# 可以用c#来操作office文件如excel和word文档等待,前提是你的vs必须安装office sharepoint编程组件支持,如果没有安装可以点击vs的工具->获取工具和功能,此时会打开vs安装程序,找到office sharepoint点击安装,需要退出正在运行的vs

Visual C# 2010 引入了可以简化对 Office API 对象的访问的新功能。这些新功能包括命名实参和可选实参、名为 **dynamic** 的新类型，以及在 COM 方法中将实参传递为引用形参（就像它们是值形参）的功能。

在本主题中，你将利用这些新功能来编写创建并显示 Microsoft Office Excel 工作表的代码。然后，你将编写添加包含链接到 Excel 工作表的图标的 Office Word 文档的代码。

若要完成本演练，你的计算机上必须安装 Microsoft Office Excel 2007 和 Microsoft Office Word 2007 或更高版本。

如果你使用的操作系统早于 Windows Vista，请确保安装 .NET Framework 2.0。

| **https://wizardforcel.gitbooks.io/msdn-csharp/content/img/note.jpg 注意** |
| --- |
| 以下说明中的某些 Visual Studio 用户界面元素在计算机上出现的名称或位置可能会不同。 这些元素取决于你所使用的 Visual Studio 版本和你所使用的设置。 有关详细信息，请参阅[个性化 Visual Studio IDE](https://msdn.microsoft.com/zh-CN/library/mt269425.aspx)。 |

**创建新的控制台应用程序**

1. 启动 Visual Studio。
2. 在**“文件”**菜单上，指向**“新建”**，然后单击**“项目”**。 出现 **新建项目** 对话框。
3. 在“已安装的模板”窗格中，展开“Visual C#”，然后单击“Windows”。
4. 查看“新建项目”对话框的顶部，确保“.NET Framework 4”（或更高版本）选为目标框架。
5. 在**“模板”**窗格中，单击**“控制台应用程序”**。
6. 在“名称”字段中键入项目的名称。
7. 单击“确定”。

新项目将出现在“解决方案资源管理器”中。

**添加引用**

1. 在“解决方案资源管理器”中，右键单击你的项目名称，然后单击“添加引用”。将显示**“添加引用”**对话框。
2. 在“程序集”页上，在“组件名称”列表中选择“Microsoft.Office.Interop.Word”，然后按住 Ctrl 键并选择“Microsoft.Office.Interop.Excel”。如果未看到程序集，你可能需要确保安装并显示它们（参阅[如何：安装 Office 主互操作程序集](https://msdn.microsoft.com/zh-CN/library/kh3965hw.aspx)）
3. 单击“确定”。

**添加必要的 using 指令**

1. 在“解决方案资源管理器”中，右键单击“Program.cs”文件，然后单击“查看代码”。
2. 将以下 **using** 指令添加到代码文件的顶部。
3. using Excel = Microsoft.Office.Interop.Excel;
4. using Word = Microsoft.Office.Interop.Word;

**创建银行帐户列表**

1. 将以下类定义粘贴到“Program.cs”中的 Program 类下。
2. public class Account
3. {
4. public int ID { get; set; }
5. public double Balance { get; set; }
6. }
7. 将以下代码添加到 Main 方法，以创建包含两个帐户的 bankAccounts 列表。
8. // Create a list of accounts.
9. var bankAccounts = new List&lt;Account&gt; {
10. new Account {
11. ID = 345678,
12. Balance = 541.27
13. },
14. new Account {
15. ID = 1230221,
16. Balance = -127.44
17. }
18. };

**声明将帐户信息导出到 Excel 的方法**

1. 将以下方法添加到 Program 类以设置 Excel 工作表。

方法 [Add](http://go.microsoft.com/fwlink/?LinkId=210910) 有一个可选参数，用于指定特定的模板。如果希望使用形参的默认值，你可以借助可选形参（Visual C# 2010 中新增）忽略该形参的实参。由于以下代码中未发送任何参数，**Add** 将使用默认模板并创建新的工作簿。C# 早期版本中的等效语句要求占位符参数：ExcelApp.Workbooks.Add(Type.Missing).

static void DisplayInExcel(IEnumerable&lt;Account&gt; accounts)

{

var excelApp = new Excel.Application();

// Make the object visible.

excelApp.Visible = true;

// Create a new, empty workbook and add it to the collection returned

// by property Workbooks. The new workbook becomes the active workbook.

