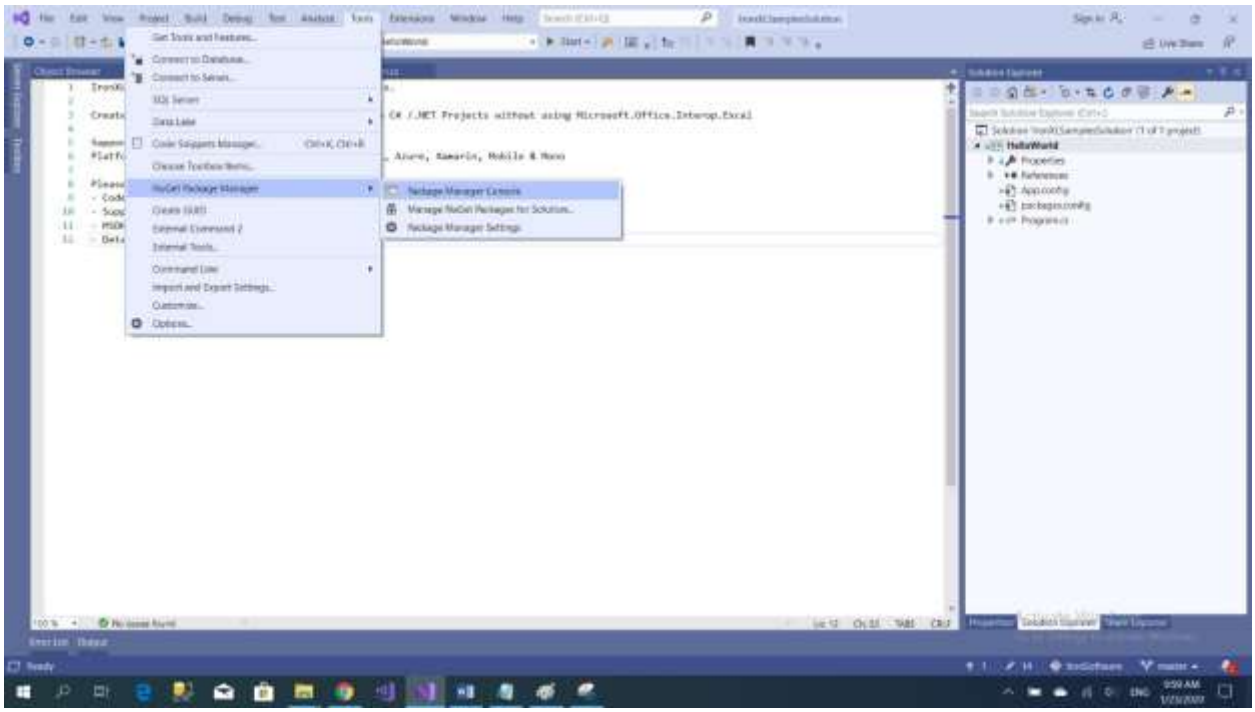


3- And we are Done

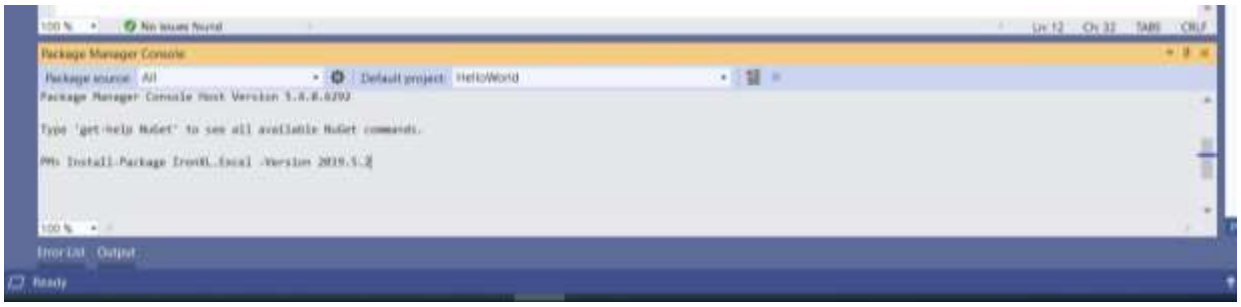


Using NuGet Package Console manager

1- From tools -> NuGet Package Manager -> Package Manager Console



