主要特性

下面列出一些你也许觉得有用的ComponentOne Excel for .NET主要特性

- 一个命令就能保存或者加载一个Excel文件
- Excelfor.NET简单易用。它能够让你只使用一条命令保存或加载工作簿或者操作表格控件制成的表。
- 在单个格子中读或写数据
- 加载或产生一个C1XLBook后，如果你的表是简单的表格，那么你就能够在每个单独的表中访问数据。例如：
  XLSheetsheet=C1XLBook.Sheets[0];
  sheet[0,0].Value=DateTime.Now;
- 在每个格子中格式化数据
  每个格子相关的格式象数据一样保存到格子中，这样就很容易访问。例如：

  ```
  XLStylestyle=newXLStyle(c1XLBook1);
  style.Format="dd-MM-yyyy";
  style.Font=newFont("CourierNew",14);
  XLSheetsheet=C1XLBook.Sheets[0];
  sheet[0,0].Value=DateTime.Now;
  sheet[0,0].Style=style;
  ```

- 使用Excel for .NET导出到XLS文件
  其他的ComponentOne组件使用Excel for .NET导出到XLS文件。例如，C1Report使用Excel for .NET产生XLS版本的报告，这样任何有微软Excel副本的人就能看或者编辑这个报告了。
- 不需要使用微软的Excel就能读或写.xls和.xlsx文件
  Excel for .NET 读或写.xls文件（Excel97或以上版本）和xlsx文件（OpenXml 格式），其中xlsx文件可以被重用和简单地被替换或者压缩成更小的文件大小。
- 在一个格子中产生图片并将图片放置到正确的位置。
  你不仅能添加图片到格子中，现在你还可以定义格子的大小，在格子中将图片放置在正确的位置，以及定义为了适应格子的大小图片是否需要按比例缩放，剪切或伸展。
- 保存文件到流以及从流中加载文件
  工作簿现在可以使用Load和Save方法重载来直接从记忆流中读取，所以你就不是必须使用临时文件了。
- 下面是给一个表格的页头和页脚添加图片。
  使用在XLPrintSettings类中的属性就可以添加图片到表格的页头的或页脚的左、中和右的位置。

Excel for .NET快速入门

本快速入门指南将使你熟悉Excelfor.NET的一些特性。在快速入门中你将学到怎么样给程序添加一个C1XLBook，给工作簿添加格式化好的数据，以及保存和打开XLS文件。

步骤一：创建程序

在这步中你将添加一个C1XLBook 组建到你的窗体中。每个Excel文件由一个或多个表组成。

1. 创建一个新的.NET2.0程序
2. 工具盒中，双击C1XLBook 图标就会将C1XLBook 组件添加到你的程序中。C1XLBook 组件会显示在窗体下的组件托盘里。
3. 双击窗体添加Form1_Load事件并且切换到代码视图。
4. 窗体的上面添加Imports (Visual Basic)或者using (C#)声明到代码中，这样你就可以使用在C1.C1Excel命名空间中的所有命名。

Visual Basic

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

C#

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

现在你有了一个C1XLBook，你可以开始给它添加内容了。

步骤二：给C1XLBook添加内容

当你还在VisualStudio程序的代码视图中，将下面的代码添加到步骤一创建的Form_Load事件中，这些代码就会给Excel文件添加内容。

Visual Basic

```
Visual Basic
' Add content to the sheet.
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#

```
C#
// Add content to the sheet.
```

```
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;
}
```

当你运行程序，XLS文件前十行中的前三列将会填充数字。

步骤三：格式化内容

下面你将使用样式格式化内容。此步骤中的代码应该添加在Form_Load事件中步骤二的代码之后。

1. 添加下面的代码来创建两个样式：style1和style2.

   Visual Basic

   Visual Basic

   ```
   'Add style 1.
   Dim style1 As New XLStyle(C1XLBook1)
   style1.Font = New Font("Tahoma", 9, FontStyle.Bold)
   style1.ForeColor = Color.RoyalBlue
   ' Add style 2.
   Dim style2 As New XLStyle(C1XLBook1)
   style2.Font = New Font("Tahoma", 9, FontStyle.Italic)
   style2.BackColor = Color.RoyalBlue
   style2.ForeColor = Color.White
   ```

   C#

   Title Text

   ```
   // Add style 1.
   XLStyle style1 = new XLStyle(c1XLBook1);
   style1.Font = new Font("Tahoma", 9, FontStyle.Bold);
   style1.ForeColor = Color.RoyalBlue;
   // Add style 2.
   XLStyle style2 = new XLStyle(c1XLBook1);
   style2.Font = new Font("Tahoma", 9, FontStyle.Italic);
   style2.BackColor = Color.RoyalBlue;
   style2.ForeColor = Color.White;
   ```

2. 然后添加下面的代码来给内容提供样式。

   Visual Basic

   Visual Basic

   ```
   For i = 0 To 9
       ' Apply styles to the content.
       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#

```
C#
for (i = 0; i <= 9; i++)
{
    // Apply styles to the content.
   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, 2].Style = style2;
           sheet[i, 2].Style = style1;
   }
}
```

步骤四：保存和打开XLS文件

最后，添加下面的代码来保存和加载工作簿。这些代码需要添加在Form_Load事件中步骤三的代码之后。

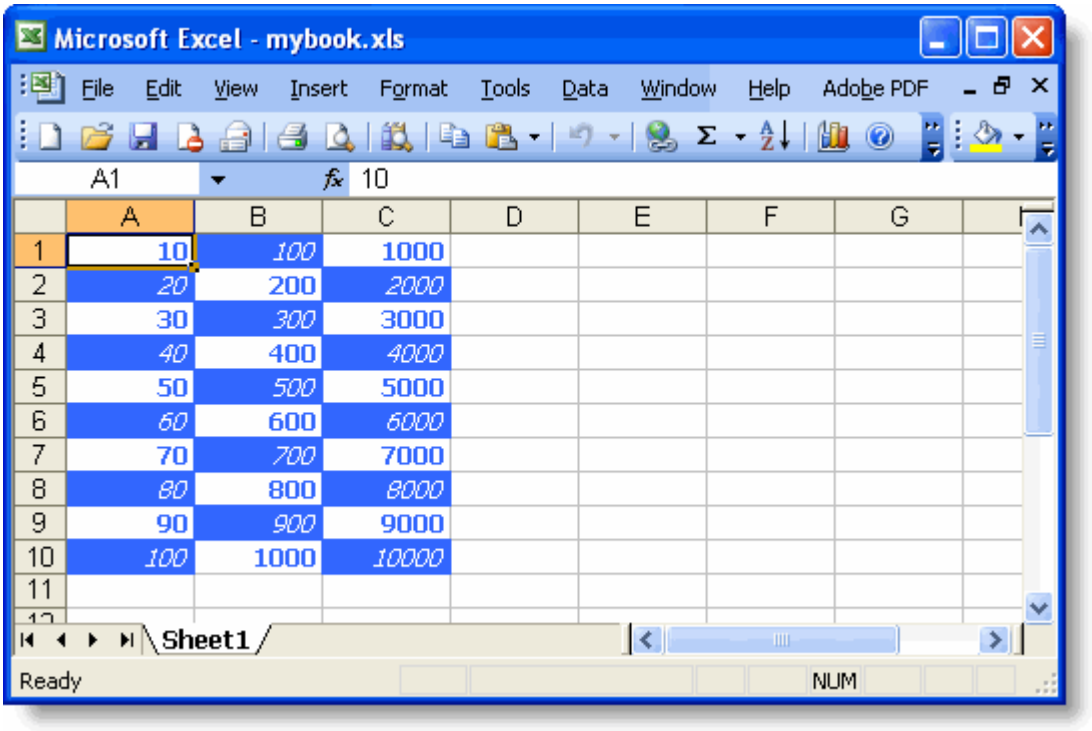
Visual Basic

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

C#

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

运行程序并且观察:



格式化好的内容添加到了Excel文件中。

恭喜！你已经完成了Excelfor.NET快速入门。

使用Excel for .NET

本主题介绍了如何使用 Excel for .NET来创建 XLS 文件，并且解释了在创建XLS文件过程中用到的主要的Excel for .NET对象，例如工作表，单元格行集合，单元格列集合，以及样式对象等等。

创建XLS文档

使用Excelfor.NET 来创建一个新的XLS文档，下面的三步是必须的：

1. 添加一个C1XLBook组件到你的项目中或者在代码中创建一个C1XLBook对象。每一个C1XLBook对象都是由一个或者多个工作表构成（XLSheet对象）。
2. 添加内容到工作表，每一个工作表都包含了很多个单元格（XLCell对象），单元格对象有Value和Style属性，可以更改单元格内容或者样式。
3. 通过C1XLBook对象的Save方法保存文档

下面的例子创建了一个新的 Excel 文件，该文档默认包含了一个工作表，然后给第一列的前100个单元格赋值1到100。如果你已经在表单上添加了C1XLBook组件，你可以忽略代码中第一步。

Visual Basic

Visual Basic

```
' 步骤1： 创建一个新的工作薄对象
Dim C1XLBook1 As New C1XLBook()
' 步骤2:给单元格内部添加内容
Dim sheet As XLSheet = C1XLBook1.Sheets(0)
Dim i As Integer
For i = 0 To 99
        sheet(i, 0).Value = i + 1
Next i
' 步骤3:保存文件
C1XLBook1.Save("c:\temp\hello.xls")
```

C#

C#

```
// 步骤1： 创建一个新的工作薄对象
C1XLBook c1XLBook1 = new C1XLBook();
 // 步骤2:给单元格内部添加内容
XLSheet sheet = c1XLBook1.Sheets[0];
for (int i = 0; i < 100; i++)
       {
        sheet[i, 0].Value = i + 1;
        }
 // 步骤3:保存文件
c1XLBook1.Save(@"c:\temp\hello.xls");
```

第二步是非常有意思的，从新创建的Excel 工作薄（C1XLBook）对象中获取第一个工作表（XLSheet）对象，在创建新的C1XLBook对象的时候，这个工作表对象会被自动创建。接下来通过单元格索引访问单元格对象，并且赋值1到100.

注意到通过XLSheet对象的行列索引来访问单元格的话，XLSheet 对象会根据需要自动创建单元格，这将使得填充工作

表变得非常容易。如果你想获取工作表的大小，可以通过XLSheet的Rows.Count以及Columns.Count属性获得。

除了可以给单元格填充值之外，你还可以给单元格设置样式，创建XLStyle对象，更改样式后，将其赋给单元格就可以了。下面的代码演示了将偶数行的单元格设置为粗体红色，而奇数行的单元格设置为斜体蓝色。如果你已经在表单上添加了C1XLBook组件，你可以忽略代码中的第一步。

Visual Basic

Visual Basic

```
' 步骤1：创建一个新的工作薄对象
Dim C1XLBook1 As New C1XLBook()
' 步骤二：为奇偶行创建样式
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
' 步骤三：填充内容到单元格
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
 ' 步骤四：保存文件
C1XLBook1.Save("c:\temp\hello.xls")
```

C#

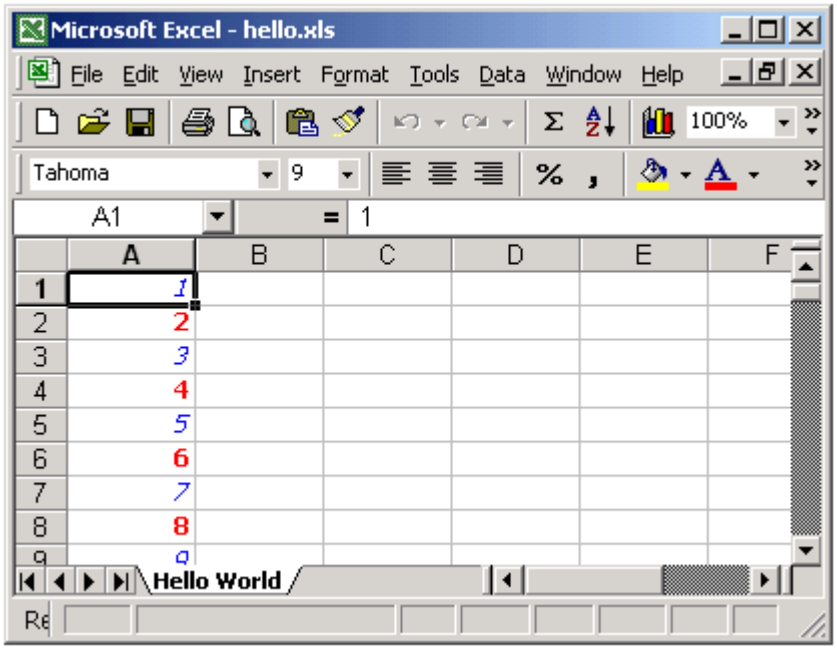
C#

```
// 步骤1：创建一个新的工作薄对象
C1XLBook c1XLBook1 = new C1XLBook();
// 步骤2：为奇偶行创建样式
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;
// 步骤三：填充内容到单元格
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;
}
// 步骤四：保存文件
c1XLBook1.Save(@"c:\temp\hello.xls");
```

这一段代码与上一个例子类似，主要的不同在于第二步，为奇偶行创建了不同的样式，这些新的样式在第三个步骤中会随同单元格内容一起被设置到单元格上面。

使用微软 Excel打开上面保存的文件，截图如下：



工作表（Worksheets）

工作表（Worksheet）就是Excel文件中每一个单独的网格页面，通过C1XLBook对象的Sheets属性可以访问到每一个工作表（Worksheet）对象，而每一个工作表对象都包含了很多单元格行以及单元格列，通过行，列索引可以检索到每一个单元格。