// Add has an optional parameter for specifying a praticular template.

// Because no argument is sent in this example, Add creates a new workbook.

excelApp.Workbooks.Add();

// This example uses a single workSheet. The explicit type casting is

// removed in a later procedure.

Excel.\_Worksheet workSheet = (Excel.Worksheet)excelApp.ActiveSheet;

}

1. 在 DisplayInExcel 的末尾添加以下代码。代码将值插入工作表第一行的前两列。
2. // Establish column headings in cells A1 and B1.
3. workSheet.Cells[1, "A"] = "ID Number";
4. workSheet.Cells[1, "B"] = "Current Balance";
5. 在 DisplayInExcel 的末尾添加以下代码。 **foreach** 循环将帐户列表中的信息放入工作表连续行的前两列。
6. var row = 1;
7. foreach (var acct in accounts)
8. {
9. row++;
10. workSheet.Cells[row, "A"] = acct.ID;
11. workSheet.Cells[row, "B"] = acct.Balance;
12. }
13. 在 DisplayInExcel 的末尾添加以下代码以将列宽调整为适合内容。
14. workSheet.Columns[1].AutoFit();
15. workSheet.Columns[2].AutoFit();

早期版本的 C# 要求显式强制转换这些操作，因为 ExcelApp.Columns[1] 返回 **Object**，**AutoFit** 为 Excel [Range](http://go.microsoft.com/fwlink/?LinkId=210911) 方法。以下各行显示强制转换。

((Excel.Range)workSheet.Columns[1]).AutoFit();

((Excel.Range)workSheet.Columns[2]).AutoFit();

如果程序集由 [/link](https://msdn.microsoft.com/zh-CN/library/dd264728.aspx) 编译器选项引用或者如果 Excel 的“嵌入互操作类型”属性设置为 true，则 Visual C# 2010 及更高版本会自动将返回的 **Object** 转换为 **dynamic**。True 是此属性的默认值。

**运行项目**

1. 在 Main 的末尾添加以下行。
2. // Display the list in an Excel spreadsheet.
3. DisplayInExcel(bankAccounts);
4. 按 Ctrl+F5。

出现包含两个帐户数据的 Excel 工作表。

**添加 Word 文档**

1. 若要说明 Visual C# 2010 以及更高版本在其他哪些方面增强了 Office 编程，可以使用以下代码打开 Word 应用程序并创建链接到 Excel 工作表的图标。

将方法 CreateIconInWordDoc（在此步骤后面提供）粘贴到 Program 类中。 CreateIconInWordDoc 利用命名参数和可选参数来降低对 [Add](http://go.microsoft.com/fwlink/?LinkId=210937) 和 [PasteSpecial](http://go.microsoft.com/fwlink/?LinkId=147099) 的方法调用的复杂度。这些调用合并了其他两项新功能，这两项新功能在简化对具有引用参数的 COM 方法的调用的 Visual C# 2010 中引入。首先，你可以将实参发送到引用形参，就像它们是值形参一样。即，你可以直接发送值，而无需为每个引用参数创建变量。编译器会生成临时变量以保存参数值，并将在你从调用返回时丢弃变量。其次，你可以忽略参数列表中的 **ref** 关键字。

**Add** 方法有四个引用参数，所有引用参数都是可选的。在 Visual C# 2010 或更高版本中，如果希望使用其默认值，可以忽略任何或所有形参的实参。在 Visual C# 2008 以及更早版本中，由于形参是引用形参，因此必须为每个形参提供实参且实参必须是变量。

**PasteSpecial** 方法可插入剪贴板的内容。该方法有七个引用参数，所有引用参数都是可选的。以下代码为其中两个形参指定实参：Link 用于创建指向剪贴板内容源的连接，DisplayAsIcon 用于将链接显示为图标。在 Visual C# 2010 中，你可以对其中两个形参使用命名实参而忽略其他形参。尽管这些是引用形参，你也不必使用 **ref** 关键字，或者创建变量以实参形式发送。你可以直接发送值。在 Visual C# 2008 以及早期版本中，你必须为每个引用形参发送变量实参。

static void CreateIconInWordDoc()

