Excel for UWP

2018.03.07 更新

グレープシティ株式会社

Excel for UWP

目次

Excel for UWP 2
主な特長 3
クイックスタート 4
手順 1:プロジェクトの設定 4
手順 2:C1XLBook へのコンテンツの追加 4-5
手順 3:XLSX ファイルの保存 5-6
手順 4:プログラムの実行 6-7
C1Excel の使い方 8
ドキュメントの作成 8
ワークシート 8-9
行と列 9
セル 9-10
スタイル 10
タスク別ヘルプ 11
ワークブックへのコンテンツの追加 11-13
セルの書式設定 13-16
ワークシートへのページブレークの追加 16-18
ワークブックの計算モードの設定 18-20
小計の作成 20-23

Excel for UWP

Excel for UWP を使用して、データを Excel にエクスポートすることができます。Microsoft Excel がインストールされている必要はありません。この使いやすいコンポーネントを使用して、XLS ファイルと XLSX ファイルを作成およびロードできます。行、列、およびセルで構成される単純なグリッドのように、各シート内のデータにアクセスして変更できます。

主な特長

以下に、**Excel for UWP** の便利な機能をいくつか示します。

- **1つのコマンドを使用したワークブックの保存またはロード**
  Excel for UWP は使いやすく、1つのコマンドを使用してワークブックをロードまたは保存し、シートをグリッドコントロールのように操作できます。
- **CSV、XLS、XLSX のサポート**
  Excel for UWP では、XLS、XLSX（OpenXML 形式）、CSV（カンマ区切り値）の3つのファイル形式を使用できます。OpenXml 形式を使用することで、小さく圧縮されたファイルを保存できます。
- **個々のセル内のデータの読み書き**
  C1XLBook をロードまたは作成したら、単純なグリッドのように各シート内のデータにアクセスできます。たとえば、次のようになります。

  C#
  ```
  XLSheet sheet = C1XLBook.Sheets[0];
  sheet[0, 0].Value = DateTime.Now;
  ```
- **各セルのデータの書式設定**
  各セルに関連付けられた書式には、セルに格納されているデータと同様に簡単にアクセスできます。たとえば、次のようになります。

  C#
  ```
  XLStyle style = new XLStyle(c1XLBook1);
  style.Format = "dd-MM-yyyy";
  style.Font = new Font("Courier New", 14);
  XLSheet sheet = C1XLBook.Sheets[0];
  sheet[0, 0].Value = DateTime.Now;
  sheet[0, 0].Style = style;
  ```
- **セルの数式の追加**
  Excel for UWP は、セルの数式とバイナリ解析を完全にサポートしています。XLCell.Formula プロパティを使用して、セルの数式を指定できます。
- **グループ化と小計**
  行および列の小計を計算できます。コードでアウトラインレベルのグループ化を宣言して、合計や小計を最適に表示できます。

クイックスタート

このクイックスタートガイドでは、**Excel for UWP** のいくつかの機能について詳しく説明します。このクイックスタートでは、プロジェクトに **C1XLBook** を追加し、ワークブックに書式設定されたデータを追加し、XLS ファイルを保存してから開きます。

手順 1:プロジェクトの設定

この手順では、新しいプロジェクトを作成し、C1.UWP.Excelアセンブリへの参照を追加します。

1. 新しい Windows ストアプロジェクトを作成します。プロジェクトが開いたら、**MainPage.xaml** ファイルをダブルクリックして開きます。
2. ソリューションエクスプローラで**[参照]**を右クリックし、リストから**[参照の追加]**を選択します。
   a. C1.UWP.Excel.dll を参照します。
   b. **[OK]**をクリックして、アセンブリ参照をアプリケーションに追加します。
3. XAML ビューで、<Grid> </Grid> タグの間にカーソルを置きます。
4. ページに2つの標準 Button コントロールと1つの標準 TextBox コントロールを追加します。
   a. 最初のボタンのマークアップを次のように編集します。

   XAML マークアップ
   ```
   <Button x:Name="HelloButton" Content="Click Hello" />
   ```
   b. 2つ目のボタンのマークアップを次のように編集します。

   XAML マークアップ
   ```
   <Button x:Name="SaveButton" Content="保存" />
   ```
   c. TextBox コントロールのマークアップを次のように編集します。

   XAML マークアップ
   ```
   <TextBox
      Name="_tbContent"
      Text="Empty"
      IsReadOnly="True"
      AcceptsReturn="True"
      FontFamily="Courier New"
      Background="White" Margin="465,10,242,722" />
   ```
5. HelloButton用の**HelloButton_Click** というイベントを作成して、MainPage.xaml のコードビューに切り替えます。これにより、コードに **HelloButton_Click** イベントも追加されます。
6. デザインビューに切り替え、SaveButton をダブルクリックして、**SaveButton_Click** イベントをコードに追加します。
7. **using**(C#)文をフォームの先頭のコードに追加します。これで、C1.Xaml.Excel 名前空間内のすべての名前を使用できます。

   C#
   ```
   using C1.Xaml.Excel;
   using Windows.UI;
   ```

これで、C1XLBook にコンテンツを追加できます。

手順 2:C1XLBook へのコンテンツの追加

この手順では、プロジェクトを設定するコードを追加し、**HelloButton_Click** イベントを編集します。

1. C1XLBook を定義するには、次のコードを MainPage クラスのすぐ下に追加します。

   C#
   ```
   C1XLBook _book;
   ```
2. 次のコードを **InitializeComponent()** メソッドに追加して、新しい C1XLBook とテキストを TextBox コントロールに追加します。

