Excel for .Net

2018.02.27 更新

グレープシティ株式会社

Excel for .NET

目次

目次 1
はじめに 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:コンテンツの書式設定 5-6
手順 4:XLS の保存と開始 6-7
Excel for .NET の使い方 8
文書の作成 8-10
ワークシート 10
行と列 10-11
セル 11
スタイル 11
よくある質問と回答 12
Excel for .NETの制限事項 13
タスク別ヘルプ 14
ワークブックに値を追加する 14-15
セルを結合する 15-16
セルの書式を設定する 16-18
既存ワークブックの行を新規ワークブックへコピーする 18-19
セルに画像を挿入する 19-24
セルにコメント追加する 24-25
ワークシートにページ区切りを追加する 25-26
ワークブックの計算モードを設定する 26-27
OpenXml ファイルのインポートとエクスポート 27-29
小計を作成する 29-31
CSV ファイルへの保存と読み込みを行う 31-33
マクロ有効 Excel ファイルのロードおよび保存 33-35

はじめに

Excel for .NET により、Excel® データは任意の .NET アプリケーションから送出された1つのコマンドになり、Microsoft® Excel のインストールさえも不要となります。Excel 97 以降の XLS ファイルを作成またはロードします。**Excel for .NET** は新しい Office 2007 OpenXml 形式をサポートしているため、より小さく圧縮した XLSX ファイルを保存できます。

Excel for .NET のメインコンポーネントは **C1XLBook** オブジェクトで, 枚以上のシートを含む Excel ワークブックを表します。C1XLBook を使用し既存の Excel ファイルをロードするか、新規作成します。次に、シート、スタイル、ハイパーリンク、画像、ヘッダー／フッター、改ページその他多くを追加します。 それが終わったら、, C1XLBook をファイルまたはストリームに保存すれば操作は完了です。Excel のコピーがあれば、誰でもデータにアクセスできます。こんなに簡単です。

主な特長

役立ちそうな **Excel for .NET** の主な特長として、次の事項が挙げられます。

- 1つのコマンドによるワークブックの保存／ロード

  Excel for .NET は使いやすく、グリッドコントロールの場合と同様に、1つのコマンドを使ってワークブックのロード／保存したりシートを操作したりできます。

- 個別セルのデータの読み取りと書き込み

  C1XLBook をロードまたは作成した後、簡単なグリッドの場合と同様に個別シート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。

```
XLSheet sheet = C1XLBook.Sheets[0];
sheet[0, 0].Value = DateTime.Now;
```

- 各セルのデータを書式設定します。

  各セルに関連付けられた形式はセル内に格納されたデータと同様にアクセスが簡単です。たとえば、次のようになります。:

```
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;
```

- **Excel for .NET** を使った XLS ファイルへのエクスポート

  他の ComponentOne コンポーネントでは、**Excel for .NET** を使用して XLS ファイルをエクスポートします。たとえば、**C1Report** は **Excel for .NET** を 使用してレポートの XLS バージョンを作成するため、Microsoft Excel のコピーがあれば誰でも閲覧と編集を行えます。

- Microsoft Excel を使わない .xls と .xlsx ファイルの読み取り／書き込み

  Excel for .NET は .xls（Excel 97 以降）と .xlsx（OpenXml 形式）ファイルの読み取り／書き込みを行います。後者の xlsx は再使用されて、より小さいファイルサイズに簡単に交換または圧縮できます。Microsoft Excel がインストールされている必要もありません。

- **マクロ有効 Excel ファイルのロードおよび保存**

  Excel for .NET は、OpenXMLMacro ファイル形式のロードおよび保存をサポートします。Visual Basic Application（VBA）言語で記述されたマクロ有効 Excel ファイル（XLSM）を **C1Excel** にロードおよび保存できるようになりました。

- 画像の作成とセル内への配置

  画像をセルに追加するだけでなく、セルサイズ、セル内の画像の位置、および画像を拡大縮小するか、クリッピングするか、セルに合わせて引き伸ばすかどうかを指定できます。

- ストリームでのファイルの保存とロード

  Load と **Save** メソッドの新しいオーバーロードにより、ワークブックはメモリストリームの直接読み取り／書き込みが行えるようになったため、一時ファイルの使用が不要です。

- シートのヘッダーとフッターへの画像の追加

  XLPrintSettings クラスのプロパティを使用して、画像をシートのヘッダーまたはフッターの左、中央、または右部分に追加します。

クイックスタート

このクイックスタートを通じて、**Excel for .NET** のいくつかの機能に習熟することができます。このクイックスタートでは、プロジェクトへの C1XLBook の追加、ワークブックへの書式設定済みデータの追加、XLS ファイルの保存および開始を行う方法について学びます。

手順 1:プロジェクトの設定

これで、C1XLBook が追加され、それにコンテンツを追加できるようになりました。

1. 新しい .NET 2.0 プロジェクトを作成します.
2. ツールボックスで、**C1XLBook** アイコンをダブルクリックし、C1XLBook コンポーネントをプロジェクトに追加します。C1XLBook コンポーネントは、フォームの下のコンポーネントトレイに表示されます。
3. フォームをダブルクリックし、**Form1_Load** イベントを追加してコードビューに切り替えます。
4. C1.C1Excel 名前空間内ですべての名前を使用できるように、フォームの最上部に **Imports**(Visual Basic)または **using**(C#)ステートメントを追加します。

Visual Basic コードの書き方

```
Visual Basic
Imports C1.C1Excel
```

C# コードの書き方

```
C#
using C1.C1Excel;
```

これで、C1XLBook が追加され、それにコンテンツを追加できるようになりました。

手順 2:C1XLBook へのコンテンツの追加

Visual Studio プロジェクトのコードビューにいる間に、手順 1:プロジェクトの設定 で 成した **Form_Load** イベントに以下のコードを追加します。このコードは、コンテンツを Excel ワークブックに追加します。

Visual Basic コードの書き方

```
Visual Basic
' コンテンツをシートに追加します。
Dim i As Integer
Dim sheet as XLSheet = C1XLBook1.Sheets(0)
For i = 0 To 9
    sheet(i, 0).Value = (i + 1) * 10
    sheet(i, 1).Value = (i + 1) * 100
    sheet(i, 2).Value = (i + 1) * 1000
Next i
```

C# コードの書き方

```
C#
// コンテンツをシートに追加します。
```

```
int i;
C1.C1Excel.XLSheet sheet = c1XLBook1.Sheets[0];
for (i = 0; i <= 9; i++)
{
   sheet[i, 0].Value = (i + 1) * 10;
   sheet[i, 1].Value = (i + 1) * 100;
   sheet[i, 2].Value = (i + 1) * 1000;
}
```

XLS ファイルの最初の3列内の最初の 10 行は、プロジェクトの実行時に数値が入力されます。

手順 3:コンテンツの書式設定

次に、スタイルを使用してコンテンツを書式設定します。この手順のコードは、手順 2:C1XLBook へのコンテンツの追加 で **Form_Load** イベント内に作成したコードの後に追加する必要があります。

1. 以下のコードを追加して、style1 と style2 という2つの新しいスタイルを作成します。

   Visual Basic コードの書き方

   Visual Basic

   ```
   ' style1 を追加します。
   Dim style1 As New XLStyle(C1XLBook1)
   style1.Font = New Font("Tahoma", 9, FontStyle.Bold)
   style1.ForeColor = Color.RoyalBlue
   ' style2 を追加します。
   Dim style2 As New XLStyle(C1XLBook1)
   style2.Font = New Font("Tahoma", 9, FontStyle.Italic)
   style2.BackColor = Color.RoyalBlue
   style2.ForeColor = Color.White
   ```

   C# コードの書き方

   Title Text

   ```
   // style1 を追加します。
   XLStyle style1 = new XLStyle(c1XLBook1);
   style1.Font = new Font("Tahoma", 9, FontStyle.Bold);
   style1.ForeColor = Color.RoyalBlue;
   // style2 を追加します。
   XLStyle style2 = new XLStyle(c1XLBook1);
   style2.Font = new Font("Tahoma", 9, FontStyle.Italic);
   style2.BackColor = Color.RoyalBlue;
   style2.ForeColor = Color.White;
   ```

2. 次に、以下のコードを追加し、コンテンツに新しいスタイルを適用します。

   Visual Basic コードの書き方

   Visual Basic

   ```
   For i = 0 To 9
       ' コンテンツにスタイルを適用します。
       If (i + 1) Mod 2 = 0 Then
   ```

```
        sheet(i, 0).Style = style2
        sheet(i, 1).Style = style1
        sheet(i, 2).Style = style2
    Else
        sheet(i, 0).Style = style1
        sheet(i, 1).Style = style2
        sheet(i, 2).Style = style1
    End If
Next i
```

C# コードの書き方

C#

```
for (i = 0; i <= 9; i++)
{
    // コンテンツにスタイルを適用します。
   if ((i + 1) % 2 == 0)
   {
           sheet[i, 0].Style = style2;
           sheet[i, 1].Style = style1;
           sheet[i, 2].Style = style2;
   }
   else
   {
           sheet[i, 0].Style = style1;
           sheet[i, 1].Style = style2;
           sheet[i, 2].Style = style1;
   }
}
```

手順 4：XLS の保存と開始

i最後に、以下のコードを追加して、Excel ワークブックを保存したり、ロードしたりします。このコードは、手順 3：コンテンツの書式設定 で **Form_Load** イベント内に作成したコードの後に追加する必要があります。