2- Run command -> Install-Package IronXL.Excel -Version 2019.5.2

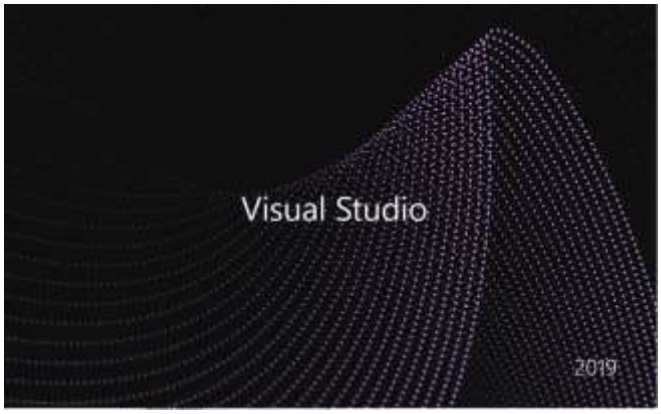


Chapter 2: Basic Operations

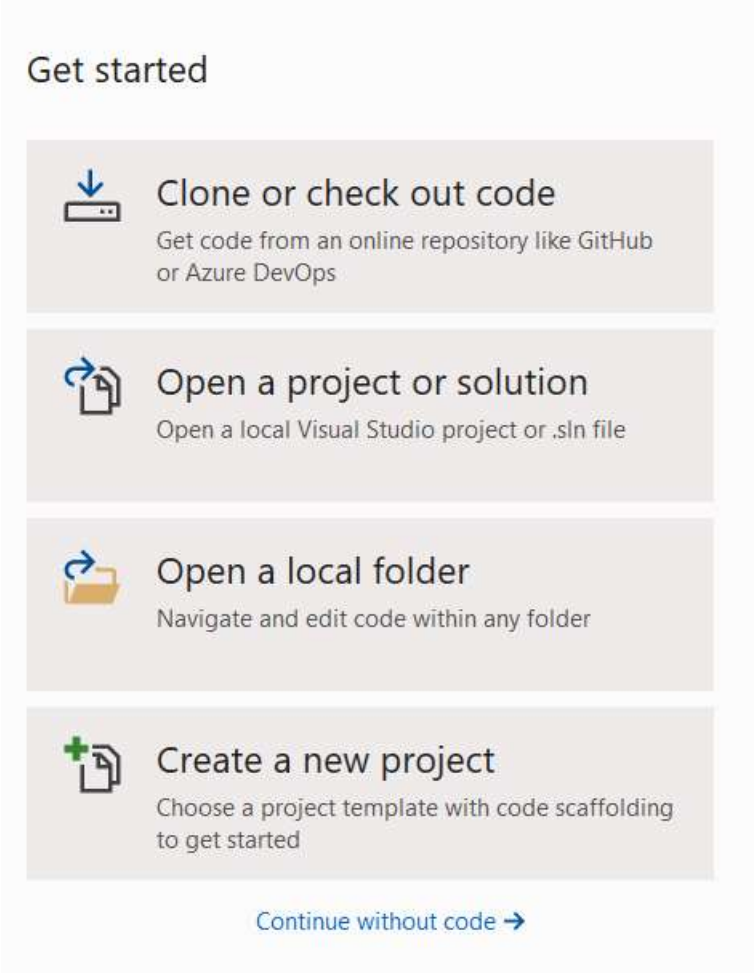
1- Sample: HelloWorld Console Application