   C#
   ```
   _book = new C1XLBook();
   _tbContent.Text = "空白のブック";
   ```
3. MainPageコンストラクタの後に、**RefreshView()** メソッドを定義する次のコードを追加します。。

   C#
   ```
   void RefreshView()
   {
   ```

```
}
```

4. 次に、**HelloButton_Click** イベントを見つけます。次のコードを追加して新しいワークブックを作成し、デフォルトで作成されたシートを取得します。データのスタイルをいくつか作成し、中身を書き込み、セルの書式を設定します。

C# コードの書き方

C#

```
// 手順1:新しいワークブックを作成します
_book = new C1XLBook();
// 手順2:デフォルトで作成されたシートを取得して名前を付けます
XLSheet sheet = _book.Sheets[0];
sheet.Name = "Hello World";
// 手順3:奇数と偶数のスタイルを作成します
XLStyle styleOdd = new XLStyle(_book);
styleOdd.Font = new XLFont("Tahoma", 9, false, true);
styleOdd.ForeColor = Color.FromArgb(255, 0, 0, 255);
XLStyle styleEven = new XLStyle(_book);
styleEven.Font = new XLFont("Tahoma", 9, true, false);
styleEven.ForeColor = Color.FromArgb(255, 255, 0, 0);
// 手順4:中身を書き込み、セルの書式を設定します
for (int i = 0; i < 100; i++)
{
    XLCell cell = sheet[i, 0];
    cell.Value = i + 1;
    cell.Style = ((i + 1) % 2 == 0) ? styleEven : styleOdd;
}
// 手順5:ユーザーがファイルを保存できるようにします
_tbContent.Text = "'Hello World' ブックが作成され、保存可能になりました。";
RefreshView();
```

手順 3:XLSX ファイルの保存

次のように、Excel ワークブックを保存するコードを追加します。**SaveButton** をクリックすると、作成したプロジェクトを任意の場所に保存できます。

1. 次のようなコードになるように、**SaveButton_Click** イベントを編集します。

C# コードの書き方

C#

```
async void SaveButton_Click(object sender, RoutedEventArgs e)
        {
        }
```

2. 次のように、Excel ワークブックの保存を処理するコードを **SaveButton_Click** イベントに挿入します。

C# コードの書き方

C#

```
Debug.Assert(_book != null);

var picker = new Windows.Storage.Pickers.FileSavePicker();
```

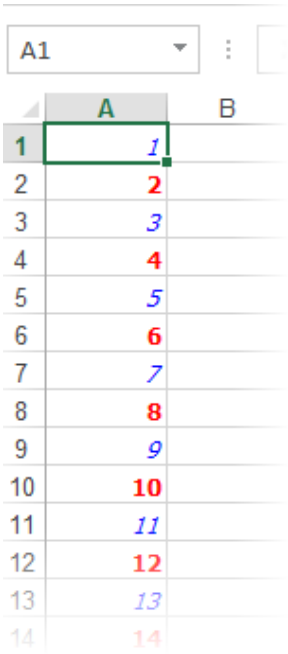
```
picker.SuggestedStartLocation =
Windows.Storage.Pickers.PickerLocationId.DocumentsLibrary;
picker.FileTypeChoices.Add("Open XML Excel ファイル", new List < string > ()
{
  ".xlsx"
});
picker.FileTypeChoices.Add("BIFF Excel ファイル", new List < string > ()
{
  ".xls"
});
picker.SuggestedFileName = "新しいブック";

var file = await picker.PickSaveFileAsync();
if (file != null)
{
  try
  {
    // 手順1:ファイルを保存します
    var fileFormat = Path.GetExtension(file.Path).Equals(".xls") ?
FileFormat.OpenXmlTemplate : FileFormat.OpenXml;
    await _book.SaveAsync(file, fileFormat);
    // 手順2:ユーザーフィードバック
    _tbContent.Text = string.Format("ファイルが保存されました: {0}.", file.Path);
    RefreshView();
  } catch (Exception x)
  {
    _tbContent.Text = string.Format("EXCEPTION: {0}", x.Message);
  }
}
```

手順 4:プログラムの実行

[F5]キーを押してアプリケーションを実行します。

1. **[Hello World]**ボタンをクリックして、Excel ファイル「Hello World」を作成します。TextBox に、このファイルが作成されたことを示すメッセージが表示されます。

2. **[保存]**ボタンをクリックします。**[名前を付けて保存]**画面が表示されます。
3. ワークブックのファイル名を入力し、**[保存]**をクリックします。
4. Excel でブックを開きます。次の画像のように表示されます。



おめでとうございます。これで、**Excel for UWP** クイックスタートは終了です。

C1Excel の使い方

以下のトピックでは、XLSX ファイルの作成方法について説明すると共に、ファイルを構成するコンポーネント（ワークシート、行、列、セル、スタイル）の作成に使用される **Excel for UWP** の主要なクラスについて説明します。

ドキュメントの作成

Excel for UWP を使用して新しい XLSX ファイルを作成するには、次の3つの手順を実行する必要があります。

1. C1.UWP.Excel.dll への参照を追加し、C1XLBook を作成します。
2. コンテンツをシートに追加します。各シートには、**XLCell.Value** プロパティと **XLCell.Style** プロパティを持ついくつかのセル（**XLCell** オブジェクト）が格納されます。
3. **C1XLBook.SaveAsync** メソッドを使用して、ブックをファイルに保存します。