Visual Basic コードの書き方

Visual Basic

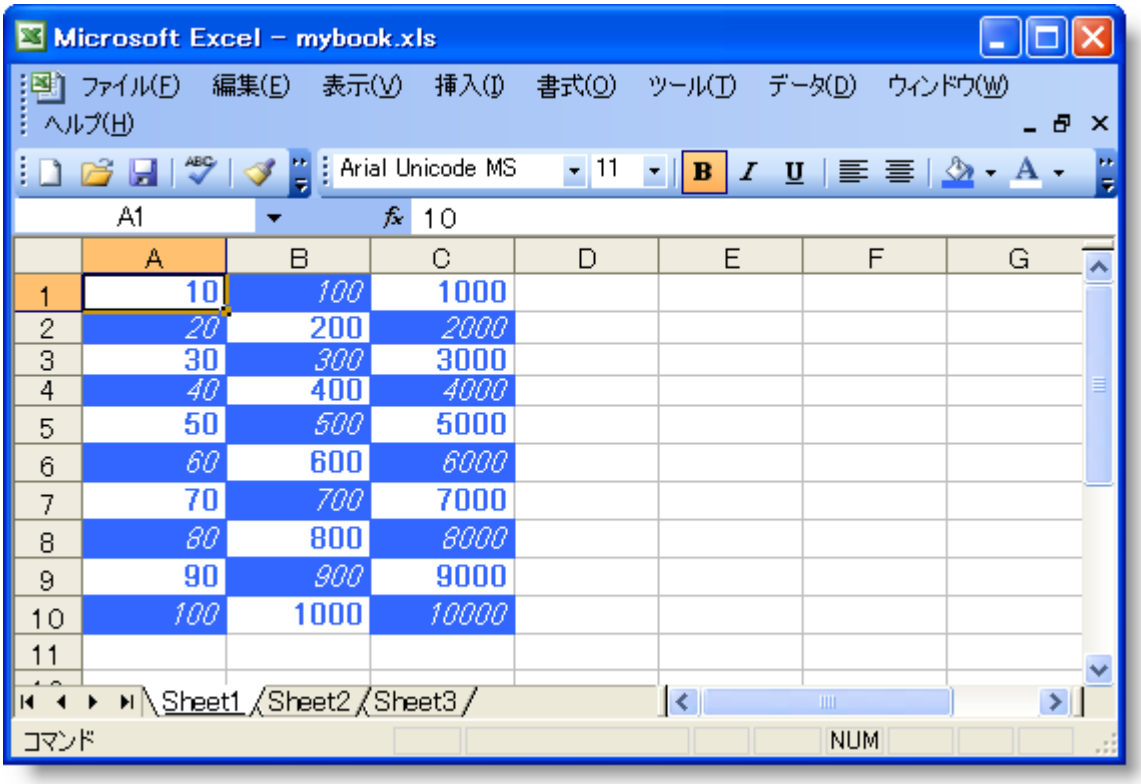
```
C1XLBook1.Save("c:\mybook.xls")
System.Diagnostics.Process.Start("C:\mybook.xls")
```

C# コードの書き方

C#

```
c1XLBook1.Save(@"c:\mybook.xls");
System.Diagnostics.Process.Start(@"c:\mybook.xls");
```

プログラムを実行して、以下を確認します。



書式設定済みコンテンツがワークブックに追加されます。

おめでとうございます!**Excel for .NET** のクイックスタートが無事完了しました。

Excel for .NET の使い方

このセクションでは、新しい XLS ファイルおよび、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、スタイルなど）の主要 **C1Excel** クラスの作成方法について説明します。

文書の作成

C1Excel で XLS ファイルを新規作成する方法は、次のとおりです。

1. **C1XLBook** オブジェクトを新規作成します。各ブックは複数のシート(**XLSheet** オブジェクト)で構成されています。
2. シートに内容を追加します。各シート内にはセル(**XLCell** オブジェクト）があります。また、セルには **Value** プロパティと **Style** プロパティがあります。
3. **Save** メソッドで、ファイルにブックを保存します。

たとえば、次のサンプルコードでは、1枚のシートに1から100までの数字を書き込んで、新しい Excel ファイルを作成します。

Visual Basic コードの書き方

Visual Basic

```
' 手順1:新規ワークブックを作成します。
Dim C1XLBook1 As New C1XLBook()
' 手順2:セルに値を挿入します。
Dim sheet As XLSheet = C1XLBook1.Sheets(0)
Dim i As Integer
For i = 0 To 99
        sheet(i, 0).Value = i + 1
Next i
' 手順3:ファイルを保存します。
C1XLBook1.Save("c:\temp\hello.xls")
```

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。
C1XLBook c1XLBook1 = new C1XLBook();
// 手順2:セルに値を挿入します。
XLSheet sheet = c1XLBook1.Sheets[0];
for (int i = 0; i < 100; i++)
       {
        sheet[i, 0].Value = i + 1;
        }
 // 手順3:ファイルを保存します。
c1XLBook1.Save(@"c:\temp\hello.xls");
```

最も重要な箇所は、手順2の部分です。コードの冒頭部では、新しい Excel ブックから、1枚のワークシートに対応する XLSheet オブジェクトを取得しています。このワークシートは、C1XLBook を 新規作成したときに自動的に作成されたものです。次に、コードでシートインデクサを使用することで、シート内のセルを参照して、1から100の値を挿入しています。

必要に応じて、**XLSheet** オブジェクトのインデクサによって自動的にセルが作成されることに注意してください。この機能のおかげで、作成したワークシートへの値の入力が簡単にできます。シートのサイズを確認するには、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。

また、値をセルに割り当てる以外の機能も利用できます。スタイルを設定することで、セル書式を設定することもできます。

XLStyle オブジェクトを作成すれば、値の場合と同じようにセルに書式を設定できるのです。次のコードは上記のコードを修正したものですが、偶数を太字の赤で、奇数を斜体の青で表示するシートを作成しています。

Visual Basic コードの書き方

Visual Basic

```
'手順1:新規ワークブックを作成します。
Dim C1XLBook1 As New C1XLBook()
'手順2:奇数セル及び偶数セルにそれぞれ異なるスタイルを設定します。
Dim styleOdd As New XLStyle(C1XLBook1)
styleOdd.Font = New Font("Tahoma", 9, FontStyle.Italic)
styleOdd.ForeColor = Color.Blue
Dim styleEven As New XLStyle(C1XLBook1)
styleEven.Font = New Font("Tahoma", 9, FontStyle.Bold)
styleEven.ForeColor = Color.Red
'手順3:セルに値及びスタイルを書き込みます。
Dim sheet As XLSheet = C1XLBook1.Sheets(0)
Dim i As Integer
For i = 0 To 99
   Dim cell As XLCell = sheet(i, 0)
   cell.Value = i + 1
   If (i + 1) Mod 2 = 0 Then
      cell.Style = styleEven
   Else
      cell.Style = styleOdd
   EndIf
Next i
 '手順4:ファイルを保存します。
C1XLBook1.Save("c:\temp\hello.xls")
```

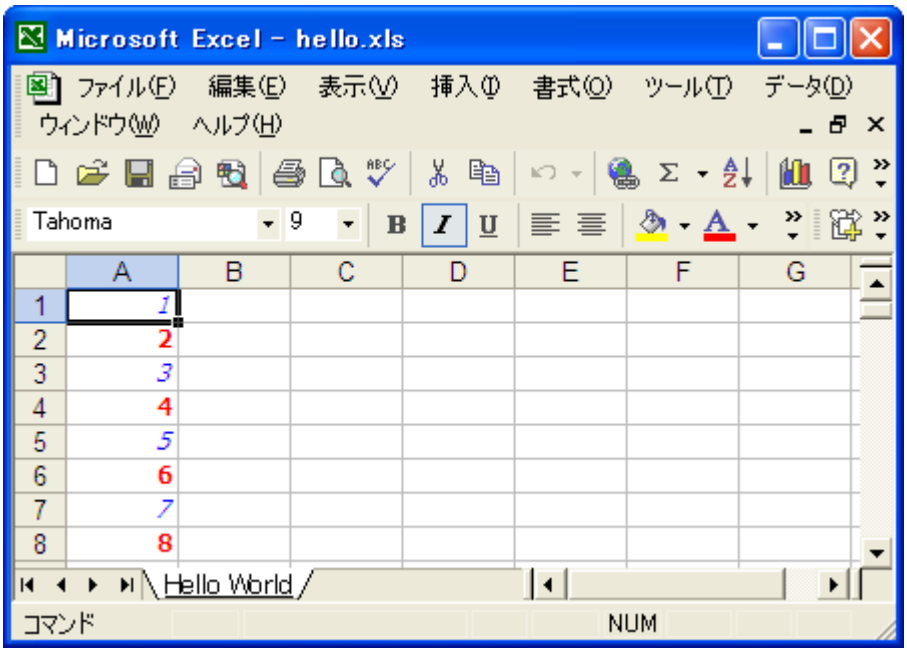
C# コードの書き方

C#

```
//手順1:新規ワークブックを作成します。
C1XLBook c1XLBook1 = new C1XLBook();
//手順2:奇数セル及び偶数セルにそれぞれ異なるスタイルを設定します。
XLStyle styleOdd    = new XLStyle(c1XLBook1);
styleOdd.Font       = new Font("Tahoma", 9, FontStyle.Italic);
styleOdd.ForeColor  = Color.Blue;
XLStyle styleEven   = new XLStyle(c1XLBook1);
styleEven.Font      = new Font("Tahoma", 9, FontStyle.Bold);
styleEven.ForeColor = Color.Red;
//手順3:セルに値及びスタイルを書き込みます。
XLSheet sheet = c1XLBook1.Sheets[0];
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4:ファイルを保存します。
c1XLBook1.Save(@"c:\temp\hello.xls");
```

このコードは、最初のコードとほぼ同じものです。大きな違いは、手順2の部分を記述しなおすことで、奇数セルと偶数セルにそれぞれ別スタイルを設定している点です。セルの値の場合と同様に、手順3でも新たにセルにスタイルを設定しています。

上記のコードで作成したファイルを Microsoft Excel で開くと、次のように表示されます。



ワークシート

ワークシートとは、Excel ファイル内の個々のグリッドのことです。ワークシートは、**C1XLBook** クラスの Sheets プロパティからアクセスできる **XLSheet** オブジェクトによって表されます。各シートには名前が付いています。また、各シートには行と列の各コレクションが存在します。各セルには、行や列のインデックスを取得できる **XLSheet** インデクサを利用してアクセスできるようになっています。

XLSheet オブジェクトの **Rows** コレクションと **Columns** コレクションは、インデクサを使用して自動的に拡張できます。たとえば、次のようなコードを作成するとします。シートの行数が1000以下の場合、新しい行が自動的に追加され、有効な行が返されます 。同じことが **XLColumn** インデクサや **XLCell** インデクサに当てはまります。これは .NET で通常採用されているコレクションインデクサの動作とは若干異なりますが、この仕様によって **XLSheet** オブジェクトの作成や追加が簡単になっています。

Visual Basic コードの書き方