{

var wordApp = new Word.Application();

wordApp.Visible = true;

// The Add method has four reference parameters, all of which are

// optional. Visual C# 2010 allows you to omit arguments for them if

// the default values are what you want.

wordApp.Documents.Add();

// PasteSpecial has seven reference parameters, all of which are

// optional. This example uses named arguments to specify values

// for two of the parameters. Although these are reference

// parameters, you do not need to use the ref keyword, or to create

// variables to send in as arguments. You can send the values directly.

wordApp.Selection.PasteSpecial( Link: true, DisplayAsIcon: true);

}

在 Visual C# 2008 或早期版本的语言中，需要以下更复杂的代码。

static void CreateIconInWordDoc2008()

{

var wordApp = new Word.Application();

wordApp.Visible = true;

// The Add method has four parameters, all of which are optional.

// In Visual C# 2008 and earlier versions, an argument has to be sent

// for every parameter. Because the parameters are reference

// parameters of type object, you have to create an object variable

// for the arguments that represents 'no value'.

object useDefaultValue = Type.Missing;

wordApp.Documents.Add(ref useDefaultValue, ref useDefaultValue,

ref useDefaultValue, ref useDefaultValue);

// PasteSpecial has seven reference parameters, all of which are

// optional. In this example, only two of the parameters require

// specified values, but in Visual C# 2008 an argument must be sent

// for each parameter. Because the parameters are reference parameters,

// you have to contruct variables for the arguments.

object link = true;

object displayAsIcon = true;

wordApp.Selection.PasteSpecial( ref useDefaultValue,

ref link,

ref useDefaultValue,

ref displayAsIcon,

ref useDefaultValue,

ref useDefaultValue,

ref useDefaultValue);

}

1. 在 Main 的末尾添加以下语句。
2. // Create a Word document that contains an icon that links to
3. // the spreadsheet.
4. CreateIconInWordDoc();
5. 在 DisplayInExcel 的末尾添加以下语句。 **Copy** 方法可将工作表添加到剪贴板。
6. // Put the spreadsheet contents on the clipboard. The Copy method has one
7. // optional parameter for specifying a destination. Because no argument
8. // is sent, the destination is the Clipboard.
9. workSheet.Range["A1:B3"].Copy();
10. 按 Ctrl+F5。

将出现包含图标的 Word 文档。双击该图标以将工作表置于前台。

**设置嵌入互操作类型属性**

1. 当调用运行时不需要主互操作程序集 (PIA) 的 COM 类型时，可能实现其他增强。删除 PIA 的依赖项可实现版本独立性并且更易于部署。有关不使用 PIA 编程的优势的详细信息，请参阅[演练：嵌入托管程序集中的类型（C# 和 Visual Basic）](https://msdn.microsoft.com/zh-CN/library/dd409610.aspx)。

此外，由于可以通过使用类型 **dynamic**（而非 **Object**）表示 COM 方法必需并返回的类型，因此更易于编程。具有类型 **dynamic** 的变量在运行时以前均不会计算，从而消除了显式强制转换的需要。有关详细信息，请参阅[使用类型 dynamic（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/dd264736.aspx)。

在 Visual C# 2010 中，默认行为是嵌入类型信息，而不是使用 PIA。由于该默认行为，因此不需要显式强制转换，之前的几个示例也得到简化。例如，DisplayInExcel 中 worksheet 的声明会写为 Excel.\_Worksheet workSheet = excelApp.ActiveSheet 而非 Excel.\_Worksheet workSheet = (Excel.Worksheet)excelApp.ActiveSheet。在相同方法中对 **AutoFit** 的调用还将要求在不进行默认行为的情况下显式强制转换，因为 ExcelApp.Columns[1] 返回 **Object**，并且 **AutoFit** 为 Excel 方法。以下代码显示强制转换。

((Excel.Range)workSheet.Columns[1]).AutoFit();

((Excel.Range)workSheet.Columns[2]).AutoFit();