このコードは、最初に新しいワークブックを作成し、次に自動的に作成される XLSheet を取得します。その後、中身が書き込まれてセルが書式設定される前に、偶数と奇数に適用されるスタイルを作成します。

XLSheet オブジェクト内のインデクサは、必要に応じてセルを自動的に作成します。これで、作成するワークシートに簡単に値を入力できます。シートのサイズを調べる場合は、シートの **Rows.Count** プロパティと **Columns.Count** プロパティを使用します。

手順3で適用するスタイルは、偶数の数値が太字の赤色の文字で、奇数が青の斜体で表示されるシートを作成します。

C# コードの書き方

C#

```
private void HelloButton_Click(object sender, RoutedEventArgs e)
        {
            // 手順1:新しいワークブックを作成します
            _book = new C1XLBook();
            // 手順2:デフォルトで作成されたシートを取得して名前を付けます
            XLSheet sheet = _book.Sheets[0];
            sheet.Name = "Hello World";
            // 手順3:奇数と偶数のスタイルを作成します
            XLStyle styleOdd = new XLStyle(_book);
            styleOdd.Font = new XLFont("Tahoma", 9, false, true);
            styleOdd.ForeColor = Color.FromArgb(255, 0, 0, 255);
            XLStyle styleEven = new XLStyle(_book);
            styleEven.Font = new XLFont("Tahoma", 9, true, false);
            styleEven.ForeColor = Color.FromArgb(255, 255, 0, 0);
            // 手順4:中身を書き込み、セルの書式を設定します
            for (int i = 0; i < 100; i++)
            {
                XLCell cell = sheet[i, 0];
                cell.Value = i + 1;
                cell.Style = ((i + 1) % 2 == 0) ? styleEven : styleOdd;
            }
            // 手順5:ユーザーがファイルを保存できるようにします
            _tbContent.Text = "'Hello World' ブックが作成され、保存可能になりました。";
            RefreshView();
        }
```

ワークシート

ワークシートは、Excel ファイルに含まれる個々のグリッドです。ワークシートは **XLSheet** オブジェクトで表され、これには**C1XLBook** クラスの **Sheets** プロパティからアクセスできます。各シートには名前があり、一連の行と列から成ります。各セルには、行インデックスと列インデックスを取る **XLSheet** インデクサを使用してアクセスできます。

XLSheet オブジェクトの **XLSheet.Rows** コレクションと **XLSheet.Columns** コレクションは、それぞれのインデクサを使用する際に自動的に拡張されます。たとえば、次のコードを記述した場合、このシートに 1001 行まで行がなければ、いくつかの新しい行が自動的に追加され、有効な行が返されます。**XLColumn** と **XLCell** のインデクサも同様です。これは、.NET の大部分のコレクションインデクサの動作とは異なりますが、こうすることで、**XLSheet** オブジェクトを簡単に作成し、値を入力できるようになります。

Visual Basic コードの書き方

Visual Basic

```
Dim sheet As XLSheet = book.Sheets(0)
Dim row As XLRow = sheet.Rows(1000)
```

C# コードの書き方

C#

```
XLSheet sheet = book.Sheets[0];
XLRow row = sheet.Rows[1000];
```

行と列

XLSheet オブジェクトには、シート内の個別の行と列を公開する行と列のコレクションが含まれています。公開された **XLRow** オブジェクトと **XLColumn** オブジェクトを使用して、シート内の各行と列のサイズ（列の幅と行の高さ）、表示/非表示、およびスタイルを割り当てることができます。これらの値を割り当てない場合は、シートのデフォルト値が使用されます（**XLSheet.DefaultRowHeight** プロパティと **XLSheet.DefaultColumnWidth** プロパティを参照）。

XLRow オブジェクトと **XLColumn** オブジェクトのデフォルトのサイズは –1 です。この場合は、シートのデフォルト値が使用されます。

セル

XLSheet オブジェクトにはセルも含まれています。セルには、行インデックスと列インデックスを取るインデクサを使用してアクセスできます。セルは、セル値とスタイルを持つ **XLCell** オブジェクトで表されます。

行と列の場合と同様に、セルインデクサもシートを自動的に拡張します。たとえば、次のように記述したとします。

Visual Basic コードの書き方

Visual Basic

```
Dim cell As XLCell = sheet(10, 10)
```

C# コードの書き方

C#

```
XLCell cell = sheet[10,10];
```

シートに含まれる行と列の数がそれぞれ 11 未満の場合は、いくつかの行と列が追加され、有効な **XLCell** オブジェクトが返されます。

シートは自動的に拡張されるため、このインデクサが **null** 参照を返すことはありません。シートに特定のセルが存在するかどうかを確認したいが、不要なセルを作成しないようにするには、インデクサの代わりにシートの **XLSheet.GetCell** メソッドを使用します。

XLCell オブジェクトには、セルのコンテンツを格納する **XLCell.Value** プロパティがあります。このプロパティは **object** 型で、文字列、数値、ブール値、DateTime、または null オブジェクトを格納できます。その他の型のオブジェクトは、Excel ファイルに保存できません。

また、**XLCell** オブジェクトには、セルの外観を定義する **XLCell.Style** プロパティがあります。**Style** プロパティを **null** に設定した場合、セルはデフォルトのスタイルで表示されます。それ以外の場合は、このプロパティにセルの外観（フォント、配置、色、書式設定など）を定義する **XLStyle** オブジェクトを設定する必要があります。