```
Visual Basic
Dim sheet As XLSheet = C1XLBook1.Sheets(0)
Dim row As XLRow = sheet.Rows(1000)
```

C# コードの書き方

```
C#
XLSheet sheet = c1XLBook1.Sheets[0];
XLRow row = sheet.Rows[1000];
```

行と列

XLSheet オブジェクトには行と列のコレクションがあります。これら2つのコレクションは、シートのそれぞれの行と列を外部に公開しています。公開された **XLRow** オブジェクトと **XLColumn** オブジェクトを利用すれば、シートの各行や列に対して、サイズ（列の幅、行の高さ）、表示、およびスタイルを設定することができます。この値を割り当てない限り、シートにはデフォルトの値が適用されます。この機能の詳細については、**DefaultRowHeight** プロパティと **DefaultColumnWidth** プロパティを参照してください。

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 になっています。これは、シートにデフォルトの値を適用することを表します。

セル

XLSheet オブジェクトには、インデクサでアクセスできるセルがあります。インデクサとは、行や列のインデックスを取得するための機能のことです。このセルは、セルに値やスタイルが設定された**XLCell** オブジェクトとして表されます。

行や列の場合のように、セルのインデクサでもシートを自動的に拡張できます。この動作については、次のコード例が参考になります。

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];
```

シート内の行と列がそれぞれ10以下の場合、行と列が追加されて、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ので、インデクサが **null** 値の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しているかどうかを確認する場合や、セルを不必要に作成したくない場合には、インデクサの代わりにシートの **GetCell** メソッドを使用します。

XLCell オブジェクトにはセルの内容を表す **Value** プロパティがあります。このプロパティは **objec**t 型ですが、文字列や数値型、ブール型、DateTime 型、NULL 値のオブジェクトなどを含みます。これ以外の型のオブジェクトは、Excel ファイルに保存できないことに注意してください。

XLCell オブジェクトには、セルの外観を定義する **Style** プロパティもあります。**Style** プロパティが NULL 値に設定されると、セルはデフォルトのスタイルで表示されます。**NULL** 値以外の場合、**XLStyle** オブジェクトを設定して、フォント、配置、色、書式設定などのセルの外観を定義しておく必要があります。

スタイル

XLStyle クラスは、シート内のセルや行、列の外観を設定するためのクラスです。**XLStyle** には、セル内の値の表示用のフォントやレイアウト、色、書式設定などのスタイル要素を指定するためのプロパティが定義されています。各 **XLStyle** オブジェクトにスタイルの全要素を設定しておく必要はありません。たとえば、**XLStyle** で書式設定だけを指定しておけば、セルは指定された書式設定で表示されます。このとき、フォントやレイアウトなどの一部のスタイル要素を指定しておかなくても、デフォルトの設定がそのまま表示されます。

よくある質問と回答

C1Excel についてよくある質問は以下の通りです:

1）**セルに設定された数式の読み込みや保存はできますか？**

C1Excel ではファイルから読み込んだ数式を保持し、ファイルへの保存時にそれらを書き戻すことができます。C1Excelで、数式を作成したり変更したりすることも可能です。

2）**「名前が付けられた参照の範囲」を読み込んだり保存できますか？**

C1Excelでは「名前が付けられた参照の範囲」の読み書きは可能です。この範囲を保持しながら既存ファイルをロードし、セルを修正し、ファイルをディスクに保存する機能を利用することができます。ただし、「名前が付けられた参照の範囲」の作成、および追加機能は、現時点ではサポートされていません。

3）**XLOpaqueShape オブジェクトと XLPictureShape オブジェクトの違いは？**

C1ExcelBook は Excel シートの様々な図形（画像、グラフィックス、コントロール及びコメントなど）をロード・保存するために、**XLPictureShape** 及び **XLOpaqueShape** オブジェクトを提供します。

画像（画像ファイル）は **XLPictureShape** オブジェクトとしてロードされ、編集（明るさやコントラストの変更、切り取り、回転など）を行なうことができます。また、セルの Value プロパティを用いて、セルに画像を追加することも可能です。詳細については、「セルに画像を挿入する」を参照してください。追加・編集された画像は、Excel の保存時に指定したセル上に保存されます。

画像以外のオブジェクト（オーシェープ、クリップアート、テキストボックス、コメントなど）は **XLOpaqueShape** オブジェクトとしてロードされます。そのプロパティは参照のみ可能で、**XLOpaqueShape** オブジェクトの編集を行なったり PictureBox に表示したりすることはできません。また任意のセルに、これらの画像以外のオブジェクトを追加することはできません。

XLOpaqueShape クラス、XLPictureShape クラスは**XLShape** クラスを継承します。

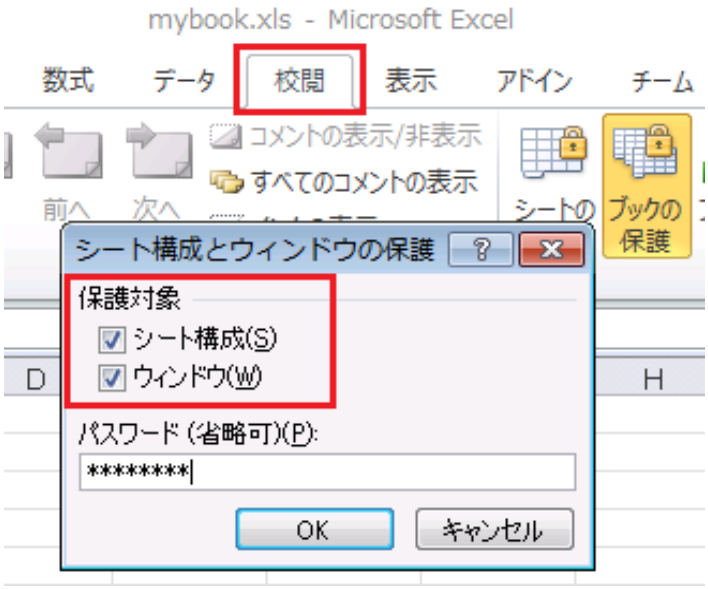
Excel for .NETの制限事項

Excel for .NET には、いくつかの既知の制限事項があります。

- PivotTable 宣言は定義されていません。
- VBA マクロはサポートされておりますが限定的です。**Excel for .NET** では Open XML フォーマットの不透明コピーのみがサポートされています。
- **Fill** プロパティを使用して書式化されたセルを .xlsx 形式から .xls 形式へ保存し直した際、.xlsx 形式の行および列が .xls 形式の最大値を超えると ArgumentException 例外がスローされます。
- Excel ファイルを読み込み保存しなおす際、ファイル形式（BIFF8、OpenXml）に関係なく、チャートオブジェクトは保存されません。
- 「シート構成とウィンドウの保護」はサポートされません：

  Excel for .NET を使用して、シート構成／ウィンドウがパスワードで保護された Microsoft Excelファイルを読み込むと、「Central dir not found」例外または FileLoadException 例外が発生することがあります。

  次の画像は、Microsoft Excel 2010 にてワークブックのシート構成／ウィンドウがパスワードで保護されている様子を示しています:



タスク別ヘルプ

タスク別ヘルプは、Visual Studio のプログラミングに習熟し、連結コントロールと非連結コントロールの一般的な使い方を知っていることを前提にしています。ヘルプで概略された各手順に従っていくと、**C1Excel** の多彩な機能をデモ表示するプロジェクトを作成し、**C1Excel** を使って何ができるかを体験できます。

各タスク別ヘルプトピックでは、新しい .NET プロジェクトが既に作成されていることを前提としています。

ワークブックに値を追加する

このセクションでは、新しいワークブックの作成方法および最初の10個のセルに値を追加する方法を説明します。

1. ツールボックスにある **C1XLBook** コンポーネントをダブルクリックして Form1 に追加します。
2. 最初の10個のセルに値を設定します。

Visual Basic コードの書き方

Visual Basic

```
Dim sheet As XLSheet = C1XLBook1.Sheets(0)
Dim i As Integer
  For i = 0 To 9
    sheet(i,0).Value = i + 1
  Next i
```

C# コードの書き方

C#

```
XLSheet sheet = c1XLBook1.Sheets[0];
for (int i = 0; i <= 9; i++)
        {
        sheet[i,0].Value = i + 1;
        }
```

3. ワークブックを保存し、開きます。

Visual Basic コードの書き方

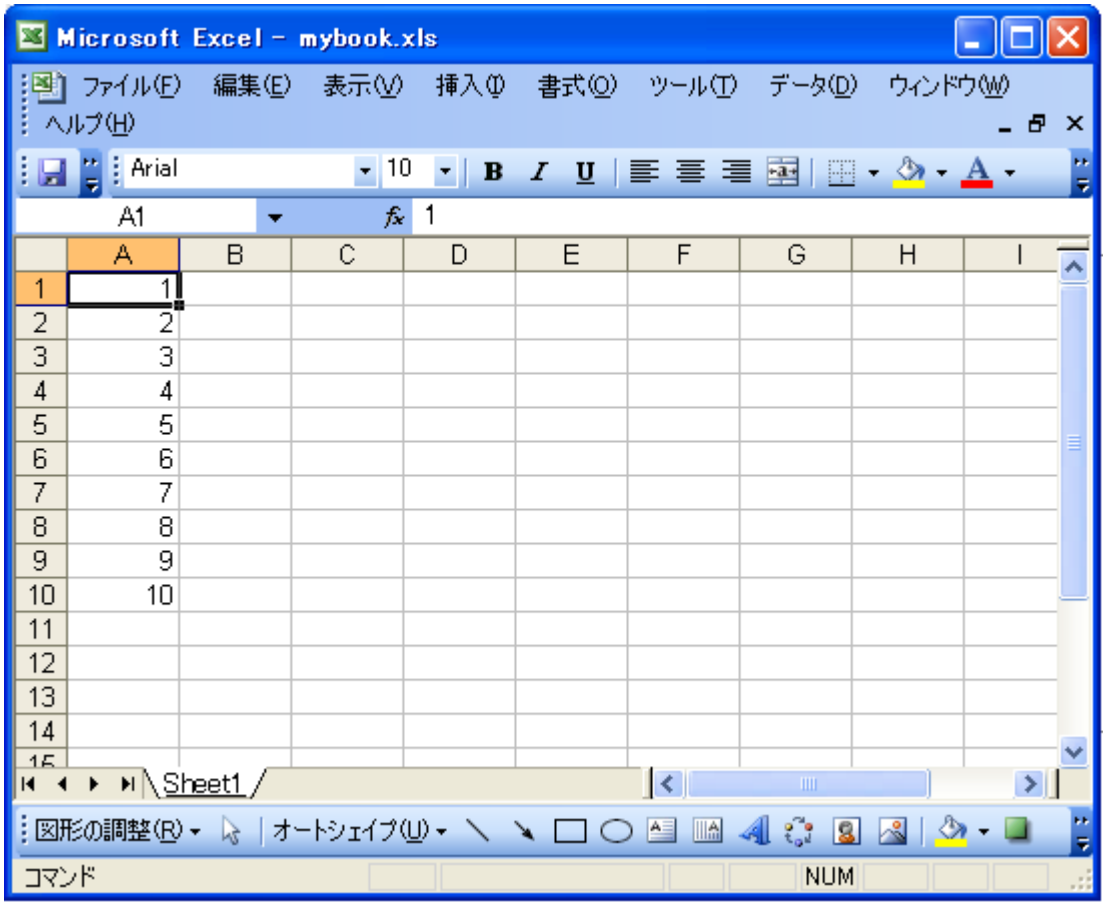
Visual Basic

```
C1XLBook1.Save("c:\mybook.xls")
System.Diagnostics.Process.Start("C:\mybook.xls")
```