Follow coming steps to create HelloWorld Project

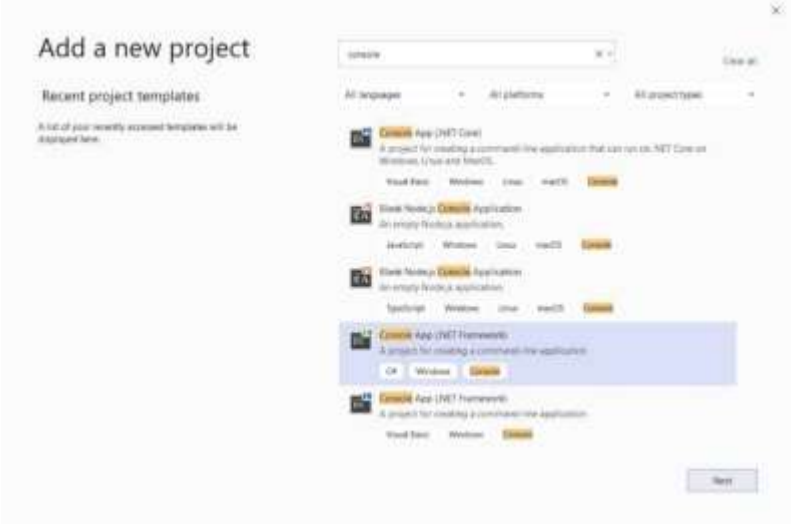
1.1. Open visual studio



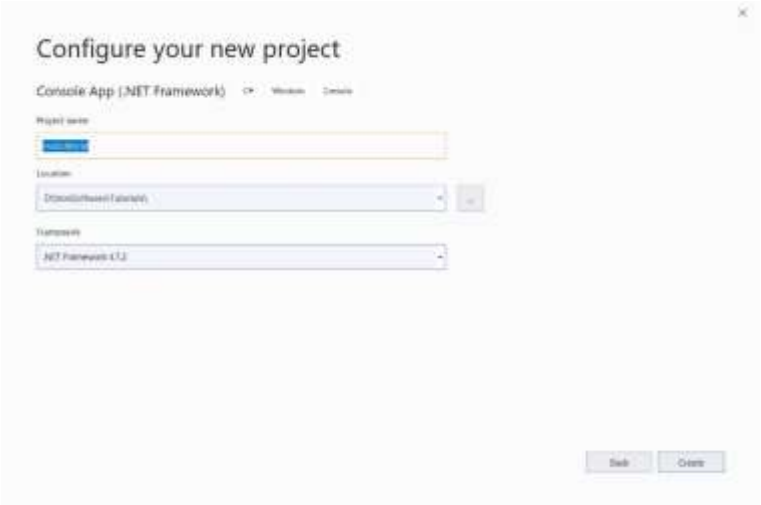
1.2. Choose Create new project



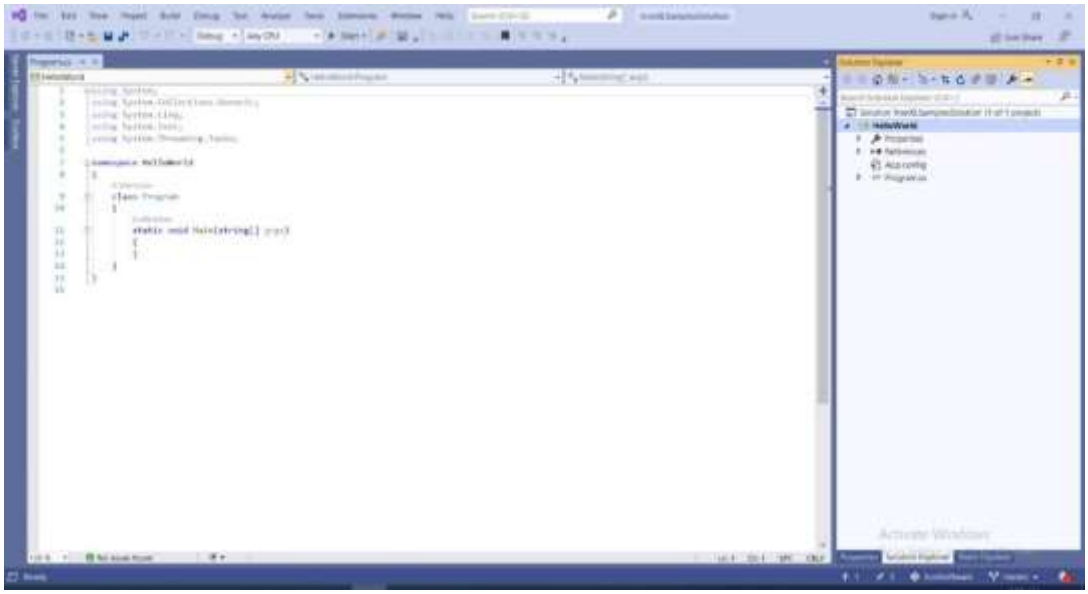
1.3. Choose Console App (.NET framework)