スタイル

XLStyle クラスは、シート内のセル、行、または列の外観を定義します。**XLStyle** には、セル値の表示に使用されるフォント、配置、色、書式設定などのスタイル要素を指定するプロパティが含まれます。すべての **XLStyle** オブジェクトですべてのスタイル要素を定義する必要はありません。たとえば、**XLStyle** で書式設定のみを指定した場合、セルは指定された書式設定で表示されますが、他のスタイル要素（フォント、配置など）にはデフォルト設定が適用されます。

タスク別ヘルプ

タスク別ヘルプは、Visual Studio プログラミングにある程度慣れていることを前提としています。[ヘルプ]に示される手順に従って作業を進めるだけで、**Excel for UWP** のさまざまな機能を具体的に紹介するプロジェクトを作成しながら、**Excel for UWP** の主要な機能と特長を理解できます。

また、タスク別ヘルプトピックでは、新しい Windows ストアプロジェクトが既に作成されていること、およびコードに適切なディレクティブ（**C#** の場合は **using** C1.Xaml.Excel、**Visual Basic** の場合は **Imports C1.Xaml.Excel**）が追加されていることを前提としています。

ワークブックへのコンテンツの追加

新しいワークブックを作成し、最初の 10 個のセルに値を追加するには、次の手順に従います。

XAML ビュー

1. ソリューションエクスプローラで**[参照]**を右クリックし、リストから**[参照の追加]**を選択します。
   a. C1.UWP.Excel.dll を参照します。
   b. **[OK]**をクリックして、アセンブリ参照をアプリケーションに追加します。
2. XAML ビューで、`<Grid> </Grid>` タグの間にカーソルを置きます。
3. ページに2つの標準 Button コントロールと1つの標準 **TextBox** コントロールを追加します。
   a. 最初のボタンのマークアップを次のように編集します。

   XAML マークアップ
   ```
   <Button x:Name="HelloButton" Content="Click Hello" />
   ```
   b. 2つ目のボタンのマークアップを次のように編集します。

   XAML マークアップ
   ```
   <Button x:Name="SaveButton" Content="保存..." />
   ```
   c. TextBox コントロールのマークアップを次のように編集します。

   XAML マークアップ
   ```
   <TextBox
      Name="_tbContent"
      Text="空白"
      IsReadOnly="True"
      AcceptsReturn="True"
      FontFamily="Courier New"
      Background="White" Margin="465,10,242,722" />
   ```
4. HelloButton用の**HelloButton_Click** というイベントを作成して、MainPage.xaml のコードビューに切り替えます。これにより、コードに **HelloButton_Click** イベントも追加されます。
5. デザインビューに切り替え、SaveButton をダブルクリックして、**SaveButton_Click** イベントをコードに追加します。これで、コードビューが開きます。

コードビュー

1. ページの先頭に using 文を追加します。

   C#
   ```
   using C1.Xaml.Excel;
   ```
2. 次のようなコードを **MainPage** クラスに追加します。

   C#
   ```
   public sealed partial class MainPage : Page
   ```

```
    {
        C1XLBook _book;
    }
```

3. **InitializeComponent()** メソッドに次のコードを追加して、**C1XLBook** を作成します。

C#

```
_book = new C1XLBook();
```

4. RefreshView() メソッドを追加します。後でこのメソッドをコード内で呼び出します。

C#

```
void RefreshView()
        {
        }
```

5. デフォルトのシートを取得して、名前を付けます。
6. 奇数と偶数のスタイルを作成します。
7. 最初の 10 個のセルに値を追加します。

C# コードの書き方

C#

```
            // 手順1:新しいワークブックを作成します
            _book = new C1XLBook();
            // 手順2:デフォルトで作成されたシートを取得して名前を付けます
            XLSheet sheet = _book.Sheets[0];
            sheet.Name = "Hello World";
            // 手順3:奇数と偶数のスタイルを作成します
            XLStyle styleOdd = new XLStyle(_book);
            styleOdd.Font = new XLFont("Tahoma", 9, false, true);
            styleOdd.ForeColor = Color.FromArgb(255, 0, 0, 255);
            XLStyle styleEven = new XLStyle(_book);
            styleEven.Font = new XLFont("Tahoma", 9, true, false);
            styleEven.ForeColor = Color.FromArgb(255, 255, 0, 0);
            // 手順4:中身を書き込み、セルの書式を設定します
            for (int i = 0; i < 100; i++)
            {
                XLCell cell = sheet[i, 0];
                cell.Value = i + 1;
                cell.Style = ((i + 1) % 2 == 0) ? styleEven : styleOdd;
            }
```

8. Excel で開くことができるように、ワークブックを保存します。

C# コードの書き方

C#

```
async void SaveButton_Click(object sender, RoutedEventArgs e)
        {
            Debug.Assert(_book != null);
            var picker = new Windows.Storage.Pickers.FileSavePicker();
            picker.SuggestedStartLocation =
Windows.Storage.Pickers.PickerLocationId.DocumentsLibrary;
            picker.FileTypeChoices.Add("Open XML Excel ファイル", new List() {
".xlsx" });
```

```
            picker.FileTypeChoices.Add("BIFF Excel ファイル", new List() { ".xls"
});
            picker.SuggestedFileName ="新しいブック";
            var file = await picker.PickSaveFileAsync();
            if (file != null)
            {
                try
                {
                    // 手順1:ファイルを保存します
                    var fileFormat = Path.GetExtension(file.Path).Equals(".xls")
? FileFormat.Biff8 : FileFormat.OpenXml;
                    await _book.SaveAsync(file, fileFormat);
                    // 手順2:ユーザーフィードバック
                    _tbContent.Text = string.Format("ファイルを保存しました : {0}.",
file.Path);
                    RefreshView();
                }
                catch (Exception x)
                {
                    _tbContent.Text = string.Format("例外 : {0}", x.Message);
                }
            }
        }
```