C# コードの書き方

C#

```
c1XLBook1.Save(@"c:\mybook.xls");
System.Diagnostics.Process.Start(@"C:\mybook.xls");
```



セルを結合する

C1Excel.MergedCells を使用し、結合するセルの範囲を指定することでセルを結合することが可能です。結合したセルを使用して、シート上の結合した範囲の検査、追加、クリアなどを行います。結合した各範囲が **C1.C1Excel.XLCellRange** obオブジェクトで表されます。列を追加・削除する際、セルの結合状態が保持されます。

セルを結合するには、下記の手順に従います:

1. ツールボックスにある**C1XLBook** コンポーネントをダブルクリックして Form1 に追加します。
2. 下記のコードを使用して2つの **XLCellRanges** (**_ColRange** と **_RowRange**) を作成し、MergedCells コレクションに追加します。このコードで結合したセルにもスタイルが適用されます。たとえば、以下のコードを**Form1_Load** イベントに追加できます。

Visual Basic コードの書き方

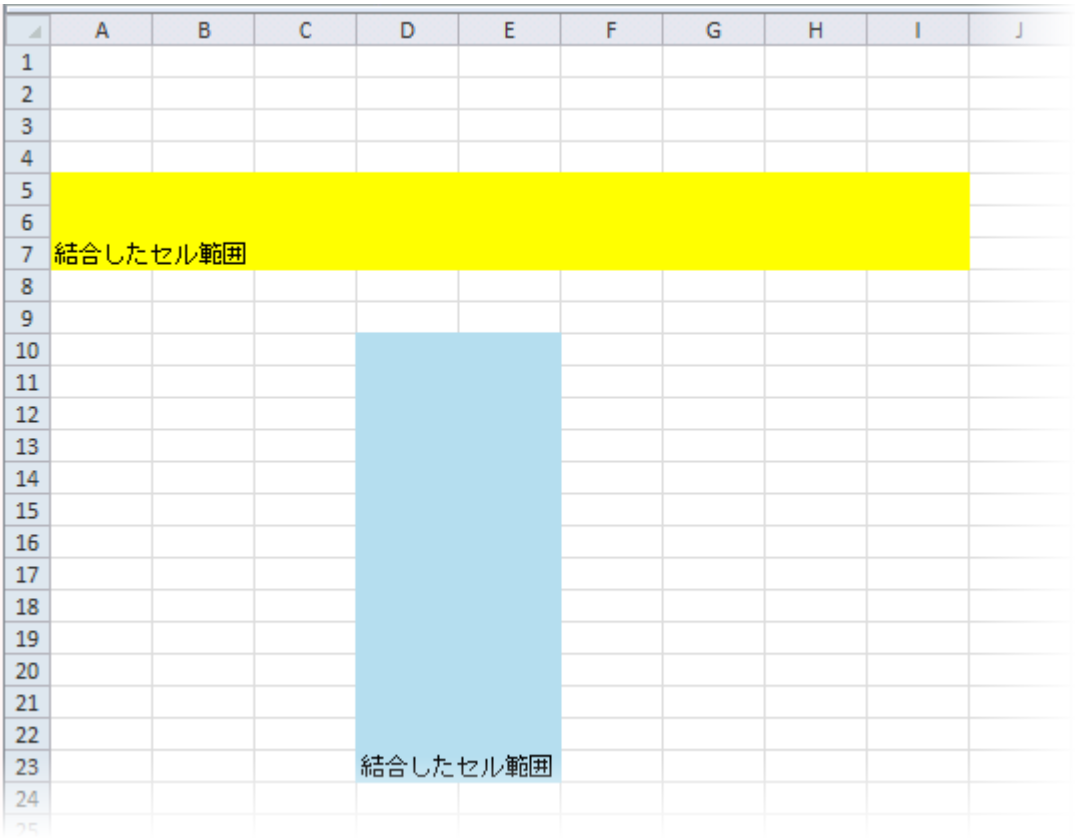
Visual Basic

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        //結合する範囲を選択します。
    Dim _ColRange As C1.C1Excel.XLCellRange = New C1.C1Excel.XLCellRange(4, 6,
0, 8)
        Dim _RowRange As C1.C1Excel.XLCellRange = New C1.C1Excel.XLCellRange(9,
21, 3, 4)
         //結合したセルにテキストを設定します。
       C1XLBook1.Sheets(0)(4, 0).Value = "結合したセル範囲"
        C1XLBook1.Sheets(0)(9, 3).Value = "結合したセル範囲"
        //セルを結合します。
```

```
    C1XLBook1.Sheets(0).MergedCells.Add(_ColRange)
       C1XLBook1.Sheets(0).MergedCells.Add(_RowRange)
       //結合したセルのスタイルを定義して適用します。
     Dim _Colstyle As C1.C1Excel.XLStyle = New C1.C1Excel.XLStyle(C1XLBook1)
       _Colstyle.BackColor = Color.Yellow
       Dim _Rowstyle As C1.C1Excel.XLStyle = New C1.C1Excel.XLStyle(C1XLBook1)
       _Rowstyle.BackColor = Color.LightBlue
       C1XLBook1.Sheets(0)(4, 0).Style = _Colstyle
       C1XLBook1.Sheets(0)(9, 3).Style = _Rowstyle
       //ワークブックを保存し、開きます。
   C1XLBook1.Save("c:\mybook.xls")
       System.Diagnostics.Process.Start("C:\mybook.xls")

   End Sub
```



セルの書式を設定する

このセクションでは、ワークブックのセルの書式を設定する方法を説明します。

1. ツールボックスから **C1XLBook** をフォームに追加するには、コンポーネントをダブルクリックします。
2. **XLStyle** オブジェクトの新しいインスタンスを作成し、オブジェクトのプロパティに値を設定します。

Visual Basic コードの書き方

Visual Basic

```
Dim style1 As New XLStyle(C1XLBook1)
style1.ForeColor = Color.Gold
style1.BackColor = Color.Blue
```

```
style1.Format = "$  .00"
```

C# コードの書き方

```
C#
XLStyle style1 = new XLStyle(c1XLBook1);
style1.ForeColor = Color.Gold;
style1.BackColor = Color.Blue;
style1.Format = "$ .00";
```

3. 列目のセルに値を設定し、新しいスタイルを適用します。

Visual Basic コードの書き方

```
Visual Basic
Dim sheet As XLSheet = C1XLBook1.Sheets(0)
Dim i As Integer
For i = 0 To 9
        sheet(i,0).Value = i + 1
        sheet(i, 0).Style = style1
Next i
```

C# コードの書き方

```
C#
C1.C1Excel.XLSheet sheet = c1XLBook1.Sheets[0];
int i;
for (i = 0; i <= 9; i++)
        {
         sheet[i,0].Value = i + 1;
         sheet[i, 0].Style = style1;
        }
```

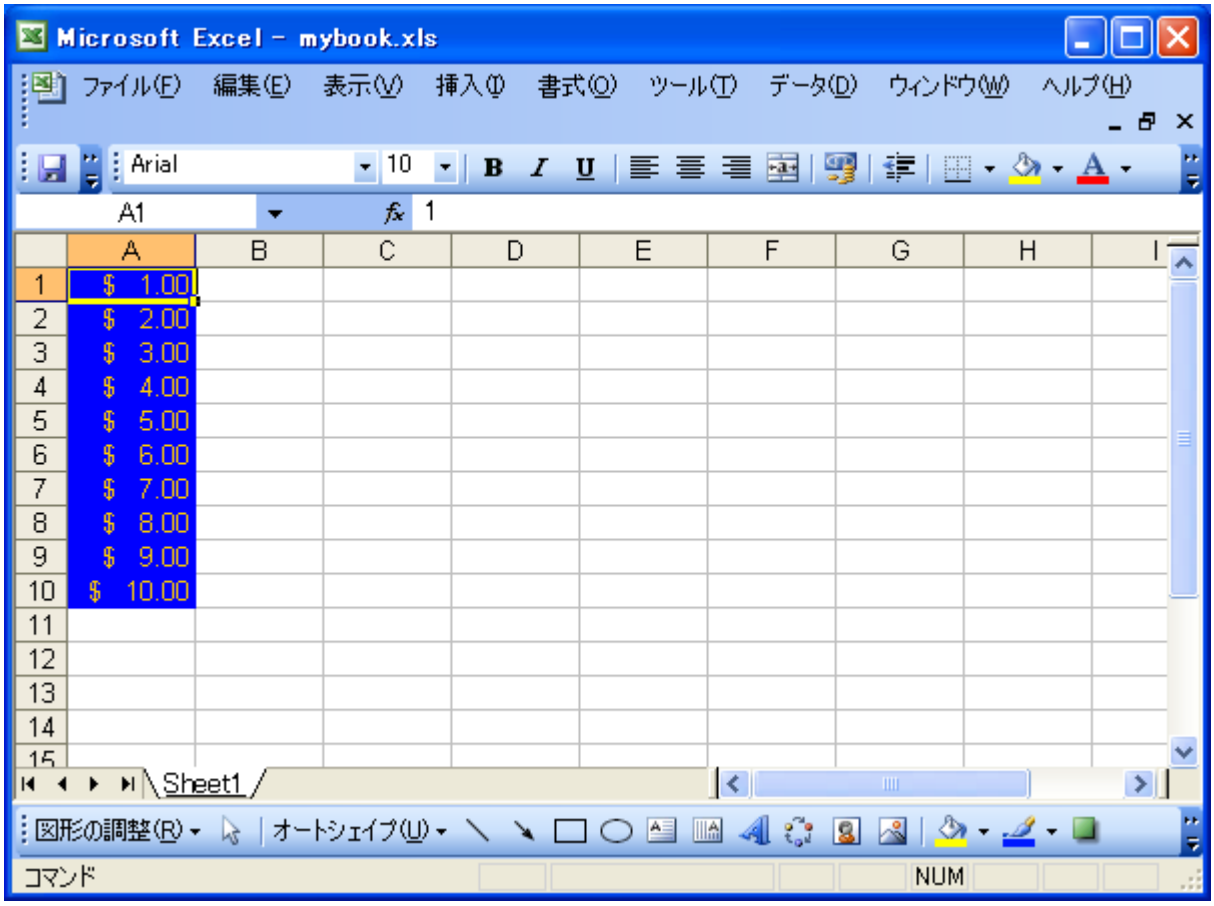
4. ワークブックを保存し、開きます。

Visual Basic コードの書き方