当访问工作表（XLSheet）对象的索引的时候，其Rows和Columns集合会根据需要自动扩展。在下面的代码中XLSheet对象并没有1001行，当通过索引访问10001行的时候，新的行将会被自动添加，并且返回一个实际的行。通过XLColumn和XLCell索引访问单元格也是一样的。这跟.NET中大多数集合的表现方式是不一样的，但是非常易于数据填充。

Visual Basic

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

C#

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

Rows和Columns

工作表（XLSheet）对象对外公开了Rows和Columns两个集合属性用以操纵工作表上的每一行或者每一列，XLRow和XLColumn代表了工作表上面的行，列对象，可以设定行高或者列宽，是否可见以及样式等等。如果你没有设定任何值，XLRow和XLColumn会返回工作表的默认值（参见DefaultRowHeight和DefaultColumnWidth属性）。

XLRow和XLColumn 对象的默认纬度为1，这是工作表的默认设置。

单元格

在工作表（XLSheet）对象上可以直接通行，列索引操作工作表上的每一个单元格。XLCell代表了每一个单元格对象，可以给单元格赋值或者设定样式。

与Rows集合和Columns集合一样，通过单元格索引访问一样可以自动扩展工作表。

Visual Basic

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

C#

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

如果工作表少于11行或者11列，Rows或者Columns将会被自动扩充，索引为［10,10］的单元格会被返回。

由于工作表可以自动扩充，通过索引检索单元格将永远不会返回空引用（null 对象）。如果你想检查特定索引位置的单元格是否存在的时候，你可以使用工作表（XLSheet）对象的GetCell方法来进行访问，如果返回空引用则不存在。

XLCell对象上的Value属性包含了单元格的内容，这个属性的类型是object，通常可能是字符串，数字，布尔，日期类型或者空值，其它类型的值不能被保存到Excel文件中。

XLCell对象上的Style属性包含了单元格的样式，定义了单元格的外观。如果该属性被设置为null，单元格会使用工作表的默认样式。通过XLStyle对象可以设定单元格的字体，颜色，对齐方式以及格式等等。

样式

XLStyle类型定义了工作表上行，列以及单元格的外观样式，XLStyle对象包含了很多特定的属性来设定样式，例如字体，对齐方式，颜色，格式等等。需要注意的是并不是XLStyle对象上面的每一个属性都需要设定，如果你只设定了该对象上的某一个属性，单元格在绘制的时候会应用这个属性，而其他的设置会使用工作表的默认设置。

Excel for .NET 常见问题

下面是Excelfor.NET用户遇到的一些常见问题：

Excelfor.NET是否支持计算公式和单元格命名区域？

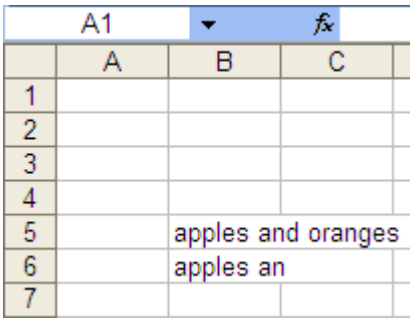
Excelfor.NET同时支持计算公式和单元格命名区域。ComponentOneStudios的2009 v2版本已经完全支持添加单元格计算公式，参见下面的例子：

C#

C#

```
private void Form1_Load(object sender, EventArgs e)
        {
            C1.C1Excel.XLSheet sheet = c1XLBook1.Sheets[0];
            sheet[4, 1].Value = "apples and oranges";
            sheet[5, 1].Value = "apples an";       // <-- result of the formula
            sheet[5, 1].Formula = "LEFT(B5,9)";
            c1XLBook1.Save(@"c:\mybook.xls");
            System.Diagnostics.Process.Start(@"c:\mybook.xls");
        }
```

运行程序结果如下：



设置的计算公式先会从 B5 单元格中获取数据，然后从该字符串的左边开始计数，取9个字符，就是“apples an”。

Excelfor.NET可以在ASP.NET2.0的页面上使用吗？

是的，Excelfor.NET完全支持ASP.NET2.0，如果你需要在一个 aspx页面上显示Excel for .NET控件的话，可以打开组件设计器的页面，拖拽一个C1XLBook 控件到你的Web页面上就可以了。

基于任务的Excel for .NET帮助文档

基于任务的帮助文档要求你具有使用VisualStudio.NET 进行编程的经验，完成本文档中的任务后，你就可以创建工程来展示 Excelfor.NET的特性，熟悉和了解Excelfor.NET的应用场景。

每一个基于任务的帮助话题都假定你已经创建了一个新的 .NET 项目，并且已经添加了合适的命名空间引用语句，比如 C# 中，使用 （usingC1.C1Excel;）， 而 Visual Basic则使用 （ImportsC1.C1Excel）。

添加内容到工作表

按照下面的步骤将创建一个新的工作表，并且添加值到第一列的前十个单元格：

1. 在工具箱上双击C1XLBook 组件，将会添加该组件到表单上。
2. 添加值到第一列的前十个单元格

Visual Basic

```
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#

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

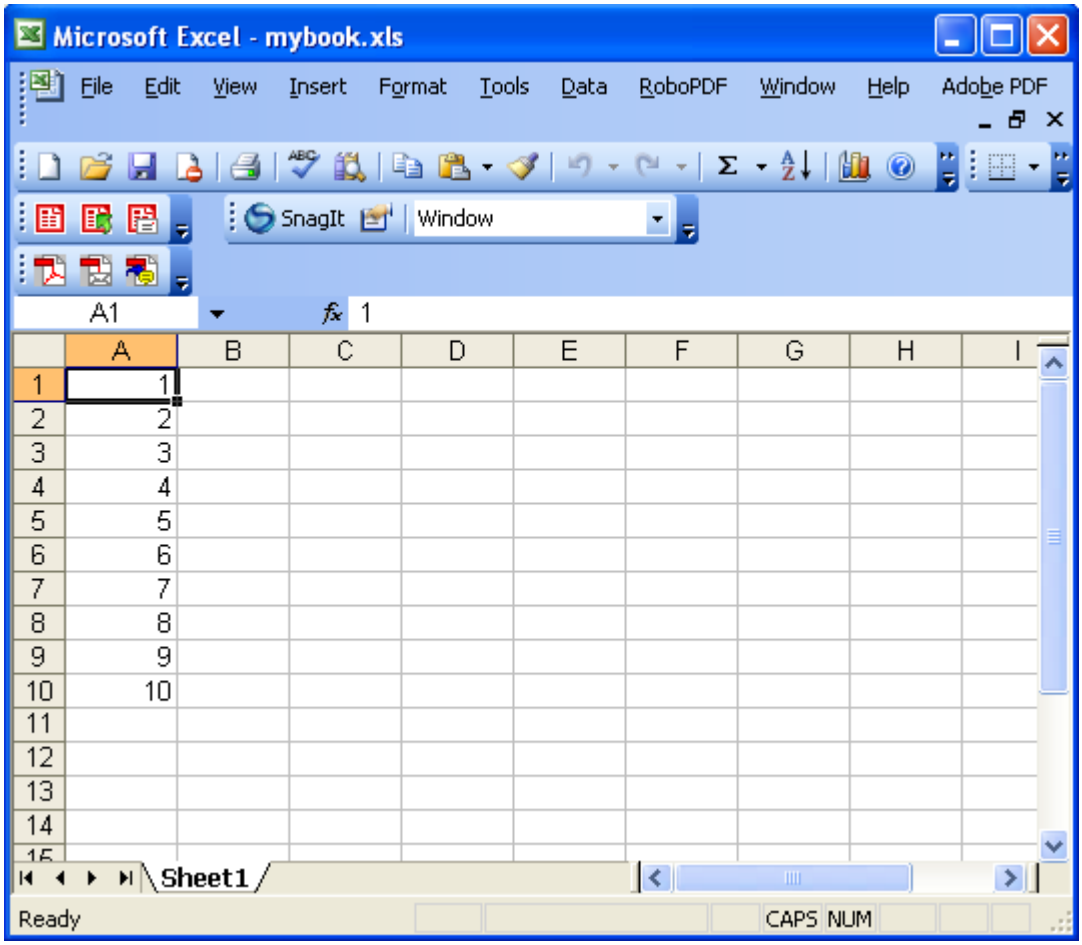
3. 保存文件，并且使用 Excel 打开该文件：

Visual Basic

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

C#

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



合并单元格

通过C1Excel 的 MergedCells 属性可以合并指定区域的单元格，该属性通常被用来在工作表上检测，创建以及删除合并的单元格。每一个合并的单元格区域都会用一个C1.C1Excel.XLCellRange 对象来代表。添加删除单元格行或者单元格列，合并的单元格会被保留。

按照下面的步骤进行单元格合并：

1. 在工具箱上双击C1XLBook 组件，将会添加该组件到表单上。
2. 按照下面的代码，先创建两个单元格区域（_ColRange和_RowRange），然后添加这两个区域到MergedCells中，最后给合并的单元格添加样式。你可以把这些代码写在Form1_Load的事件处理函数中，代码如下：

C#

C#