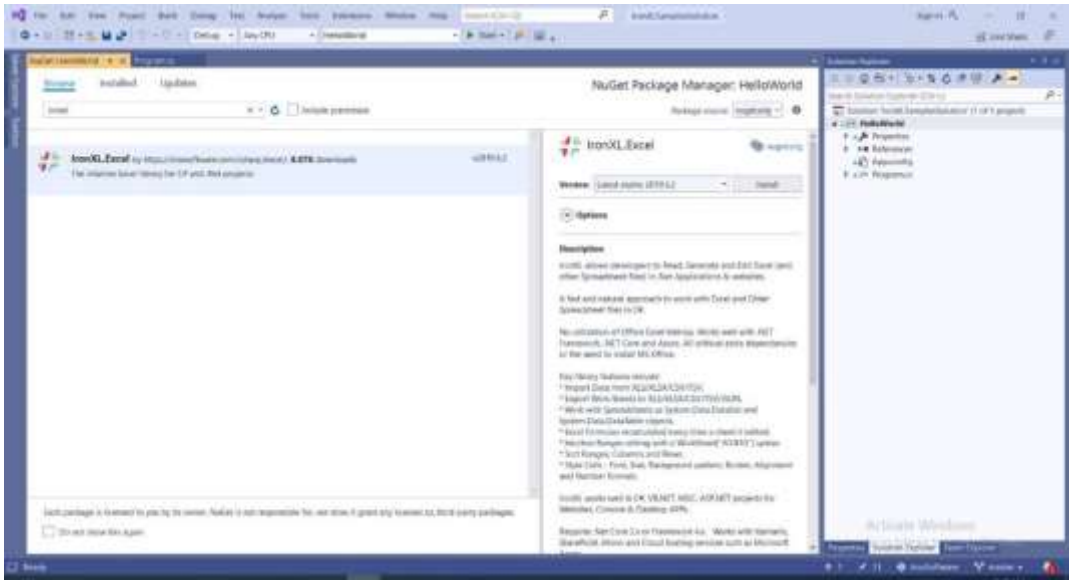
1.4. Give our sample name “HelloWorld” and click create



1.5. Now we have console application created



1.6. Add IronXL.Excel => click install



1.7. Add our first few lines that reads 1st cell in 1st sheet in excel file and print it

```
 static void Main(string[] args)
 {
  var workbook =
IronXL.WorkBook.Load($@"{Directory.GetCurrentDirectory()}\Files\HelloWorld.xlsx");
  var sheet = workbook.WorkSheets.First();
  var cell = sheet["A1"].StringValue;
  Console.WriteLine(cell);
 }
```

2. Create New Excel File

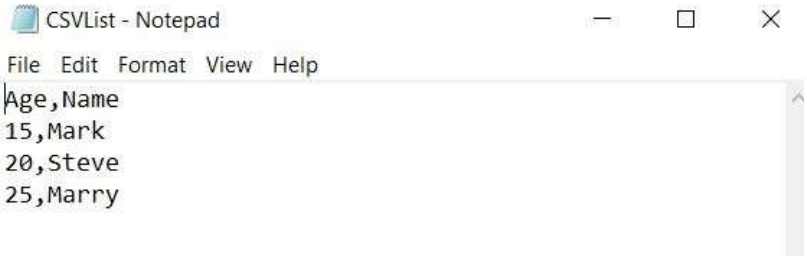
you can create new excel file using IronXL as follow

```
 static void Main(string[] args)
        {
            var newXLFile = WorkBook.Create(ExcelFileFormat.XLSX);
            newXLFile.Metadata.Title = "IronXL New File";
            var newWorkSheet = newXLFile.CreateWorkSheet("1stWorkSheet");
            newWorkSheet["A1"].Value = "Hello World";
            newWorkSheet["A2"].Style.BottomBorder.SetColor("#ff6600");
            newWorkSheet["A2"].Style.BottomBorder.Type =
IronXL.Styles.BorderType.Dashed;
        }
```

3. Open (CSV, XML, JSON List) as Workbook

3.1. Open CSV file

3.1.1. Create new text file and add to its list of names and ages as follow then save it as CSVList.csv



3.1.2. Code snippet should be like this

```
static void Main(string[] args)
{
 var workbook = IronXL.WorkBook.Load($@"{Directory.GetCurrentDirectory()}\Files\CSVList.csv");
 var sheet = workbook.WorkSheets.First();
 var cell = sheet["A1"].StringValue;
 Console.WriteLine(cell);
}
```