```
Visual Basic
C1XLBook1.Save("c:\mybook.xls")
System.Diagnostics.Process.Start("C:\mybook.xls")
```

C# コードの書き方

```
C#
c1XLBook1.Save(@"c:\mybook.xls");
System.Diagnostics.Process.Start(@"C:\mybook.xls");
```



既存ワークブックの行を新規ワークブックへコピーする

このセクションでは、既存ワークブックの行を新規ワークブックへコピーする方法を説明します。

1. 既存ワークブックをロードします。

Visual Basic コードの書き方

Visual Basic

```
Dim wb As New C1XLBook()
wb.Load("C:\test.xls")
```

C# コードの書き方

C#

```
C1XLBook wb = new C1XLBook();
wb.Load(@"C:\test.xls");
```

2. 新規 XLSheetを作成します。

Visual Basic コードの書き方

Visual Basic

```
Dim xb As New C1XLBook()
xb.Sheets.Add("Test")
```

C# コードの書き方

```
C#
C1XLBook xb = new C1XLBook();
xb.Sheets.Add("Test");
```

3. 新規 XLSheetを作成します既存ワークブックの各行を新規ワークブックのXLSheetへコピーします。

Visual Basic コードの書き方

```
Visual Basic
Dim source As XLSheet = wb.Sheets(0)
Dim dest As XLSheet = xb.Sheets("Test")
Dim row As Integer, col As Integer
    For row = 0 To source.Rows.Count - 1
      For col = 0 To source.Columns.Count - 1
          dest(row, col).Value = source(row, col).Value
      Next col
    Next row
```

C# コードの書き方

```
C#
XLSheet source = wb.Sheets[0];
XLSheet dest = xb.Sheets("Test");
for (int row = 0; row <= source.Rows.Count - 1; row++)
{
   for (int col = 0; col <= source.Columns.Count - 1; col++)
   {
       dest[row, col].Value = source[row, col].Value;
   }
}
```

4. ワークブックを保存し、開きます。

Visual Basic コードの書き方

```
Visual Basic
xb.Save("C:\test2.xls")
System.Diagnostics.Process.Start("C:\test2.xls")
```

C# コードの書き方

```
C#
xb.Save(@"c:\test2.xls");
System.Diagnostics.Process.Start(@"C:\test2.xls");
```

Test という名のシートが2枚目に追加され、既存ワークブックの1枚目のシートの内容がコピーされます。

セルに画像を挿入する

次のいずれかの方法を使用して、画像をシートに挿入できます。それぞれの方法の詳細については、以下のリンクをクリックしてください。

方法1：XLCell.Value プロパティの値へ直接画像を設定します。

この方法を使用すると、画像は指定されたセルの左上端を基点とし、元のサイズで描画されます。画像の左上隅は指定されたセルの左上隅と重なります。

1. 既存ワークブックまたは新しいワークブックをロードします。

   Visual Basic コードの書き方

   Visual Basic
   ```
   Dim wb As New C1XLBook
   wb.Load("C:\Project\WorkBook1.xls")
   ```

   C# コードの書き方

   C#
   ```
   C1XLBook wb = new C1XLBook();
   wb.Load(@"C:\Project\WorkBook1.xls");
   ```

2. 画像を指定し、**Value** プロパティの値へ設定します。

   Visual Basic コードの書き方

   Visual Basic
   ```
   Dim img As Image = Image.FromFile("C:\Project\MyImage.bmp")
   Dim sheet As XLSheet = wb.Sheets("Forecasting Report")
   sheet(0, 0).Value = img
   ```

   C# コードの書き方

   C#
   ```
   Image img = Image.FromFile(@"C:\Project\MyImage.bmp");
   XLSheet sheet = wb.Sheets("Forecasting Report");
   sheet[0,0].Value = img;
   ```

3. ワークブックを保存し、開きます。

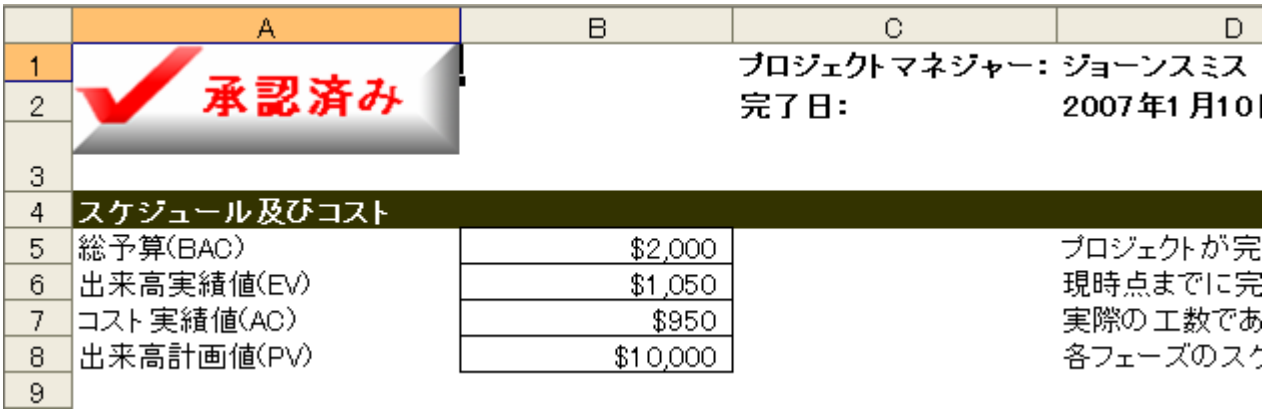
   Visual Basic コードの書き方

   Visual Basic
   ```
   wb.Save("C:\Project\WorkBook1.xls ")
   System.Diagnostics.Process.Start("C:\Project\WorkBook1.xls")
   ```

   C# コードの書き方

   C#
   ```
   wb.Save(@"C:\Project\WorkBook1.xls");
   System.Diagnostics.Process.Start(@"C:\Project\WorkBook1.xls");
   ```

このサンプルでは、画像は第1番目のセルに元のサイズで表示されます。



方法 2:XLPictureShape オブジェクトを作成し、プロパティを設定した後、XLCell.Value プロパティの値へ新しいオブジェクトを設定します。

このサンプルでは、画像は第1番目のセルに元のサイズで表示されます既存ワークブックまたは新しいワークブックをロードします。

1. 既存ワークブックまたは新しいワークブックをロードします。

   Visual Basic コードの書き方

   Visual Basic
   ```
   Dim wb As New C1XLBook
   wb.Load("C:\Project\WorkBook1.xls")
   ```

   C# コードの書き方

   C#
   ```
   C1XLBook wb = new C1XLBook();
   wb.Load(@"C:\Project\WorkBook1.xls");
   ```

2. **XLPictureShape** オブジェクトを作成し、プロパティを設定した後、Value プロパティの値に新しいオブジェクトを設定します。

   Visual Basic コードの書き方

   Visual Basic
   ```
   Dim img As Image = Image.FromFile("C:\Project\MyImage.bmp")
   Dim pic As New XLPictureShape(img, 1500, 1500)
   pic.Rotation = 30.0F
   pic.LineColor = Color.DarkRed
   pic.LineWidth = 100
   ' 指定したシートの第1番目のセルに pic(新XLPictureShapeオブジェクト)を設定します。
   Dim sheet As XLSheet = wb.Sheets("Forecasting Report")
   sheet(0, 0).Value = pic
   ```

   C# コードの書き方

   C#
   ```
   Image img = Image.FromFile(imageFileName);
   XLPictureShape pic = new XLPictureShape(img, 1500, 1500);
   pic.Rotation = 30.0f;
   ```