```
private void Form1_Load(object sender, EventArgs e)
{
        //选择需要合并的单元格区域
        XLCellRange _ColRange = new C1.C1Excel.XLCellRange(4, 6, 0, 8);
        XLCellRange _RowRange = new C1.C1Excel.XLCellRange(9, 21, 3, 4);
         // 为合并的单元格设定文本
        c1XLBook1.Sheets[0][4, 0].Value = "Merged Cells";
        c1XLBook1.Sheets[0][9, 3].Value = "Merged Cells";
        //合并单元格
        c1XLBook1.Sheets[0].MergedCells.Add(_ColRange);
```

```
        c1XLBook1.Sheets[0].MergedCells.Add(_RowRange);
        //定义样式，并且应用样式到合并单元格
        XLStyle _Colstyle = new C1.C1Excel.XLStyle(c1XLBook1);
        _Colstyle.BackColor = Color.Yellow;
        XLStyle _Rowstyle = new XLStyle(c1XLBook1);
        _Rowstyle.BackColor = Color.LightBlue;
        c1XLBook1.Sheets[0][4, 0].Style = _Colstyle;
        c1XLBook1.Sheets[0][9, 3].Style = _Rowstyle;
        //保存并且打开工作薄
        c1XLBook1.Save(@"c:\mybook.xls");
        System.Diagnostics.Process.Start(@"C:\mybook.xls");
}
```

格式化单元格

完成下面步骤:

1. 在工具箱上双击C1XLBook 组件，将会添加该组件到表单上。
2. 创建一个新的样式，然后设定样式属性：

Visual Basic

```
Visual Basic
Dim style1 As New XLStyle(C1XLBook1)
style1.ForeColor = Color.Gold
style1.BackColor = Color.Blue
style1.Format = "$  .00"
```

C#

C#

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

3. 给工作表的第一列单元格添加内容和样式：

Visual Basic

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#

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. 保存文件，然后使用 Excel 打开该文件。

Visual Basic

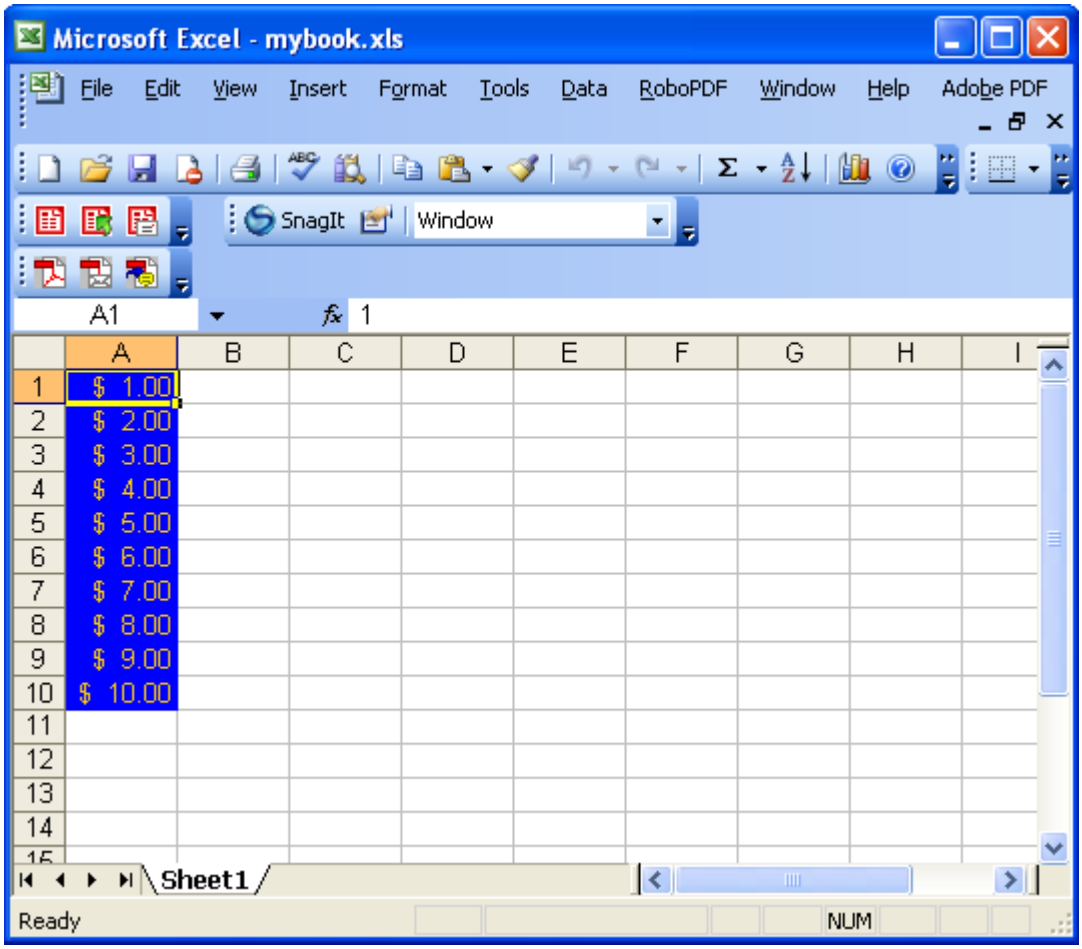
Visual Basic

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

C#

C#

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



在工作薄之间拷贝单元格行

按照下面的步骤可以从一个工作薄中拷贝一些单元格行到另一个工作薄：

1. 加载一个现存的工作薄：

   Visual Basic

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

   C#

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

2. 创建一个新的工作表（XLSheet）对象

   Visual Basic

   Visual Basic
   ```
   Dim xb As New C1XLBook()
   ```