3.2. open XML File

create an XML file that contains countries list as following
root element "countries" that have children elements "country" and each country has properties that define the country like code, continent, etc.

```
<?xml version="1.0" encoding="utf-8"?>
<countries xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <country code="ae" handle="united-arab-emirates" continent="asia"
iso="784">United Arab Emirates</country>
            <country code="gb" handle="united-kingdom" continent="europe" alt="England
Scotland Wales GB UK Great Britain Britain Northern" boost="3" iso="826">United
Kingdom</country>
            <country code="us" handle="united-states" continent="north america" alt="US
America USA" boost="2" iso="840">United States</country>
            <country code="um" handle="united-states-minor-outlying-islands" continent="north
america" iso="581">United States Minor Outlying Islands</country>
</countries>
```

3.3. and now copy the following code snippet that open XML as workbook

```
static void Main(string[] args)
 {
   var xmldataset = new DataSet();
   xmldataset.ReadXml($@"{Directory.GetCurrentDirectory()}\Files\CountryList.xml");
   var workbook = IronXL.WorkBook.Load(xmldataset);
   var sheet = workbook.WorkSheets.First();
 }
```

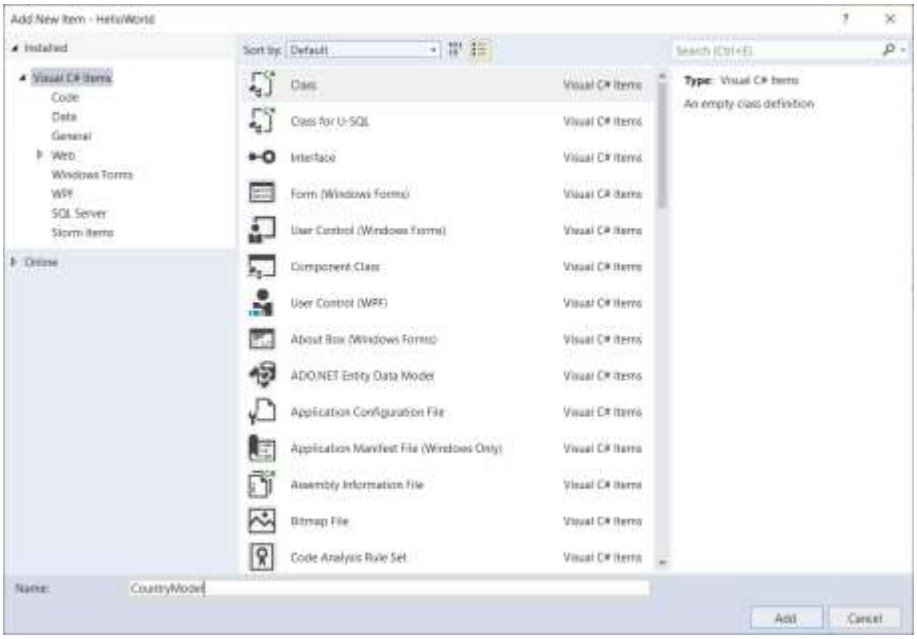
3.4. open JSON List as workbook

create JSON country list as following

```
[
  {
    "name": "United Arab Emirates",
    "code": "AE"
  },
```

```
  {
    "name": "United Kingdom",
    "code": "GB"
  },
  {
    "name": "United States",
    "code": "US"
  },
  {
    "name": "United States Minor Outlying Islands",
    "code": "UM"
  }
]
```