セルの書式設定

ブックのセルを書式設定するには、次の手順に従います。

XAML ビュー

1. ソリューションエクスプローラで**[参照]**を右クリックし、リストから**[参照の追加]**を選択します。
   a. C1.UWP.Excel.dll を参照します。
   b. **[OK]**をクリックして、アセンブリ参照をアプリケーションに追加します。
2. XAML ビューで、<Grid> </Grid> タグの間にカーソルを置きます。
3. ページに2つの標準 Button コントロールと1つの標準 **TextBox** コントロールを追加します。
   a. 最初のボタンのマークアップを次のように編集します。

   XAML マークアップ

   ```
   <Button x:Name="HelloButton" Content="Click Hello" />
   ```

   b. 2つ目のボタンのマークアップを次のように編集します。

   XAML マークアップ

   ```
   <Button x:Name="SaveButton" Content="保存..." />
   ```

   c. TextBox コントロールのマークアップを次のように編集します。

   XAML マークアップ

   ```
   <TextBox
      Name="_tbContent"
      Text="空白"
      IsReadOnly="True"
      AcceptsReturn="True"
      FontFamily="Courier New"
   ```

```
    Background="White" Margin="465,10,242,722" />
```

4. HelloButton用の**HelloButton_Click** というイベントを作成して、MainPage.xaml のコードビューに切り替えます。これにより、コードに **HelloButton_Click** イベントも追加されます。
5. デザインビューに切り替え、SaveButton をダブルクリックして、**SaveButton_Click** イベントをコードに追加します。これで、コードビューが開きます。

コードビュー

1. ページの先頭に using 文を追加します。

C#
```
using C1.Xaml.Excel;
```

2. 次のようなコードを **MainPage** クラスに追加します。

C#
```
public sealed partial class MainPage : Page
    {
        C1XLBook _book;
    }
```

3. **InitializeComponent()** メソッドに次のコードを追加して、**C1XLBook** を作成します。

C#
```
_book = new C1XLBook();
```

4. RefreshView() メソッドを追加します。後でこのメソッドをコード内で呼び出します。

C#
```
void RefreshView()
        {
        }
```

5. デフォルトで作成されたシートを取得して、名前を付けます。
6. ワークブックにコンテンツを追加し、新しいスタイルを作成して、セルにそのスタイルを適用します。

C# コードの書き方

C#
```
 private void HelloButton_Click(object sender, RoutedEventArgs e)
        {
            // 手順1:新しいワークブックを作成します
            _book = new C1XLBook();
            // 手順2:デフォルトで作成されたシートを取得して名前を付けます
            XLSheet sheet = _book.Sheets[0];
            sheet.Name = "Hello World";
            // 手順3:前景色と背景色のプロパティを設定し、いくつかの書式をセルに追加します
            XLStyle style1 = new XLStyle(_book);
            style1.ForeColor = Colors.Yellow;
            style1.BackColor = Colors.Blue;
            style1.Format = "$ .00";
            // 手順4:中身を書き込み、セルの書式を設定します
            int i;
                for (i = 0; i <= 9; i++)
                {
                    sheet[i, 0].Value = i + 1;
                    sheet[i, 0].Style = style1;
```

```
                }
```

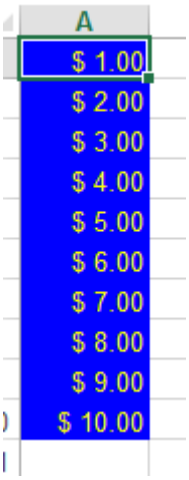
7. ワークブックを保存します。

C# コードの書き方

C#

```
async void SaveButton_Click(object sender, RoutedEventArgs e)
        {
         Debug.Assert(_book != null);
            var picker = new Windows.Storage.Pickers.FileSavePicker();
            picker.SuggestedStartLocation =
Windows.Storage.Pickers.PickerLocationId.DocumentsLibrary;
            picker.FileTypeChoices.Add("Open XML Excel ファイル", new List() {
".xlsx" });
            picker.FileTypeChoices.Add("BIFF Excel ファイル", new List() { ".xls"
});
            picker.SuggestedFileName ="新しいブック";
            var file = await picker.PickSaveFileAsync();
            if (file != null)
            {
                try
                {
                    // 手順1:ファイルを保存します
                    var fileFormat = Path.GetExtension(file.Path).Equals(".xls")
? FileFormat.Biff8 : FileFormat.OpenXml;
                    await _book.SaveAsync(file, fileFormat);
                    // 手順2:ユーザーフィードバック
                    _tbContent.Text = string.Format("ファイルを保存しました : {0}.",
file.Path);
                    RefreshView();
                }
                catch (Exception x)
                {
                    _tbContent.Text = string.Format("例外 : {0}", x.Message);
                }
            }
        }
```

保存したファイルを開くと、次の図のように表示されます。



ワークシートへのページブレークの追加

XLColumn.PageBreak プロパティと **XLRow.PageBreak** プロパティを使用して、OpenXML（.xlsx）形式のファイルの行および列にページブレークを簡単に追加できます。