```
pic.LineColor = Color.DarkRed;
pic.LineWidth = 100;
// 指定したシートの第1番目のセルに pic(新XLPictureShapeオブジェクト)を設定します。
XLSheet sheet = wb.Sheets("Forecasting Report");
sheet[0,0].Value = pic;
```

3. ワークブックを保存し、開きます。

Visual Basic コードの書き方

Visual Basic

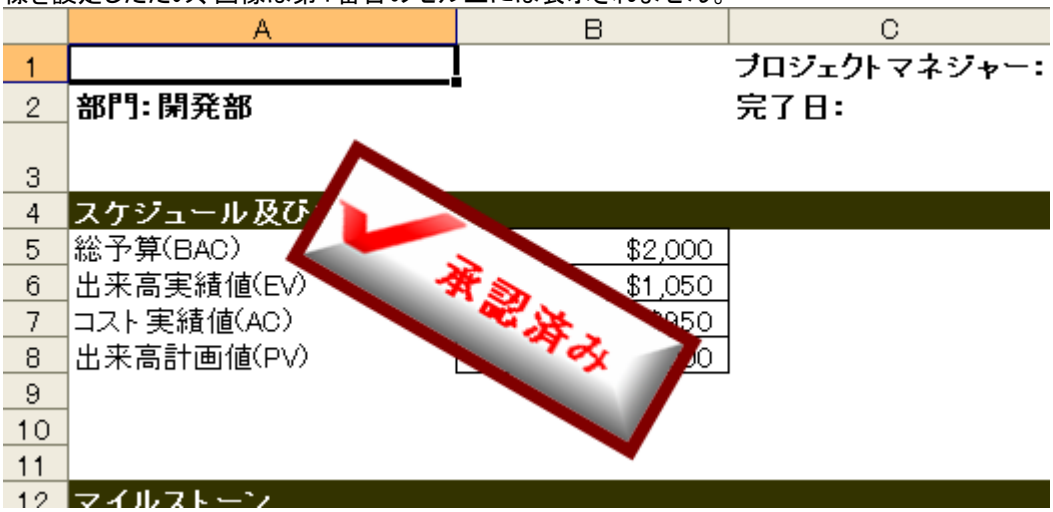
```
wb.Save("C:\Project\WorkBook1.xls ")
System.Diagnostics.Process.Start("C:\Project\WorkBook1.xls")
```

C# コードの書き方

C#

```
wb.Save(@"C:\Project\WorkBook1.xls");
System.Diagnostics.Process.Start(@"C:\Project\WorkBook1.xls");
```

上記コードを実行すると、第1番目のセルの画像は、濃赤色の境界線、30°の回転角付きの画像で上書きされます。ただし座標を設定したため、画像は第1番目のセル上には表示されません。



方法 3 :XLPictureShape オブジェクトを作成し、プロパティを設定した後、新しいオブジェクトをシートのShapeCollectionに追加します。

この方法では、**XLPictureShape** のコンストラクタを使用し、シート内の画像の位置・サイズなを調整します。画像は、シートの **ShapeCollection** に追加されます。

1. 既存ワークブックまたは新規ワークブックをロードします。

Visual Basic コードの書き方

Visual Basic

```
Dim wb As New C1XLBook
wb.Load("C:\Project\WorkBook1.xls")
```

C# コードの書き方

C#

```
C1XLBook wb = new C1XLBook();
wb.Load(@"C:\Project\WorkBook1.xls");
```

2. XLPictureShape オブジェクトを作成し、プロパティを設定した後、新しいオブジェクトをシートの ShapeCollection に追加します。

Visual Basic コードの書き方

Visual Basic

```
Dim img As Image = Image.FromFile("C:\Project\MyImage.bmp")
Dim pic As New XLPictureShape(img, 3000, 3500, 2500, 900)
pic.Rotation = 30.0F
pic.LineColor = Color.DarkRed
pic.LineWidth = 100
' 指定したシートのShapeCollectionに pic(新XLPictureShapeオブジェクト)を追加します。
Dim sheet As XLSheet = wb.Sheets("Forecasting Report")
sheet.Shapes.Add(pic)
```

C# コードの書き方

C#

```
Image img = Image.FromFile(@"C:\Project\MyImage.bmp");
XLPictureShape pic = new XLPictureShape(img, 3000, 3500, 2500, 900);
pic.Rotation = 30.0f;
pic.LineColor = Color.DarkRed;
pic.LineWidth = 100;
// 指定したシートのShapeCollectionに pic(新XLPictureShapeオブジェクト)を追加します。
XLSheet sheet = wb.Sheets("Forecasting Report");
sheet.Shapes.Add(pic)
```

3. ワークブックを保存し、開きます。

Visual Basic コードの書き方

Visual Basic

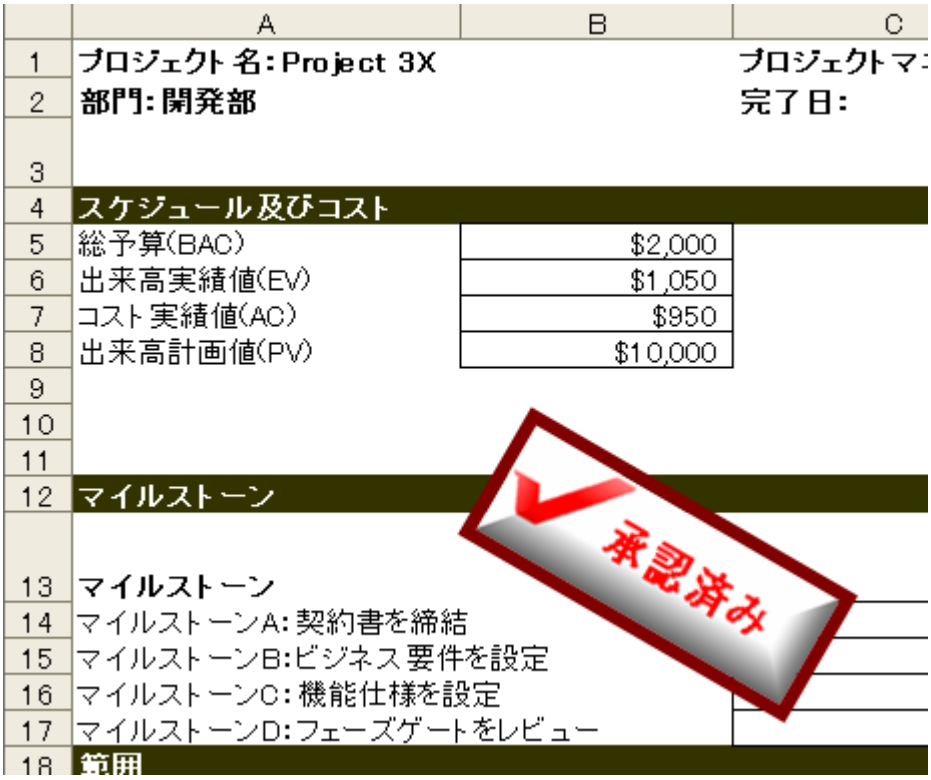
```
wb.Save("C:\Project\WorkBook1.xls ")
System.Diagnostics.Process.Start("C:\Project\WorkBook1.xls")
```

C# コードの書き方

C#

```
wb.Save(@"C:\Project\WorkBook1.xls");
System.Diagnostics.Process.Start(@"C:\Project\WorkBook1.xls");
```

上記コードでは、画像は指定された **ShapeCollection** に追加されたため、第1番目のセルの値を上書きしません。画像の高さ、幅、垂直位置、水平位置も指定可能です。



セルにコメント追加する

セルにコメントを追加するには、以下の手順に従います。

1. ツールボックスにある **C1XLBook** コンポーネントをダブルクリックして Form1 に追加します
2. 下記コードを使用してワークブックのセルにテキストを追加します。

Visual Basic コードの書き方

```
Visual Basic
C1XLBook1.Sheets(0)(2, 3).Value = "test"
```

C# コードの書き方

```
C#
c1XLBook1.Sheets[0][2, 3].Value = "test";
```

3. **XLCommentCollection** にコメントを追加し、下記コードを使用してコメントを表示するボックスを作成します。

Visual Basic コードの書き方

```
Visual Basic
C1XLBook1.Sheets(0).Comments.Add(2, 3, "John", "Test comment")
C1XLBook1.Sheets(0).Comments(0).TextBox.Rectangle = New Rectangle(220, 210, 1900, 1200)
```

C# コードの書き方

```
C#
c1XLBook1.Sheets[0].Comments.Add(2, 3, "John", "Test comment");
```

```
c1XLBook1.Sheets[0].Comments[0].TextBox.Rectangle = new Rectangle(220, 210, 1900, 1200);
```

4. ワークブックを保存し、開きます。

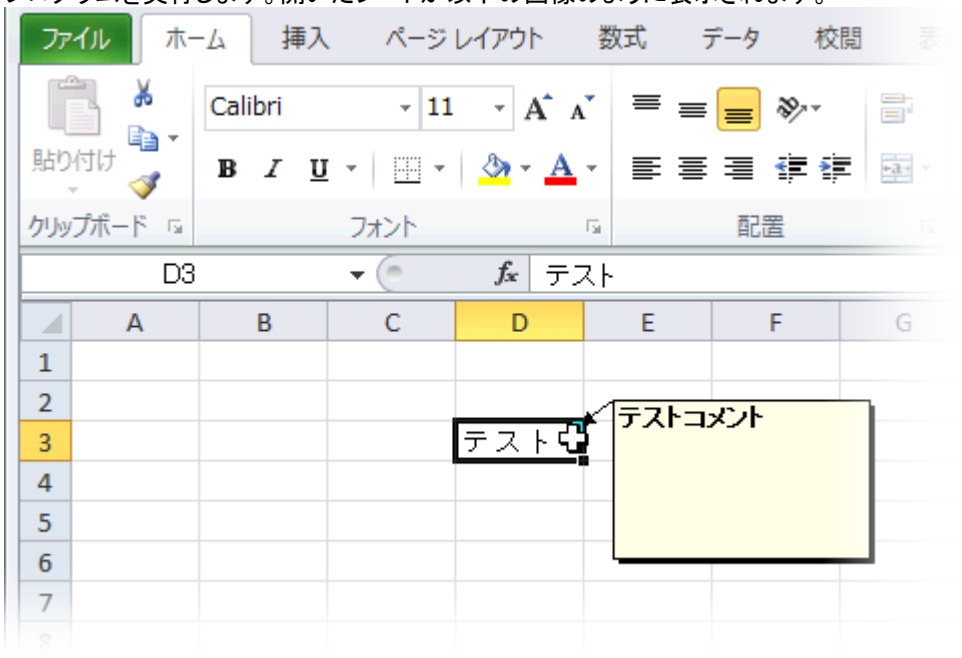
Visual Basic コードの書き方

```
Visual Basic
C1XLBook1.Save("c:\mybook.xls")
System.Diagnostics.Process.Start("C:\mybook.xls")
```

C# コードの書き方