1. 若要更改默认行为并使用 PIA 代替嵌入类型信息，请展开“解决方案资源管理器”中的“引用”节点，然后选择“Microsoft.Office.Interop.Excel”或“Microsoft.Office.Interop.Word”。
2. 如果看不到“属性”窗口，请按“F4”。
3. 在属性列表中找到“嵌入互操作类型”，将其值更改为“False”。同样地，你还可以通过在命令提示符下使用 [/reference](https://msdn.microsoft.com/zh-CN/library/yabyz3h4.aspx) 编译器选项代替 [/link](https://msdn.microsoft.com/zh-CN/library/dd264728.aspx) 进行编译。

**将其他格式添加到表格**

1. 将在 DisplayInExcel 中对 **AutoFit** 的两个调用替换为以下语句。
2. // Call to AutoFormat in Visual C# 2010.
3. workSheet.Range["A1", "B3"].AutoFormat(
4. Excel.XlRangeAutoFormat.xlRangeAutoFormatClassic2);

[AutoFormat](http://go.microsoft.com/fwlink/?LinkId=210948) 方法有七个值参数，每个值参数都是可选的。使用命名参数和可选参数，你可以为这些参数中的所有或部分提供参数，也可以不为它们中的任何一个提供。在上一条语句中，仅为其中一个形参 Format 提供实参。由于 Format 是参数列表中的第一个参数，因此无需提供参数名称。但是，如果包含参数名称，语句则可能更易于理解，如以下代码所示。

// Call to AutoFormat in Visual C# 2010.

workSheet.Range["A1", "B3"].AutoFormat(Format:

Excel.XlRangeAutoFormat.xlRangeAutoFormatClassic2);

1. 按 Ctrl+F5 查看结果。其他格式在 [XlRangeAutoFormat](http://go.microsoft.com/fwlink/?LinkId=210967) 枚举中列出。
2. 将步骤 1 中的语句与以下代码比较（以下代码显示 Visual C# 2008 或早期版本中要求的参数）。
3. // The AutoFormat method has seven optional value parameters. The
4. // following call specifies a value for the first parameter, and uses
5. // the default values for the other six.
6. // Call to AutoFormat in Visual C# 2008\. This code is not part of the
7. // current solution.
8. excelApp.get\_Range("A1", "B4").AutoFormat(Excel.XlRangeAutoFormat.xlRangeAutoFormatTable3,
9. Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing,
10. Type.Missing);

以下代码显示完整示例。

using System;

using System.Collections.Generic;

using System.Linq;

using Excel = Microsoft.Office.Interop.Excel;

using Word = Microsoft.Office.Interop.Word;

namespace OfficeProgramminWalkthruComplete

{

class Walkthrough

{

static void Main(string[] args)

{

// Create a list of accounts.

var bankAccounts = new List<Account>

{

new Account {

ID = 345678,

Balance = 541.27

},

new Account {

ID = 1230221,

Balance = -127.44

}

};

// Display the list in an Excel spreadsheet.

DisplayInExcel(bankAccounts);

// Create a Word document that contains an icon that links to

// the spreadsheet.

CreateIconInWordDoc();

}

static void DisplayInExcel(IEnumerable<Account> accounts)

{

var excelApp = new Excel.Application();

// Make the object visible.

excelApp.Visible = true;

// Create a new, empty workbook and add it to the collection returned

// by property Workbooks. The new workbook becomes the active workbook.

// Add has an optional parameter for specifying a praticular template.

// Because no argument is sent in this example, Add creates a new workbook.

excelApp.Workbooks.Add();

// This example uses a single workSheet.

Excel.\_Worksheet workSheet = excelApp.ActiveSheet;

// Earlier versions of C# require explicit casting.