```
xb.Sheets.Add("Test")
```

C#

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

3. 逐行从现有的工作薄中拷贝数据，然后复制到新的工作表：

Visual Basic

```
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#

```
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. 保存文件，使用 Excel 打开新创建的文件：

Visual Basic

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

C#

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

添加图片到单元格

按照下面的方法可以添加图片到单元格或者工作表，点击下面的链接可以看见详细的步骤：

方法1: 直接给单元格的 Value 属性赋值一个图片对象。

按照这个方法添加到工作表的图片会保持原始尺寸，图片的左上角会对齐单元格的左上角。

1. 加载一个现存的工作薄，创建一个新的工作表，并且添加内容。

   Visual Basic

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

   C#

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

2. 加载图片，并且赋给单元格的Value属性。

   Visual Basic

   ```
   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#

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

3. 保存文件，然后使用 Excel 打开该文件：

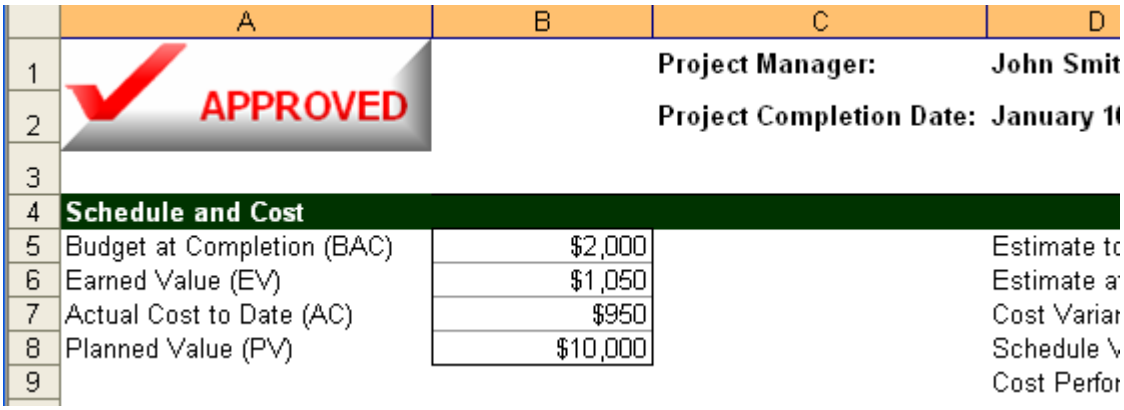
   Visual Basic

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

   C#

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

在这个例子中图片会作为第一个单元格的值，并且保持其原有尺寸出现在单元格中。



方法 2: 创建一个新的XLPictureShape 对象，设置属性，然后将该对象设置为单元格的 Value 属性。

使用这个方法添加的图片，开发人员可以改变大小，旋转角度，透明度，对比度以及边框等等。

1. 加载现有的工作薄，添加一个新的工作表，并且增加一些内容。

   Visual Basic

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

   C#

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

2. 创建一个XLPictureShape对象，更改属性，然后赋给单元格的值属性。

   Visual Basic

   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
   ' 设定图片到工作表的第一个单元格
   Dim sheet As XLSheet = wb.Sheets("Forecasting Report")
   sheet(0, 0).Value = pic
   ```

   C#

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

```
pic.LineColor = Color.DarkRed;
pic.LineWidth = 100;
// 设定图片到工作表的第一个单元格
XLSheet sheet = wb.Sheets("Forecasting Report");
sheet[0,0].Value = pic;
```

3. 保存工作薄，然后使用 Excel 打开：

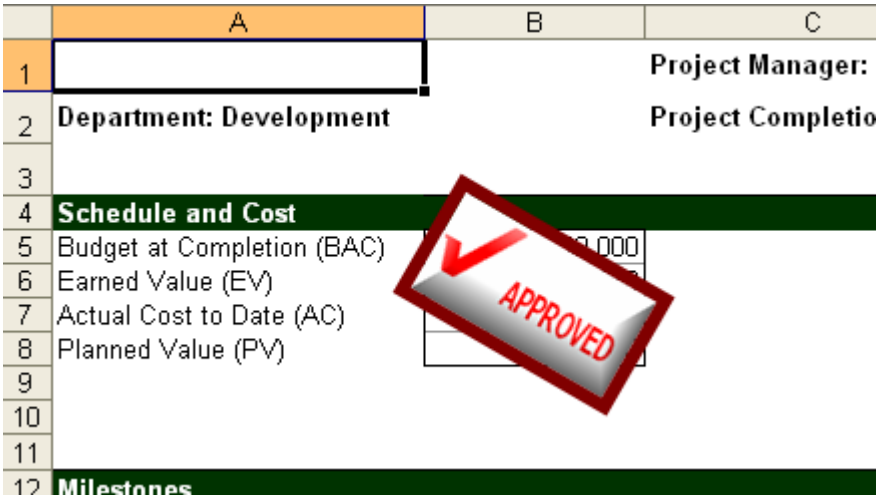
Visual Basic

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

C#

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

在这个例子中图片将会作为第一个单元格的值，然后旋转 30°，添加深红色的边框。 由于我们已经指定了这个图片的位置，它并不会出现在第一个单元格中。



方法3: 创建一个XLPictureShape 对象，设定其属性，然后添加该对象到工作表的ShapeCollection 集合中。

这个方法使用XLPictureShape 对象的构造器来指定图片的边界，这个图形对象被直接添加到工作表的ShapeCollection 集合中，而不是绑定到单元格上面。

1. 加载现有的工作薄，添加一个新的工作表，并且增加一些内容。

Visual Basic

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

C#

C#

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

2. 创建一个XLPictureShape对象，更改属性，然后添加到工作表的 ShapeCollection 集合中去

Visual Basic

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
' add the pic to specified sheet's ShapeCollection
Dim sheet As XLSheet = wb.Sheets("Forecasting Report")
sheet.Shapes.Add(pic)
```