XAML ビュー

1. ソリューションエクスプローラで**[参照]**を右クリックし、リストから**[参照の追加]**を選択します。
   a. C1.UWP.Excel.dll を参照します。
   b. **[OK]**をクリックして、アセンブリ参照をアプリケーションに追加します。
2. XAML ビューで、`<Grid> </Grid>` タグの間にカーソルを置きます。
3. ページに2つの標準 Button コントロールと1つの標準 **TextBox** コントロールを追加します。
   a. 最初のボタンのマークアップを次のように編集します。

   XAML マークアップ
   ```
   <Button x:Name="HelloButton" Content="Click Hello" />
   ```
   b. 2つ目のボタンのマークアップを次のように編集します。

   XAML マークアップ
   ```
   <Button x:Name="SaveButton" Content="保存..." />
   ```
   c. TextBox コントロールのマークアップを次のように編集します。

   XAML マークアップ
   ```
   <TextBox
      Name="_tbContent"
      Text="空白"
      IsReadOnly="True"
      AcceptsReturn="True"
      FontFamily="Courier New"
      Background="White" Margin="465,10,242,722" />
   ```
4. HelloButton用の**HelloButton_Click** というイベントを作成して、MainPage.xaml のコードビューに切り替えます。これにより、コードに **HelloButton_Click** イベントも追加されます。
5. デザインビューに切り替え、SaveButton をダブルクリックして、**SaveButton_Click** イベントをコードに追加します。これで、コードビューが開きます。

コードビュー

1. ページの先頭に using 文を追加します。

   C#

```
using C1.Xaml.Excel;
```

2. 次のようなコードを **MainPage** クラスに追加します。

```
C#
public sealed partial class MainPage : Page
    {
        C1XLBook _book;
    }
```

3. **InitializeComponent()** メソッドに次のコードを追加して、**C1XLBook** を作成します。

```
C#
_book = new C1XLBook();
```

4. RefreshView() メソッドを追加します。後でこのメソッドをコード内で呼び出します。

```
C#
void RefreshView()
        {
        }
```

5. いくつかのテキスト値およびページブレークを追加します。

C# コードの書き方

```
C#
private void HelloButton_Click(object sender, RoutedEventArgs e)
        {
            // 手順1:新しいワークブックを作成します
            _book = new C1XLBook();
           // テキスト値およびページブレークを追加します
           _book.Sheets[0][2, 3].Value = "page1";
           _book.Sheets[0].Rows[2].PageBreak = true;
           _book.Sheets[0][0, 1].Value = "test1";
           _book.Sheets[0][0, 2].Value = "test2";
          _book.Sheets[0].Columns[1].PageBreak = true;
           _book.Sheets[0][3, 3].Value = "page2";

            // 手順5:ユーザーがファイルを保存できるようにします
            _tbContent.Text = "'Hello World' ブックが作成され、保存可能になりました。";
            RefreshView();
        }
```

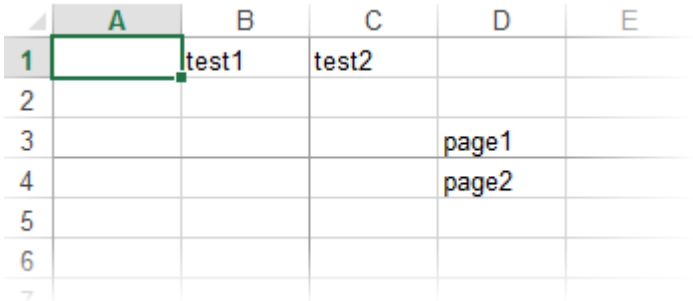
6. ワークブックを保存します。

C# コードの書き方

```
C#
async void SaveButton_Click(object sender, RoutedEventArgs e)
        {
            Debug.Assert(_book != null);
            var picker = new Windows.Storage.Pickers.FileSavePicker();
            picker.SuggestedStartLocation =
Windows.Storage.Pickers.PickerLocationId.DocumentsLibrary;
            picker.FileTypeChoices.Add("Open XML Excel ファイル", new List<string>
() { ".xlsx" });
```

```
            picker.FileTypeChoices.Add("BIFF Excel ファイル", new List<string>() {
".xls" });
            picker.SuggestedFileName ="新しいブック";
            var file = await picker.PickSaveFileAsync();
            if (file != null)
            {
                try
                {
                    // 手順1:ファイルを保存します
                    var fileFormat = Path.GetExtension(file.Path).Equals(".xls")
? FileFormat.Biff8 : FileFormat.OpenXml;
                    await _book.SaveAsync(file, fileFormat);
                    // 手順2:ユーザーフィードバック
                    _tbContent.Text = string.Format("ファイルを保存しました : {0}.",
file.Path);
                    RefreshView();
                }
                catch (Exception x)
                {
                    _tbContent.Text = string.Format("例外 : {0}", x.Message);
                }
            }
        }
```

アプリケーションを実行すると、作成したファイルを保存できるようになります。Excel で、**[ページレイアウト]**タブを選択し、**[枠線]**セクションにある**[印刷]**チェックボックスをオンにします。ワークシートは次のように表示されます。



ワークブックの計算モードの設定

C1XLBook.CalculationMode プロパティは、ワークブック内のすべての数式の計算モードを指定します。**CalculationMode** 列挙は、**Manual**(手作業で計算を実行する)、**Auto**(自動的に計算を実行する)、**AutoNoTable** (テーブル以外は計算を実行する)の3つのオプションを提供します。