//Excel.\_Worksheet workSheet = (Excel.Worksheet)excelApp.ActiveSheet;

// Establish column headings in cells A1 and B1.

workSheet.Cells[1, "A"] = "ID Number";

workSheet.Cells[1, "B"] = "Current Balance";

var row = 1;

foreach (var acct in accounts)

{

row++;

workSheet.Cells[row, "A"] = acct.ID;

workSheet.Cells[row, "B"] = acct.Balance;

}

workSheet.Columns[1].AutoFit();

workSheet.Columns[2].AutoFit();

// Call to AutoFormat in Visual C# 2010\. This statement replaces the

// two calls to AutoFit.

workSheet.Range["A1", "B3"].AutoFormat(

Excel.XlRangeAutoFormat.xlRangeAutoFormatClassic2);

// Put the spreadsheet contents on the clipboard. The Copy method has one

// optional parameter for specifying a destination. Because no argument

// is sent, the destination is the Clipboard.

workSheet.Range["A1:B3"].Copy();

}

static void CreateIconInWordDoc()

{

var wordApp = new Word.Application();

wordApp.Visible = true;

// The Add method has four reference parameters, all of which are

// optional. Visual C# 2010 allows you to omit arguments for them if

// the default values are what you want.

wordApp.Documents.Add();

// PasteSpecial has seven reference parameters, all of which are

// optional. This example uses named arguments to specify values

// for two of the parameters. Although these are reference

// parameters, you do not need to use the ref keyword, or to create

// variables to send in as arguments. You can send the values directly.

wordApp.Selection.PasteSpecial(Link: true, DisplayAsIcon: true);