C#

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;
// add the pic to specified sheet's ShapeCollection
XLSheet sheet = wb.Sheets("Forecasting Report");
sheet.Shapes.Add(pic)
```

3. 保存工作薄，然后使用 Excel 打开：

Visual Basic

Visual Basic

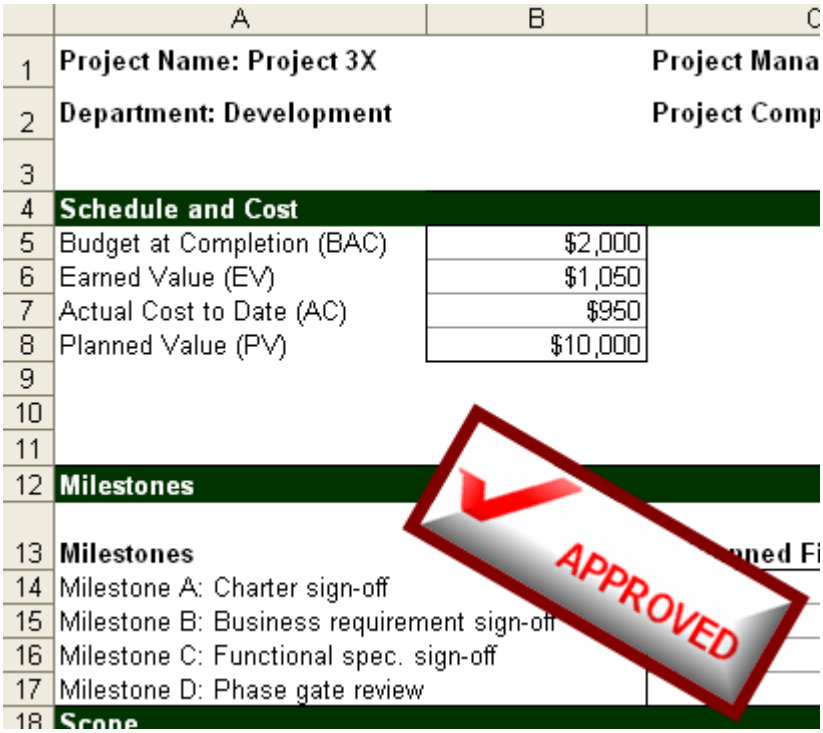
```
wb.Save("C:\Project\WorkBook1.xls ")
System.Diagnostics.Process.Start("C:\Project\WorkBook1.xls")
```

C#

C#

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

在这个例子中创建的图形对象会被添加到工作表的ShapeCollection集合中，而第一个单元格的值并不会改变。然后我们指定图形对象的高度，宽度以及水平和竖直位置。



添加批注到单元格

完成以下步骤添加批注到单元格：

1. 在工具箱上双击C1XLBook 组件，将会添加该组件到表单上。
2. 按照下面的代码给指定单元格添加文本：

Visual Basic

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

C#

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

3. 添加批注到工作表的XLCommentCollection集合，然后构造一个矩形区域来显示批注：

Visual Basic

```
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#

```
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. 保存文件，然后使用 Excel 打开：

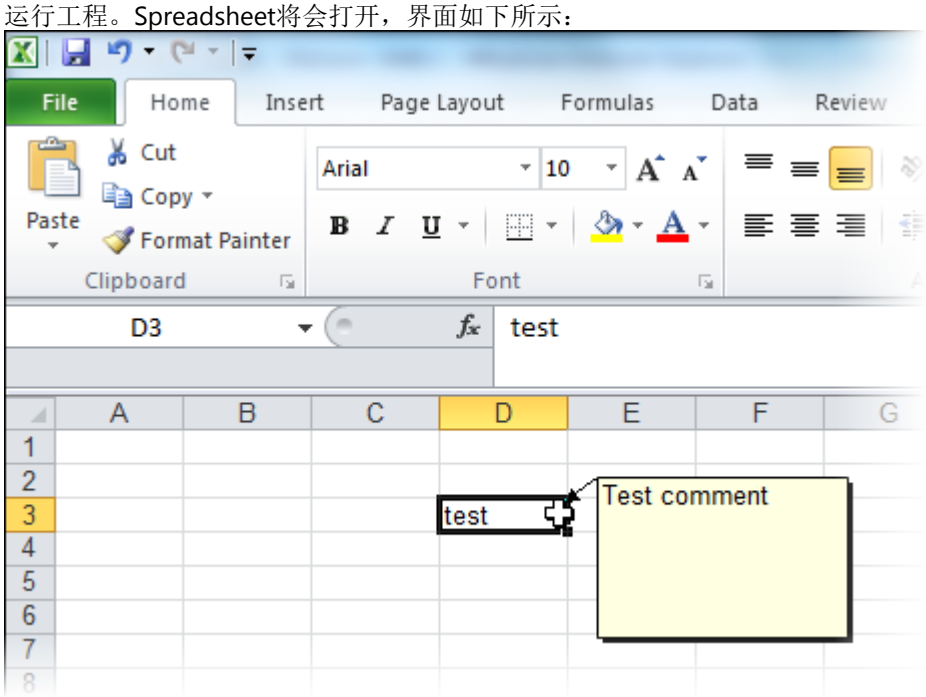
Visual Basic

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

C#

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

5. 运行工程。Spreadsheet将会打开，界面如下所示：



给表添加分页符

在OpenXML(.xlsx)格式的文件使用PageBreak和PageBreak属性可以很容易地给行和列添加分页符。

1. 双击工具箱中的C1XLBook组件把它添加到窗体中。
2. 用下面的代码添加一些文本值和分页符：

Visual Basic

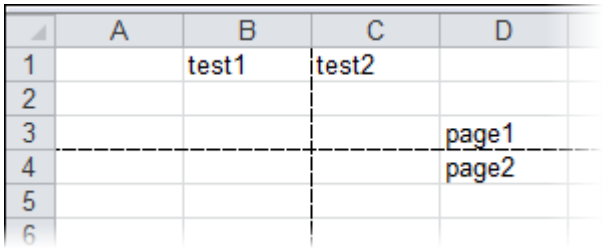
```
Visual Basic
C1XLBook1.Sheets(0)(2, 3).Value = "page1"
C1XLBook1.Sheets(0).Rows(2).PageBreak = True
C1XLBook1.Sheets(0)(0, 1).Value = "test1"
C1XLBook1.Sheets(0)(0, 2).Value = "test2"
C1XLBook1.Sheets(0).Columns(1).PageBreak = True
C1XLBook1.Sheets(0)(3, 3).Value = "page2"
' Save and open the .xlsx file
```

```
C1XLBook1.Save("c:\Save.xlsx")
System.Diagnostics.Process.Start("c:\Save.xlsx")
```

C#