XAML ビュー

1. ソリューションエクスプローラで**[参照]**を右クリックし、リストから**[参照の追加]**を選択します。
   a. C1.UWP.Excel.dll を参照します。
   b. **[OK]**をクリックして、アセンブリ参照をアプリケーションに追加します。
2. XAML ビューで、`<Grid> </Grid>` タグの間にカーソルを置きます。
3. ページに2つの標準 Button コントロールと1つの標準 **TextBox** コントロールを追加します。
   a. 最初のボタンのマークアップを次のように編集します。

   XAML マークアップ

   ```
   <Button x:Name="HelloButton" Content="Click Hello" />
   ```

   b. 2つ目のボタンのマークアップを次のように編集します。

   XAML マークアップ

   ```
   <Button x:Name="SaveButton" Content="保存..." />
   ```

   c. TextBox コントロールのマークアップを次のように編集します。

   XAML マークアップ

```
<TextBox
   Name="_tbContent"
   Text="空白"
   IsReadOnly="True"
   AcceptsReturn="True"
   FontFamily="Courier New"
   Background="White" Margin="465,10,242,722" />
```

4. HelloButton用の**HelloButton_Click** というイベントを作成して、MainPage.xaml のコードビューに切り替えます。これにより、コードに **HelloButton_Click** イベントも追加されます。
5. デザインビューに切り替え、SaveButton をダブルクリックして、**SaveButton_Click** イベントをコードに追加します。これで、コードビューが開きます。

コードビュー

計算モードを設定するには、次の手順に従います。

1. ページの先頭に using 文を追加します。

C#
```
using C1.Xaml.Excel;
```

2. 次のようなコードを **MainPage** クラスに追加します。

C#
```
public sealed partial class MainPage : Page
    {
        C1XLBook _book;
    }
```

3. **InitializeComponent()** メソッドに次のコードを追加して、**C1XLBook** を作成します。

C#
```
_book = new C1XLBook();
```

4. RefreshView() メソッドを追加します。後でこのメソッドをコード内で呼び出します。

C#
```
void RefreshView()
        {
        }
```

5. 計算を実行するために簡単な数式を追加します。

C# コードの書き方

C#
```
private void HelloButton_Click(object sender, RoutedEventArgs e)
        {
            // 手順1:新しいワークブックを作成します
            _book = new C1XLBook();
            // 手順2:デフォルトのシートを取得して、名前を付けます
            XLSheet sheet = _book.Sheets[0];
            // 手順3:簡単な数式を追加します
            sheet[7, 0].Value = "Formula: 5!";
            sheet[7, 1].Value = 122;
            sheet[7, 1].Formula = "1*2*3*4*5";
            _book.CalculationMode = CalculationMode.Auto;

            // 手順4:ユーザーがファイルを保存できるようにします
            _tbContent.Text = "'Hello World' ブックが作成され、保存可能になりました。";
            RefreshView();
        }
```

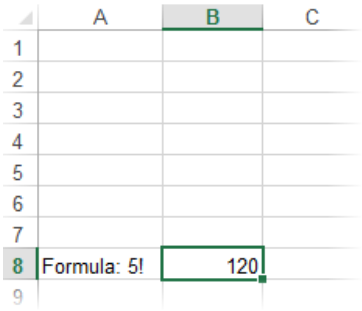
6. ワークブックを保存します。

C# コードの書き方

C#
```
async void SaveButton_Click(object sender, RoutedEventArgs e)
        {
            Debug.Assert(_book != null);
            var picker = new Windows.Storage.Pickers.FileSavePicker();
            picker.SuggestedStartLocation = Windows.Storage.Pickers.
                        PickerLocationId.DocumentsLibrary;
```

```
            picker.FileTypeChoices.Add("Open XML Excel ファイル", new List<string>() { ".xlsx" });
            picker.FileTypeChoices.Add("BIFF Excel ファイル", new List<string>() { ".xls" });
            picker.SuggestedFileName ="新しいブック";
            var file = await picker.PickSaveFileAsync();
            if (file != null)
            {
                try
                {
                    // 手順1:ファイルを保存します
                    var fileFormat = Path.GetExtension(file.Path).Equals(".xls") ?
                                        FileFormat.Biff8 : FileFormat.OpenXml;
                    await _book.SaveAsync(file, fileFormat);
                    // 手順2:ユーザーフィードバック
                    _tbContent.Text = string.Format("ファイルを保存しました : {0}.", file.Path);
                    RefreshView();
                }
                catch (Exception x)
                {
                    _tbContent.Text = string.Format("例外 : {0}", x.Message);
                }
            }
        }
```

プロジェクトを実行し、Excel ファイルを保存してから開きます。**CalculationMode** を **Auto** に設定したため、セル(7,1)の値が **122** ではなく **120**(**1*2*3*4*5** の結果)になっていることに注目してください。



小計の作成

次のコードは、ブックのセルを書式設定する例を示します。

XAML ビュー

次の手順を実行して、このトピック用の XAML ビューを作成します。

1. ソリューションエクスプローラで**[参照]**を右クリックし、リストから**[参照の追加]**を選択します。
   a. C1.UWP.Excel.dll を参照します。
   b. **[OK]**をクリックして、アセンブリ参照をアプリケーションに追加します。
2. XAML ビューで、`<Grid> </Grid>` タグの間にカーソルを置きます。
3. ページに2つの標準 Button コントロールと1つの標準 **TextBox** コントロールを追加します。
   a. 最初のボタンのマークアップを次のように編集します。