```
C#
c1XLBook1.Save(@"c:\mybook.xls");
System.Diagnostics.Process.Start(@"C:\mybook.xls");
```

5. プログラムを実行します。開いたシートが以下の画像のように表示されます。



ワークシートにページ区切りを追加する

PageBreak と PageBreak プロパティを使用して、OpenXML(.xlsx) 形式にて行と列にページ区切りを追加することができます。

1. ツールボックスにある **C1XLBook** コンポーネントをダブルクリックして Form1 に追加します。
2. 下記コードを使用して、任意のテキストとページ区切りを追加します。

Visual Basic コードの書き方

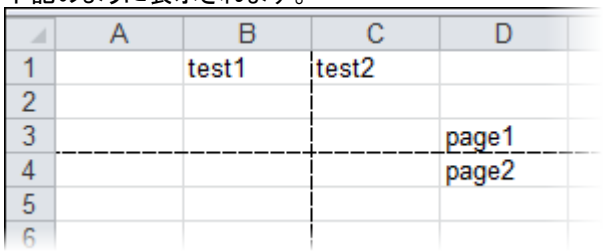
```
Visual Basic
C1XLBook1.Sheets(0)(2, 3).Value = "page1"
C1XLBook1.Sheets(0).Rows(2).PageBreak = True
C1XLBook1.Sheets(0)(0, 1).Value = "test1"
C1XLBook1.Sheets(0)(0, 2).Value = "test2"
C1XLBook1.Sheets(0).Columns(1).PageBreak = True
C1XLBook1.Sheets(0)(3, 3).Value = "page2"
```

```
' .xlsx ファイルを保存して開く
C1XLBook1.Save("c:\Save.xlsx")
System.Diagnostics.Process.Start("c:\Save.xlsx")
```

C# コードの書き方

```
C#
c1XLBook1.Sheets[0][2, 3].Value = "page1";
c1XLBook1.Sheets[0].Rows[2].PageBreak = true;
c1XLBook1.Sheets[0][0, 1].Value = "test1";
c1XLBook1.Sheets[0][0, 2].Value = "test2";
c1XLBook1.Sheets[0].Columns[1].PageBreak = true;
c1XLBook1.Sheets[0][3, 3].Value = "page2";
//.xlsx ファイルを保存して開く
c1XLBook1.Save(@"c:\Save.xlsx");
System.Diagnostics.Process.Start(@"c:\Save.xlsx");
```

3. プロジェクトを実行し、.xlsx ファイルを開きます。
4. エクセルでは、**ページレイアウト**タブに移動して、**グリッド線** 直下の **印刷** チェックボックスを選択します。ワークシートは下記のように表示されます。



ワークブックの計算モードを設定する

The **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 の列挙体では、3つのオプションが提供されます：**Manual** (計算が手動的に行われる)、**Auto** (計算が自動的に実行される)、**AutoNoTable** (テーブル以外に計算が実行される)

計算モードを設定するには、以下の手順に従います：

1. ツールボックスにある **C1XLBook** コンポーネントをダブルクリックして Form1 に追加します。
2. 下記コードを使用して簡単な数式を追加します。

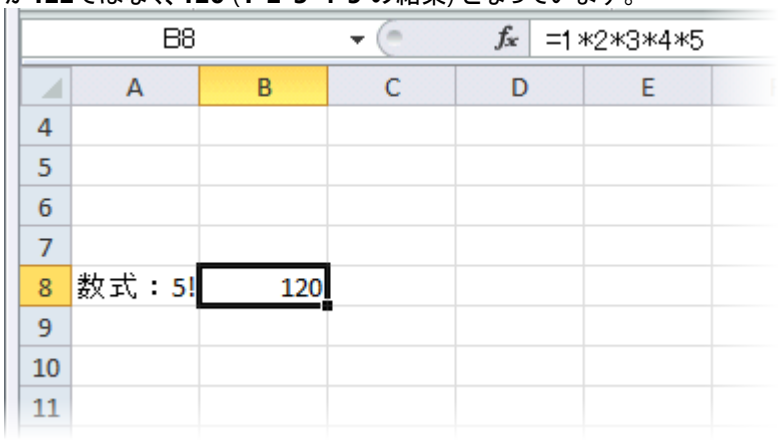
Visual Basic コードの書き方

```
Visual Basic
Dim sheet As XLSheet = c1XLBook1.Sheets(0)
' 簡単な数式
sheet(7, 0).Value = "Formula: 5!"
sheet(7, 1).Value = 122
sheet(7, 1).Formula = "1*2*3*4*5"
c1XLBook1.CalculationMode = CalculationMode.Auto
c1XLBook1.Save("c:\Save.xlsx")
System.Diagnostics.Process.Start("c:\Save.xlsx")
```

C# コードの書き方

```
C#
XLSheet sheet = c1XLBook1.Sheets[0];
// 簡単な数式
sheet[7, 0].Value = "Formula: 5!";
sheet[7, 1].Value = 122;
sheet[7, 1].Formula = "1*2*3*4*5";
c1XLBook1.CalculationMode = CalculationMode.Auto;
c1XLBook1.Save(@"c:\Save.xlsx");
System.Diagnostics.Process.Start(@"c:\Save.xlsx");
```

3. プロジェクトを実行し、エクセルファイルを開きます。CalculationMode を**Auto** に設定しているので、セル(7, 1) の値が**122**ではなく、**120** (**1*2*3*4*5** の結果) となっています。



OpenXml ファイルのインポートとエクスポート

Excel for .NET は Microsoft Excel 2007 OpenXml ファイルの読み取り／書き込みが行えるようになりました。OpenXml は Microsoft が Office 2007 に導入したオープンな標準形式です。OpenXml は XML をベースにし、BIFF8 などのプロプライエタリなバイナリ形式とは対照的にドキュメントが公開されているため、OpenXml ファイルはアプリケーションによって操作しやすくなっています。OpenXml ファイルは Zip 圧縮を使って圧縮された多数の XML ファイルを格納します。これらは圧縮されているため、OpenXml ファイルは通常、.doc や .xls ファイルなどの従来のドキュメントファイルよりもサイズが大幅に小さくなります。

Excel for .NET は OpenXml ファイル内のデータと書式情報のロードと保存は行いますが、数式のロードと保存は行いません。数式は、現時点で OpenXML 形式によってサポートされていない BIFF 形式で opaque としてコピーされます。数式を含むファイルをロードしてそれらを保存する場合、数式は削除されます。これは、数式を保持する従来の .xls（BIFF8）形式と異なっています。

OpenXml 形式をサポートするために、**C1XLBook** の **Load** および **Save** メソッドは ファイルのロード／保存時に使用するファイル形式の指定に使用される **FileFormat** パラメータを取るオーバーロードを受け取ります。

ファイル名を指定しない場合は、**Excel for .NET** はファイル形式をファイル名の拡張子から推測します。つまり、「XLSX」と「ZIP」拡張子のファイルはデフォルトで OpenXml ファイルとしてロードおよび保存されます。他のファイルは BIFF8（.xls）形式としてロードおよび保存されます。

たとえば、次のようになります。

C# コードの書き方

```
C#
// ファイル拡張子に依存してロードおよび保存
c1Excel1.Load("somefile.xls"); // biff8ファイルを読み込む
c1Excel1.Save("somefile.xlsx"); //OpenXmlとして保存
c1Excel1.Save("somefile.zip"); //OpenXmlとして保存
```

```
// FileFormat を指定してロードおよび保存
c1Excel1.Load("somefile.xls", FileFormat.Biff8);
c1Excel1.Save("somefile.xlsx", FileFormat.OpenXml);
```

ストリームでのファイルのロード／保存時に形式を指定することもできます。**FileFormat** を指定しない場合は、**Excel for .NET** は BIFF8 形式をデフォルトとして使用します。

ここで小さい動作変更が暗黙に含まれていることに注意してください。以下のステートメントを考えます。

```
C1Excel1.Save("somefile.xlsx");
```

Excel for .NET の以前のバージョンでは、このステートメントで BIFF8 ファイルが（無効な拡張子で）保存されました。今は、このステートメントで OpenXml ファイルが（有効な拡張子で）保存されます。このようなコードがアプリケーションに含まれている場合、アップグレード時にそれを以下のように変更する必要があります。

C# コードの書き方

C#

```
// 無効な拡張子を持つファイルを意図的に保存
C1Excel1.Save("somefile.xlsx", FileFormat.Biff8);
```

ブックを OpenXml ファイルにエクスポートするには、以下の手順を実行します。

1. 既存のブックをロードします。

   Visual Basic コードの書き方

   Visual Basic

   ```
   Dim wb As New C1XLBook()
   wb.Load("C:\test.xlsx")
   'あるいは
   Dim wb As New C1XLBook()
   wb.Load("C:\test.xlsx", C1.C1Excel.FileFormat.OpenXml)
   ```

   C# コードの書き方

   C#

   ```
   C1XLBook wb = new C1XLBook();
   wb.Load(@"C:\test.xlsx");
   // あるいは
   C1XLBook wb = new C1XLBook();
   wb.Load(@"C:\test.xlsx", C1.C1Excel.FileFormat.OpenXml);
   ```

2. ブックを OpenXml 形式ファイルにエクスポートします。

   Visual Basic コードの書き方

   Visual Basic

   ```
   Dim wb As New C1XLBook()
   ' 何かコンテンツを追加
   Dim sheet As XLSheet = wb.Sheets(0)
   Dim i As Integer
   For i = 0 To 9
           sheet(i,0).Value = i + 1
   ```

```
Next i
' OpenXml 形式ファイルにエクスポート
wb.Save("C:\test.xlsx")
' あるいは
' OpenXml 形式ファイルにエクスポート
wb.Save("C:\test.xlsx", C1.C1Excel.FileFormat.OpenXml)
```

C# コードの書き方

C#

```
C1XLBook wb = new C1XLBook();
// 何かコンテンツを追加
XLSheet sheet = wb.Sheets[0];
for (int i = 0; i <= 9; i++)
        {
        sheet[i,0].Value = i + 1;
        }
// OpenXml 形式ファイルにエクスポート
wb.Save(@"C:\test.xlsx");
// あるいは小計を作成する
// OpenXml 形式ファイルにエクスポート
wb.Save(@"C:\test.xlsx", C1.C1Excel.FileFormat.OpenXml);
```

小計を作成する

下記のサンプルコードは、ワークブックの書式設定方法の例となります。

1. ツールボックスにある**C1XLBook** コンポーネントをダブルクリックして Form1 に追加します。
2. **コードビュー**に移動して、フォーム上部に下記の文を追加します。
   - Import C1.C1Excel (Visual Basic)
   - using C1.C1Excel; (C#)
3. 以下のコードを**Form_Load** イベントに追加します。

Visual Basic コードの書き方

Visual Basic