```
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";
// Save and open the .xlsx file
c1XLBook1.Save(@"c:\Save.xlsx");
System.Diagnostics.Process.Start(@"c:\Save.xlsx");
```

3. 运行程序来打开xlsx文件。
4. 在Excel中，选择PageLayout选项卡，并且选择在Gridlines下面的Print复选框。工作表就会显示成下图的样子：



设定工作簿的计算模式

CalculationMode 属性定义了工作簿中所有计算式的计算模式。枚举提供下面三个选项：Manual（你手动执行计算），Auto（计算自动完成），或者AutoNoTable（除了表以外，计算自动执行）。

设定计算模式，参照下面的步骤：

1. 双击C1XLBook组件将它添加到你的窗体中。
2. 使用下面的代码添加简单的公式：

Visual Basic

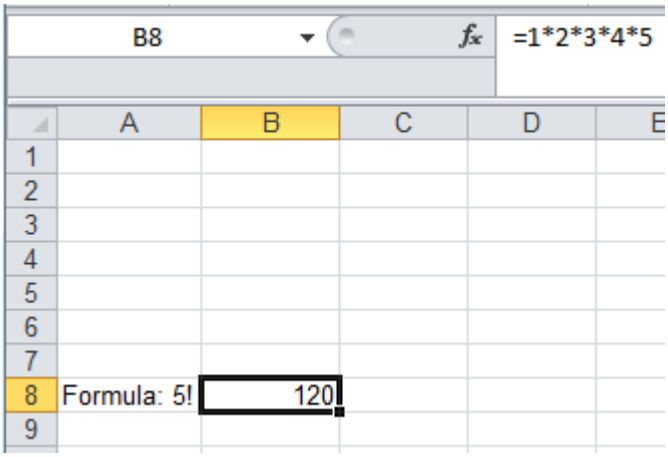
```
Visual Basic
Dim sheet As XLSheet = c1XLBook1.Sheets(0)
' simple formula
sheet(7, 0).Value = "Formula: 5!"
sheet(7, 1).Value = 122
sheet(7, 1).Formula = "1*2*3*4*5"
c1XLBook1.CalculationMode = CalculationMode.Auto
c1XLBook1.Save("c:\Save.xlsx")
System.Diagnostics.Process.Start("c:\Save.xlsx")
```

C#

```
C#
```

```
XLSheet sheet = c1XLBook1.Sheets[0];
// simple formula
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. 运行程序打开Excel文件。注意单元格(7,1)的值是120，也就是说单元格（7，1）的值不是122而是1*2*3*4*5的乘积，这是因为CalculationMode设定为了Auto。



导入导出OpenXML文件

ComponentOneExcelfor.NET现在可以读和写微软Excel2007OpenXml文件。 OpenXml文件是微软办公软件2007引入的一个开放的，基于标准格式的文件。OpenXml文件更容易被应用程序操作因为它是基于Xml的文件并且文件是公开的，而不是象BIFF8一样的专用的二进制格式。OpenXml文件包含大量的使用Zip压缩的XML文件。因为文件是被压缩过的，所以OpenXml文件通常比诸如.doc和.xls文件这样的传统文件更小。

.Excelfor.NET能够加载和保存OpenXml文件中的数据以及格式化的信息；但是，公式不能被加载和保存。公式是按BIFF格式复制的是不透明的，现在为止公式还不支持OpenXml格式。如果你加载了包含公式信息的OpenXml格式文件并且保存，包含的公式就会被移除。
这点和传统的.xls格式文件或者BIFF8格式文件不同，传统文件可以保留公式。

为了支持Openxml格式，C1XLBook的Load和Save方法带有FileFormat参数，这个参数用来定义加载后保存文件时文件的格式。

如果文件名没有定义，则Excelfor.NET从文件的扩展名来推断文件的格式：带有"XLSX"和"ZIP"扩展名的文件加载和保存时会被认为是OpenXml文件，而其他的文件默认就会加载和保存为BIFF8或者.xls格式的文件。

例如:

C#

C#

```
//load and save relying on file extension
c1Excel1.Load("somefile.xls"); // load biff 8 file
c1Excel1.Save("somefile.xlsx"); // save file as OpenXml
c1Excel1.Save("somefile.zip"); // save file as OpenXml
// load and save specifying the FileFormat
```

```
c1Excel1.Load("somefile.xls", FileFormat.Biff8);
c1Excel1.Save("somefile.xlsx", FileFormat.OpenXml);
```

当从工作流中加载和保存文件时你也能定义文件的格式。如果没有定义FileFormat，则Excelfor.NET默认使用BIFF8格式。

注意这里有一个小的行为改变。注意看下面的语句：

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

在Excelfor.NET以前的版本中，上面的语句会保存为BIFF8文件（带个错误的扩展名）。现在，会保存为OpenXml文件（带有正确的扩展名）。如果你的应用程序带有类似的代码，你必须升级的时候按下面的方式修改：

C#

```
C#
// deliberately save file with wrong extension
C1Excel1.Save("somefile.xlsx", FileFormat.Biff8);
```

要导出工作簿到OpenXml文件，需要完成下面的步骤:

1. 加载一个已经存在的工作簿：

   Visual Basic

   ```
   Visual Basic
   Dim wb As New C1XLBook()
   wb.Load("C:\test.xlsx")
   ' or
   Dim wb As New C1XLBook()
   wb.Load("C:\test.xlsx", C1.C1Excel.FileFormat.OpenXml)
   ```

   C#

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

2. 导出工作簿到OpenXml文件

   Visual Basic

   ```
   Visual Basic
   Dim wb As New C1XLBook()
   ' Add some content
   Dim sheet As XLSheet = wb.Sheets(0)
   Dim i As Integer
   For i = 0 To 9
           sheet(i,0).Value = i + 1
   ```

```
Next i
' Export to OpenXml Format file
wb.Save("C:\test.xlsx")
' or
' Export to OpenXml Format file
wb.Save("C:\test.xlsx", C1.C1Excel.FileFormat.OpenXml)
```

C#

C#

```
C1XLBook wb = new C1XLBook();
// Add some content
XLSheet sheet = wb.Sheets[0];
for (int i = 0; i <= 9; i++)
        {
        sheet[i,0].Value = i + 1;
        }
// Export to OpenXml Format file
wb.Save(@"C:\test.xlsx");
// or
// Export to OpenXml Format file
wb.Save(@"C:\test.xlsx", C1.C1Excel.FileFormat.OpenXml);
```

创建小计

下面的代码举例说明了如何格式化工作簿中的单元格。

1. 双击工具盒中的C1XLBook组件将它添加到你的窗体上。
2. 选择View|Code并且在你的窗体上添加下面的语句：
   - ImportC1.C1Excel(VisualBasic)
   - usingC1.C1Excel;(C#)
3. 在Form_Load事件中添加下面的代码：