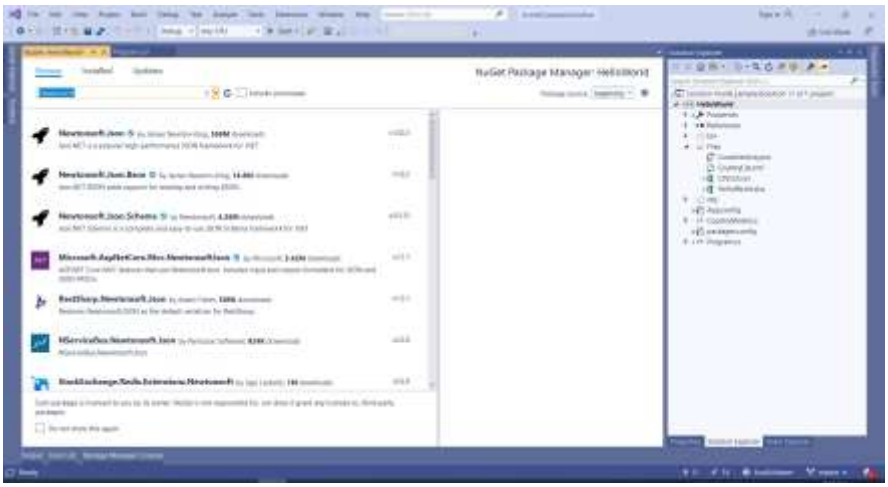
3.4.1. create country model that will map to JSON



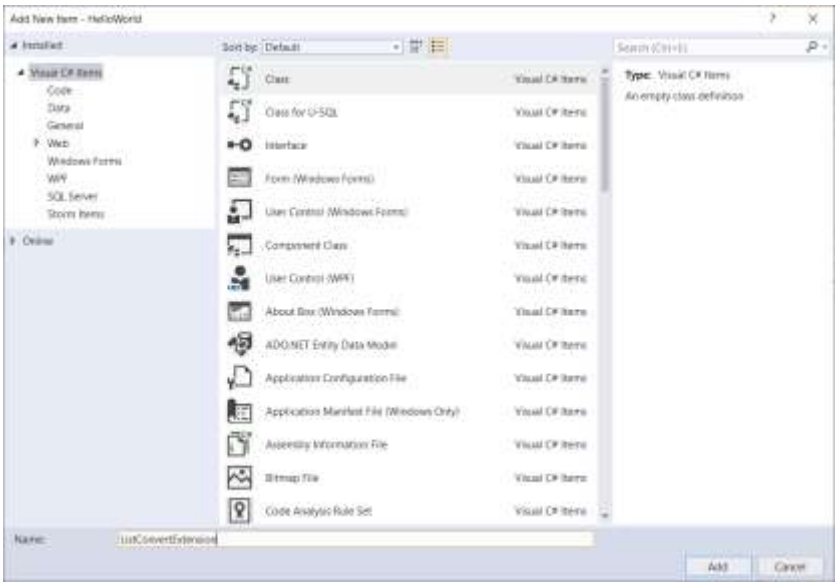
3.4.2. and here is the class code snippet

```
public class CountryModel
    {
        public string name { get; set; }
        public string code { get; set; }
    }
```

3.4.3. Add Newtonsoft library to convert JSON to list of country models



3.4.4. To convert list to dataset we have to create new extension for list as following, first add extension class with the name "ListConvertExtension"



3.4.5. Then add this code snippet to it

```
public static class ListConvertExtension
    {
        public static DataSet ToDataSet<T>(this IList<T> list)
        {
            Type elementType = typeof(T);
            DataSet ds = new DataSet();
            DataTable t = new DataTable();
            ds.Tables.Add(t);

            //add a column to table for each public property on T
            foreach (var propInfo in elementType.GetProperties())
            {
                Type ColType = Nullable.GetUnderlyingType(propInfo.PropertyType) ??
propInfo.PropertyType;

                t.Columns.Add(propInfo.Name, ColType);
            }

            //go through each property on T and add each value to the table
            foreach (T item in list)
```

```
            {
                DataRow row = t.NewRow();

                foreach (var propInfo in elementType.GetProperties())
                {
                    row[propInfo.Name] = propInfo.GetValue(item, null) ?? DBNull.Value;
                }

                t.Rows.Add(row);
            }

            return ds;
        }
    }
```

3.4.6. And finally load this dataset as workbook

```
static void Main(string[] args)
        {
            var jsonFile = new
StreamReader($@"{Directory.GetCurrentDirectory()}\Files\CountriesList.json");
            var countryList =
Newtonsoft.Json.JsonConvert.DeserializeObject<CountryModel[]>(jsonFile.ReadToEnd());
            var xmldataset = countryList.ToDataSet();
            var workbook = IronXL.WorkBook.Load(xmldataset);
            var sheet = workbook.WorkSheets.First();
        }
```

4. Save and Export

We can save or export excel file to multiple files formats like (".xlsx",".csv",".html") using one of the following commands.