```
Private Sub Form1_Load(sender As Object, e As EventArgs)
   Dim book As New C1XLBook()
   Dim sheet As XLSheet = book.Sheets(0)
   Dim totalStyle As New XLStyle(book)
   totalStyle.Font = New Font(book.DefaultFont, FontStyle.Bold)
   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
   book.Save("c:\mybook.xls")
   System.Diagnostics.Process.Start("C:\mybook.xls")
End Sub
```

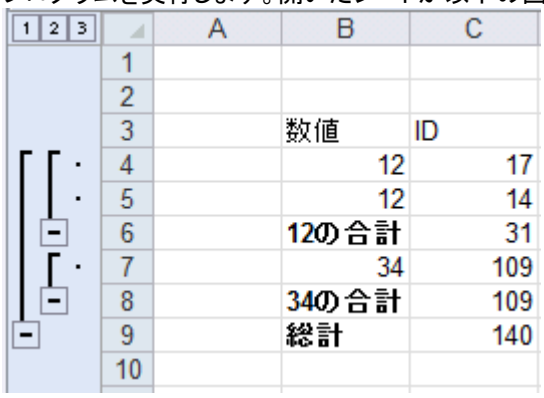
C# コードの書き方

C#

```
private void Form1_Load(object sender, EventArgs e)
        {
            C1XLBook book = new C1XLBook();
            XLSheet sheet = book.Sheets[0];
            XLStyle totalStyle = new XLStyle(book);
            totalStyle.Font = new Font(book.DefaultFont, FontStyle.Bold);
            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;
            book.Save(@"c:\mybook.xls");
            System.Diagnostics.Process.Start(@"C:\mybook.xls");
        }
```

4. プログラムを実行します。開いたシートが以下の画像のように表示されます。



	A	B	C
1			
2			
3		数値	ID
4		12	17
5		12	14
6		12の合計	31
7		34	109
8		34の合計	109
9		総計	140
10			

SUBTOTAL の数式で指定された行の合計が取得されます。

CSV ファイルへの保存と読み込みを行う

Excel for .NET では、コンマ区切りファイルの保存と読み込みを対応します。CSV ファイルは、数値とテキストを含める表形式データを読みやすく表示するには、データを普通テキストで格納する通常なファイル形式です。

以下のサンプルコードは、.csv ファイルの保存方法と読み込み方法の例となります。

1. ツールボックスにある **C1XLBook** コンポーネントをダブルクリックして Form1 に追加します。
2. **コードビュー**に移動して、フォーム上部に下記の文を追加します。
   - Import C1.C1Excel (Visual Basic)
   - using C1.C1Excel; (C#)
3. 以下のコードを**Form_Load** イベントに追加すると、10個の値を持つシートを作成し、ワークブックを .csv 形式で保存します。

Visual Basic コードの書き方

Visual Basic

```
Private Sub Form1_Load(sender As Object, e As EventArgs)
   Dim sheet As XLSheet = c1XLBook1.Sheets(0)
    For i As Integer = 0 To 9
          sheet(i, 0).Value = i + 1
          sheet(i, 1).Value = 10 - i;
   Next
    sheet.SaveCsv("c:\test.csv")
   System.Diagnostics.Process.Start("C:\test.csv")
 End Sub
```

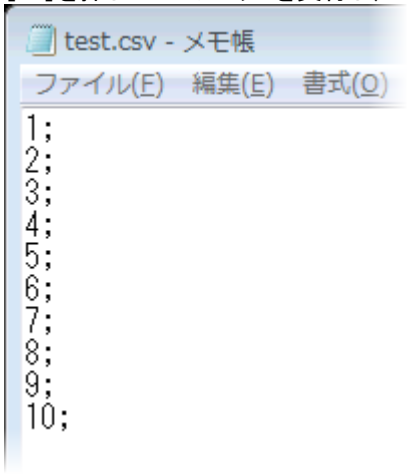
C# コードの書き方

C#

```
private void Form1_Load(object sender, EventArgs e)
        {
            XLSheet sheet = c1XLBook1.Sheets[0];
             for (int i = 0; i= 9; i++)
            {
sheet[i, 0].Value = i + 1;
sheet[i, 1].Value = 10 - i;
            }
             sheet.SaveCsv(@"c:\test.csv");
            System.Diagnostics.Process.Start(@"C:\test.csv");

        }
```

4. [F5]を押してプロジェクトを実行し、.csv ファイルを表示します。



5. test.csv ファイルにいくつかの新規の値を追加します。新規に追加された値を保存するには、ファイルを再度保存する必要があります。**Form1_Load** イベントに下記のように **LoadCsv** と **SaveCsv** メソッドのコードを追加することで実現できます。

Visual Basic コードの書き方

Visual Basic

```
Private Sub Form1_Load(sender As Object, e As EventArgs)
   Dim sheet As XLSheet = c1XLBook1.Sheets(0)
   For i As Integer = 0 To 9
          sheet(i, 0).Value = i + 1
          sheet(i, 1).Value = 10 - 1
   Next
   sheet.SaveCsv("c:\test.csv")
   sheet.LoadCsv("c:\test.csv")

   For i As Integer = 10 To 19
          sheet(i, 0).Value = i + 1
          sheet(i, 1).Value = 10 -1
   Next

   sheet.SaveCsv("c:\test.csv")
   System.Diagnostics.Process.Start("C:\test.csv")
End Sub
```

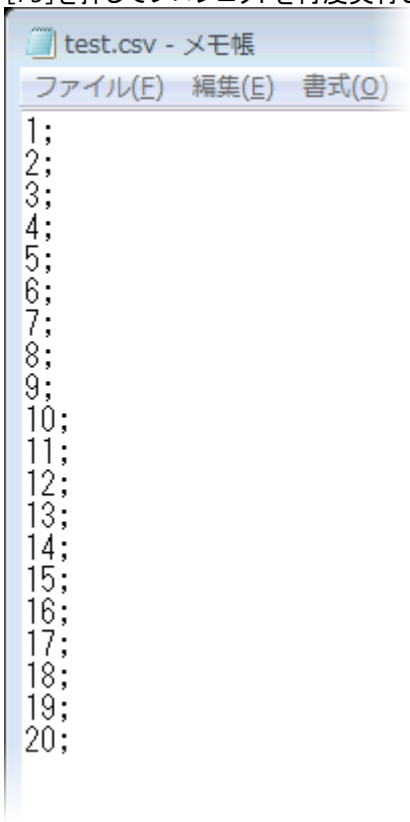
C# コードの書き方

C#

```
private void Form1_Load(object sender, EventArgs e)
        {
           XLSheet sheet = c1XLBook1.Sheets[0];
           for (int i = 0; i = 9; i++)
           {
sheet[i, 0].Value = i + 1;
sheet[i, 1].Value = 10 - i;
           }
          sheet.SaveCsv(@"c:\test.csv");
          sheet.LoadCsv(@"c:\test.csv");

           for (int i = 10; i = 19; i++)
           {
sheet[i, 0].Value = i + 1;
sheet[i, 1].Value = 20 - i;
           }
            sheet.SaveCsv(@"c:\test.csv");
           System.Diagnostics.Process.Start(@"C:\test.csv");
        }
```

6. [F5]を押してプロジェクトを再度実行し、.csv ファイルを表示します。



マクロ有効 Excel ファイルのロードおよび保存

Excel for .NET は、OpenXMLMacro ファイル形式のロードおよび保存をサポートします。Visual Basic Application（VBA）言語で記述されたマクロ有効 Excel ファイル（XLSM）を C1Excel にロードおよび保存できるようになりました。XLSM は、

Microsoft Excel によって作成される Excel マクロ有効スプレッドシートです。VBA コードによってプログラムされたマクロが埋め込まれています。XLSM ファイルは、マクロが有効化されていること以外は XLSX ファイルと同じです。

次のコードは、VBA マクロを含むテンプレート xls ファイルをロードし、XLSM 形式でファイルを保存する方法の一例です。

1. ツールボックスの Button コントロールをダブルクリックしてフォームに追加します。ボタンの **Name** プロパティと **Text** プロパティを変更します。たとえば、ボタンの **Name** は btnLoad に、ボタンに表示される **Text** は Load に設定できます。
2. ツールボックスの **C1XLBook** コンポーネントをダブルクリックしてフォームに追加します。
3. フォームの Button コントロールをダブルクリックして **btnLoad_Click** イベントを追加し、コードビューに切り替えます。
4. 次のコードを**フォーム**に追加して C1.C1Excel 名前空間を追加します。

   Visual Basic でコードを書く場合

   Visual Basic
   ```
   Imports C1.C1Excel
   ```

   C# でコードを書く場合

   C#
   ```
   using C1.C1Excel;
   ```

5. 次のコードを **btnLoad_Click** イベントに追加して、ワークブックを .xlsm 形式で保存します。

   Visual Basic でコードを書く場合

   Visual Basic copyCode
   ```
   ' ブックにコピーします
   Dim xlsFileName As String = "..\..\XlsxTestBin.xlsm"
   C1XLBook1.Load(xlsFileName)
   C1XLBook1.Sheets(0)(0, 0).Value = "It is copy!"
   xlsFileName = xlsFileName.Replace(".xlsm", "Modified.xlsm")
   C1XLBook1.Save(xlsFileName)

   ' Excel ファイルを実行します
   System.Diagnostics.Process.Start(xlsFileName)
   ```

   C# でコードを書く場合

   C# copyCode
   ```
   // ブックにコピーします
   string xlsFileName = "..\\..\\XlsxTestBin.xlsm";
   c1XLBook1.Load(xlsFileName);
   c1XLBook1.Sheets[0][0, 0].Value = "It is copy!";
   xlsFileName = xlsFileName.Replace(".xlsm", "Modified.xlsm");
   c1XLBook1.Save(xlsFileName);

   // Excel ファイルを実行します
   System.Diagnostics.Process.Start(xlsFileName);
   ```

6. **[F5]**キーを押してプロジェクトを実行すると、.xslm ファイルが以下に示すように表示されます。