   Visual Basic

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
```

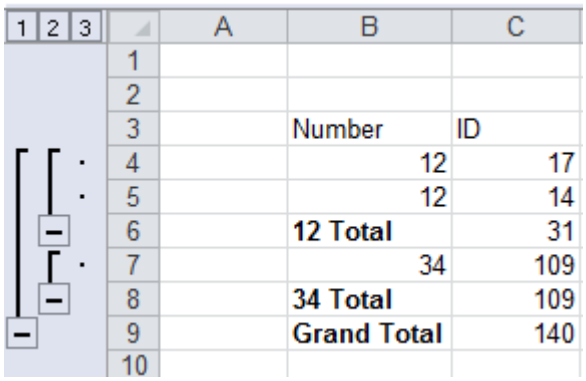
C#

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. 运行程序，就会打开一个电子表格，样子如下图：



	A	B	C
1			
2			
3		Number	ID
4		12	17
5		12	14
6		**12 Total**	31
7		34	109
8		**34 Total**	109
9		**Grand Total**	140
10			

SUBTOTAL 公式可以获得指定行的合计值。

保存和加载CSV文件

ComponentOneExcelfor.NET 支持保存和加载用逗号分隔值（CSV）文件。 CSV是一种存储表格数据的通常文件，表格数据包括数字和文本，它是一种纯文本的形式具有可读性。

下面的代码提供了如何保存和加载CSV文件的示例。

1. 双击工具盒中的C1XLBook组件将它添加到你的窗体上。
2. 选择View|Code并且在你的窗体上添加下面的语句：
   - ImportC1.C1Excel(VisualBasic)
   - usingC1.C1Excel;(C#)
3. 在Form_Load事件中添加下面的代码来创建一个带有10个值的表格并且以.csv的格式保存工作簿：

Visual Basic

Visual Basic

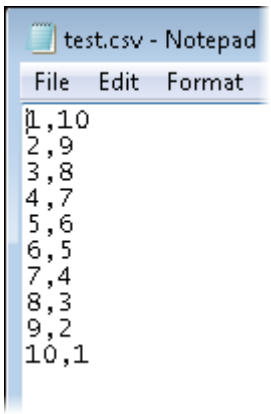
```
Private Sub Form1_Load(sender As Object, e As EventArgs)
   Dim sheet As XLSheet = c1XLBook1.Sheets(0)
    For i As Integer = 0 To 9
          sheet(i, 0).Value = i + 1
          sheet(i, 1).Value = 10 - i;
   Next
    sheet.SaveCsv("c:\test.csv")
   System.Diagnostics.Process.Start("C:\test.csv")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

C#

```
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. 按下F5运行程序，就能看到csv文件如图



5. 给test.csv文件添加一些新的值。为了将新的值保存到文件中你还需要再次保存文件。在Form1_Load事件中给LoadCsv和SaveCsv方法添加下面的代码就能做到了，代码如下

Visual Basic

```
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
```

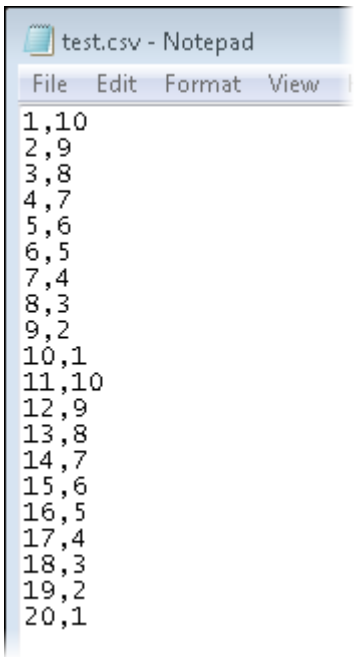
C#

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. 按下F5运行程序，就能看到csv文件如图



Excel for .NET 示例

请知悉ComponentOne软件工具带有不同的示例程序和演示程序，这些程序可以被其他包含有ComponentOne工作组的开发工具所使用。

示例可以从ComponentOne示例Explorer访问到。要在桌面上看这些示例，点击Start按钮然后点击

All Programs | ComponentOne | Studio for WinForms (or Studio for ASP.NET) | 示例s | Excel 示例s。

点击下面的链接中任意一个就能看到一系列Excelfor.NET示例：

Visual Basic

示例	描述
AutoSizeColumns	这个例子循环所有的单元格测量它们的内容，然后根据最宽入力内容来设置列宽。这个例子考虑到了单元格的内容，格式和字体.但是没有考虑内容的换行和合并。这个例子用了C1XLBook组件。
CellBorders	这个例子允许你给任何的范围设定边框。你可以控制边框的宽度，样式，颜色等等。当你完成了设定边框，点击工具栏上的最后一个按钮来把包含自定义边框的表格导出到Excel。这个例子使用了C1FlexGrid,C1XLBook和微软的ImageList组件。
CombineSheets	这个例子扫描文件夹里的所有.xls文件，克隆每个文件的第一个表，用文件名将表的名字重命名，并且将克隆的表添加到主工作簿中。然后将组合好的工作簿保存到文件中并打开文件。这个例子使用了C1XLBook组件。
ExcelPictures	这个例子创建了带有几个表的工作簿："Images"表显示了随机收集的图片，“Type”表显示了不同类型的图片，“Border”表显示了带有不同边框样式的图片，“Alignment”表显示了如何在单元格中对齐和按比例缩放图片，“Properties”表显示了其他图片属性的效果例如亮度，灰度等等。这个例子使用了C1XLBook控件。
FlexGridExcel	这个例子演示了如何加载和保存XLS文件。这个例子使用了C1FlexGrid和C1XLBook组件。
HelloWorld	这个例子实际上做得比保存"HelloWorld"文件多得多。它说明了如何创建工作簿，给工作表分配名字，创建样式并将样式分配给单元格，给单元格分配内容，并且保存工作簿..这个例子使用了C1XLBook控件。
PrintSettings	这个例子有一个Load按钮，点击打开一个已经存在的XLS文件并且显示出第一个表的打印设定。你可以改变打印设定，然后点击Save按钮保存一个新的文件（路径是c:\temp文件夹）。最后将新的文件显示在Excel中。这个例子使用了C1XLBook组件。

C#

示例	描述
AutoSizeColumns	这个例子循环所有的单元格测量它们的内容，然后根据最宽入力内容来设置列宽。这个例子考虑到了单元格的内容，格式和字体.但是没有考虑内容的换行和合并。这个例子用了C1XLBook组件。
CellBorders	这个例子允许你给任何的范围设定边框。你可以控制边框的宽度，样式，颜色等等。当你完成了设定边框，点击工具栏上的最后一个按钮来把包含自定义边框的表格导出到Excel。这个例子使用了C1FlexGrid,C1XLBook和微软的ImageList组件。
ColorPalette	这个例子表示出了Excel的调色板，包括颜色和颜色的RGB值。例子使用了C1XLBook 和C1FlexGrid 组件。
CombineSheets	这个例子扫描文件夹里的所有.xls文件，克隆每个文件的第一个表，用文件名将表的名字重命名，并且将克隆的表添加到主工作簿中。然后将组合好的工作簿保存到文件中并打开文件。

	这个例子使用了C1XLBook组件。
FlexGrid	这个例子演示了如何加载和保存XLS文件。这个例子使用了C1FlexGrid和C1XLBook组件。
HelloWorld	这个例子实际上做得比保存"HelloWorld"文件多得多。它说明了如何创建工作簿，给工作表分配名字，创建样式并将样式分配给单元格，给单元格分配内容，并且保存工作簿..这个例子使用了C1XLBook控件。
PrintSettings	这个例子有一个Load按钮，点击打开一个已经存在的XLS文件并且显示出第一个表的打印设定。你可以改变打印设定，然后点击Save按钮保存一个新的文件（路径是c:\temp文件夹）。最后将新的文件显示在Excel中。这个例子使用了C1XLBook组件。