4.1. Save to ".xlsx"

To Save to ".xlsx" use saveAs function

```
static void Main(string[] args)
        {
            var newXLFile = WorkBook.Create(ExcelFileFormat.XLSX);
            newXLFile.Metadata.Title = "IronXL New File";
            var newWorkSheet = newXLFile.CreateWorkSheet("1stWorkSheet");
            newWorkSheet["A1"].Value = "Hello World";
            newWorkSheet["A2"].Style.BottomBorder.SetColor("#ff6600");
            newWorkSheet["A2"].Style.BottomBorder.Type = IronXL.Styles.BorderType.Dashed;

            newXLFile.SaveAs($@"{Directory.GetCurrentDirectory()}\Files\HelloWorld.xlsx");
        }
```

4.2. Save to csv ".csv"

To save to ".csv" we can use SaveAsCsv and pass to it 2 parameters 1st parameter the file name and path the 2nd parameter is the delimiter like (",", " | " or ":")

```
newXLFile.SaveAsCsv($@"{Directory.GetCurrentDirectory()}\Files\HelloWorld.csv",delimiter:"|");
```

4.3. Save to JSON ".json"

To save to Json ".json" use SaveAsJson as follow

```
newXLFile.SaveAsJson($@"{Directory.GetCurrentDirectory()}\Files\HelloWorldJSON.json");
```

The result file should be as follow

```
[
  [
    "Hello World"
  ],
  [
    ""
  ]
]
```

4.4. Save to XML ".xml"

To save to xml use SaveAsXml as follow

```
newXLFile.SaveAsXml($@"{Directory.GetCurrentDirectory()}\Files\HelloWorldXML.XML");
```

Result should be like the following

```
<?xml version="1.0" standalone="yes"?>
<_x0031_stWorkSheet>
  <_x0031_stWorkSheet>
    <Column1 xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">Hello World</Column1>
  </_x0031_stWorkSheet>
  <_x0031_stWorkSheet>
    <Column1 xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" />
  </_x0031_stWorkSheet>
</_x0031_stWorkSheet>
```

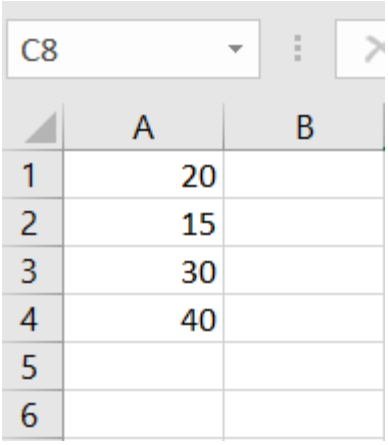
Chapter 3: Advanced Sheet Operations

In this chapter we will go throw common excel functions like (SUM, AVG, Count)

I'll write short code snippets for each function Let's go.

1- Sum Example

Let's find the sum for this list, I created excel file and named it "Sum.xlsx" and add this list of numbers manually




```
var workbook = IronXL.WorkBook.Load($@"{Directory.GetCurrentDirectory()}\Files\Sum.xlsx");
            var sheet = workbook.WorkSheets.First();
            decimal sum = sheet["A2:A4"].Sum();
            Console.WriteLine(sum);
```

2- AVG Example

Using same file, we can get the average as follow: -

```
   var workbook = IronXL.WorkBook.Load($@"{Directory.GetCurrentDirectory()}\Files\Sum.xlsx");
            var sheet = workbook.WorkSheets.First();
            decimal avg = sheet["A2:A4"].Avg();
            Console.WriteLine(avg);
```

3- Count Example

Using same file, we can get the average as follow: -

```
     var workbook = IronXL.WorkBook.Load($@"{Directory.GetCurrentDirectory()}\Files\Sum.xlsx");
            var sheet = workbook.WorkSheets.First();
            decimal count = sheet["A2:A4"].Count();
            Console.WriteLine(count);
```

Summary

In this tutorial, we have learned how to (create, open, save) excel file using different ways and also we learned the basic operations that we can do on the file like getting (Sum, Avg., Count)

Author

Ahmed Aboelmagd is a Full-stack experienced software engineer with 12+ years' experience in IT and Software development, delivered 15+ successful project in many size scales from small to an enterprise for industries like (tourism, educational, manufacturing, etc.)