XAML マークアップ
`<Button x:Name="HelloButton" Content="Click Hello" />`

   b. 2つ目のボタンのマークアップを次のように編集します。

XAML マークアップ
`<Button x:Name="SaveButton" Content="保存..." />`

   c. TextBox コントロールのマークアップを次のように編集します。

XAML マークアップ

```
<TextBox
   Name="_tbContent"
   Text="空白"
   IsReadOnly="True"
   AcceptsReturn="True"
   FontFamily="Courier New"
   Background="White" Margin="465,10,242,722" />
```

4. HelloButton用の**HelloButton_Click** というイベントを作成して、MainPage.xaml のコードビューに切り替えます。これにより、コードに **HelloButton_Click** イベントも追加されます。
5. デザインビューに切り替え、SaveButton をダブルクリックして、**SaveButton_Click** イベントをコードに追加します。これで、コードビューが開きます。

コードビュー

次の手順を実行して、このトピック用のコードビューを作成します。

1. ページの先頭に using 文を追加します。

C#
```
using C1.Xaml.Excel;
```

2. 次のようなコードを **MainPage** クラスに追加します。

C#
```
public sealed partial class MainPage : Page
    {
        C1XLBook _book;
    }
```

3. **InitializeComponent()** メソッドに次のコードを追加して、**C1XLBook** を作成します。

C#
```
_book = new C1XLBook();
```

4. **RefreshView()** メソッドを追加します。後でこのメソッドをコード内で呼び出します。

C#
```
void RefreshView()
        {
        }
```

5. セルを書式設定するためのコードを追加します。

C# コードの書き方

C#
```
private void HelloButton_Click(object sender, RoutedEventArgs e)
        {
        XLSheet sheet = _book.Sheets[0];
                // スタイルを作成します
                 XLStyle totalStyle = new XLStyle(_book);
                  totalStyle.Font = new XLFont("Arial", 12, true, false);
                // アウトラインを作成し、スタイルを適用します
                 sheet[2, 1].Value = "Number";
                 sheet[2, 2].Value = "ID";
                 sheet[3, 1].Value = 12;
                 sheet[3, 2].Value = 17;
```

```
                sheet.Rows[3].OutlineLevel = 2;
                sheet.Rows[3].Visible = false;
                sheet[4, 1].Value = 12;
                sheet[4, 2].Value = 14;
                sheet.Rows[4].OutlineLevel = 2;
                sheet.Rows[4].Visible = false;
                sheet[5, 1].Value = "12 Total";
                sheet[5, 1].Style = totalStyle;
                sheet[5, 2].Value = 31;
                sheet[5, 2].Formula = "SUBTOTAL(9,C4:C5)";
                sheet.Rows[5].OutlineLevel = 1;
                sheet[6, 1].Value = 34;
                sheet[6, 2].Value = 109;
                sheet.Rows[6].OutlineLevel = 2;
                sheet[7, 1].Value = "34 Total";
                sheet[7, 1].Style = totalStyle;
                sheet[7, 2].Value = 109;
                sheet[7, 2].Formula = "SUBTOTAL(9,C7:C7)";
                sheet.Rows[7].OutlineLevel = 1;
                sheet[8, 1].Value = "Grand Total";
                sheet[8, 1].Style = totalStyle;
                sheet[8, 2].Value = 140;
                sheet[8, 2].Formula = "SUBTOTAL(9,C4:C7)";
                sheet.Rows[8].OutlineLevel = 0;
            }
```

6. ワークブックを保存します。

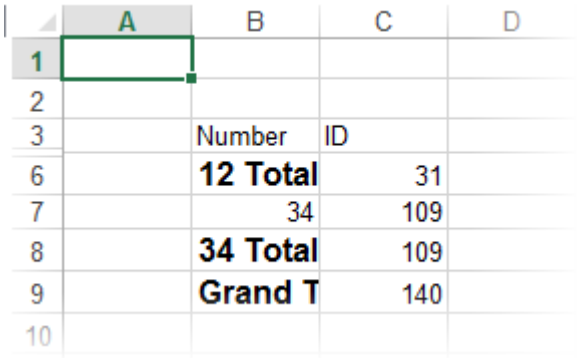
C# コードの書き方

C#

```
async void SaveButton_Click(object sender, RoutedEventArgs e)
         {
             Debug.Assert(_book != null);
             var picker = new Windows.Storage.Pickers.FileSavePicker();
             picker.SuggestedStartLocation =
Windows.Storage.Pickers.PickerLocationId.DocumentsLibrary;
             picker.FileTypeChoices.Add("Open XML Excel ファイル", new
List<string>() { ".xlsx" });
             picker.FileTypeChoices.Add("BIFF Excel ファイル", new List<string>()
{ ".xls" });
             picker.SuggestedFileName ="新しいブック";
             var file = await picker.PickSaveFileAsync();
             if (file != null)
             {
                 try
                 {
                     // 手順1:ファイルを保存します
                     var fileFormat =
Path.GetExtension(file.Path).Equals(".xls") ? FileFormat.Biff8 :
FileFormat.OpenXml;
                     await _book.SaveAsync(file, fileFormat);
                     // 手順2:ユーザーフィードバック
```

```
                    _tbContent.Text = string.Format("ファイルを保存しました: {0}.",
file.Path);
                    RefreshView();
                }
                catch (Exception x)
                {
                    _tbContent.Text = string.Format("例外 : {0}", x.Message);
                }
            }
        }
```

7. プログラムを実行します。ファイルを保存してから開きます。スプレッドシートは次のようになります。



	A	B	C	D
1				
2				
3		Number	ID	
6		**12 Total**	31	
7		34	109	
8		**34 Total**	109	
9		**Grand T**	140	
10				

SUBTOTAL 関数は、指定された行の合計を取得します。