}

}

public class Account

{

public int ID { get; set; }

public double Balance { get; set; }

}

}

# 这里有一个比较完整的c#写word文档的实例

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Runtime.InteropServices;  using System.Text;  using MSWord = Microsoft.Office.Interop.Word;  using System.IO;  using System.Reflection;  namespace Console\_WordSkill\_All  {  class Program  {  static void CreateWordDoc()  {  object path; //文件路径变量  string strContent; //文本内容变量  MSWord.Application wordApp; //Word应用程序变量  MSWord.Document wordDoc; //Word文档变量  path = Environment.CurrentDirectory + "\\TestWord.doc";  wordApp = new MSWord.Application(); //初始化  wordApp.Visible = true;//使文档可见  //如果已存在，则删除  if (File.Exists((string)path))  {  File.Delete((string)path);  }  Object Nothing = Missing.Value;  wordDoc = wordApp.Documents.Add(ref Nothing, ref Nothing, ref Nothing, ref Nothing);  #region 页面设置、页眉图片和文字设置，最后跳出页眉设置  //页面设置  wordDoc.PageSetup.PaperSize = MSWord.WdPaperSize.wdPaperA4;//设置纸张样式为A4纸  wordDoc.PageSetup.Orientation = MSWord.WdOrientation.wdOrientPortrait;//排列方式为垂直方向  wordDoc.PageSetup.TopMargin = 57.0f;  wordDoc.PageSetup.BottomMargin = 57.0f;  wordDoc.PageSetup.LeftMargin = 57.0f;  wordDoc.PageSetup.RightMargin = 57.0f;  wordDoc.PageSetup.HeaderDistance = 30.0f;//页眉位置  //设置页眉  wordApp.ActiveWindow.View.Type = MSWord.WdViewType.wdNormalView;//普通视图（即页面视图）样式  wordApp.ActiveWindow.View.SeekView = MSWord.WdSeekView.wdSeekPrimaryHeader;//进入页眉设置，其中页眉边距在页面设置中已完成  wordApp.Selection.ParagraphFormat.Alignment = MSWord.WdParagraphAlignment.wdAlignParagraphRight;//页眉中的文字右对齐  //插入页眉图片(测试结果图片未插入成功)  wordApp.Selection.ParagraphFormat.Alignment = MSWord.WdParagraphAlignment.wdAlignParagraphCenter;  //string headerfile = "4.jpg";  string headerfile = Environment.CurrentDirectory + "\\4.jpg";  MSWord.InlineShape shape1 = wordApp.ActiveWindow.ActivePane.Selection.InlineShapes.AddPicture(headerfile, ref Nothing, ref Nothing, ref Nothing);  shape1.Height = 5;//强行设置貌似无效，图片没有按设置的缩放——图片的比例并没有改变。  shape1.Width = 20;  wordApp.ActiveWindow.ActivePane.Selection.InsertAfter(" 文档页眉");//在页眉的图片后面追加几个字  //去掉页眉的横线  wordApp.ActiveWindow.ActivePane.Selection.ParagraphFormat.Borders[MSWord.WdBorderType.wdBorderBottom].LineStyle = MSWord.WdLineStyle.wdLineStyleNone;  wordApp.ActiveWindow.ActivePane.Selection.Borders[MSWord.WdBorderType.wdBorderBottom].Visible = false;  wordApp.ActiveWindow.ActivePane.View.SeekView = MSWord.WdSeekView.wdSeekMainDocument;//退出页眉设置  #endregion  #region 页码设置并添加页码  //为当前页添加页码  MSWord.PageNumbers pns = wordApp.Selection.Sections[1].Headers[MSWord.WdHeaderFooterIndex.wdHeaderFooterEvenPages].PageNumbers;//获取当前页的号码  pns.NumberStyle = MSWord.WdPageNumberStyle.wdPageNumberStyleNumberInDash;//设置页码的风格，是Dash形还是圆形的  pns.HeadingLevelForChapter = 0;  pns.IncludeChapterNumber = false;  pns.RestartNumberingAtSection = false;  pns.StartingNumber = 0; //开始页页码？  object pagenmbetal = MSWord.WdPageNumberAlignment.wdAlignPageNumberCenter;//将号码设置在中间  object first = true;  wordApp.Selection.Sections[1].Footers[MSWord.WdHeaderFooterIndex.wdHeaderFooterEvenPages].PageNumbers.Add(ref pagenmbetal, ref first);  #endregion  #region 行间距与缩进、文本字体、字号、加粗、斜体、颜色、下划线、下划线颜色设置  wordApp.Selection.ParagraphFormat.LineSpacing = 16f;//设置文档的行间距  wordApp.Selection.ParagraphFormat.FirstLineIndent = 30;//首行缩进的长度  //写入普通文本  strContent = "我是普通文本\n";  wordDoc.Paragraphs.Last.Range.Text = strContent;  wordDoc.Paragraphs.Last.Range.Text = "我再加一行试试，这里不加'\\n'";  //直接添加段，不是覆盖( += )  wordDoc.Paragraphs.Last.Range.Text += "不会覆盖的,";  //添加在此段的文字后面，不是新段落  wordDoc.Paragraphs.Last.Range.InsertAfter("这是后面的内容\n");  //将文档的前4个字替换成"哥是替换文字"，并将其颜色设为红色  object start = 0;  object end = 4;  MSWord.Range rang = wordDoc.Range(ref start, ref end);  rang.Font.Color = MSWord.WdColor.wdColorRed;  rang.Text = "哥是替换文字";  wordDoc.Range(ref start, ref end);  //写入黑体文本  object unite = MSWord.WdUnits.wdStory;  wordApp.Selection.EndKey(ref unite, ref Nothing);//将光标移到文本末尾  wordApp.Selection.ParagraphFormat.FirstLineIndent = 0;//取消首行缩进的长度  strContent = "这是黑体文本\n";  wordDoc.Paragraphs.Last.Range.Font.Name = "黑体";  wordDoc.Paragraphs.Last.Range.Text = strContent;  //写入加粗文本  strContent = "这是粗体文本\n"; //  wordApp.Selection.EndKey(ref unite, ref Nothing);//这一句不加，有时候好像也不出问题，不过还是加了安全  wordDoc.Paragraphs.Last.Range.Font.Bold = 1;  wordDoc.Paragraphs.Last.Range.Text = strContent;  //写入15号字体文本  strContent = "我这个文本的字号是15号，而且是宋体\n";  wordApp.Selection.EndKey(ref unite, ref Nothing);  wordDoc.Paragraphs.Last.Range.Font.Size = 15;  wordDoc.Paragraphs.Last.Range.Font.Name = "宋体";  wordDoc.Paragraphs.Last.Range.Text = strContent;  //写入斜体文本  strContent = "我是斜体字文本\n";  wordApp.Selection.EndKey(ref unite, ref Nothing);  wordDoc.Paragraphs.Last.Range.Font.Italic = 1;  wordDoc.Paragraphs.Last.Range.Text = strContent;  //写入蓝色文本  strContent = "我是蓝色的文本\n";  wordApp.Selection.EndKey(ref unite, ref Nothing);  wordDoc.Paragraphs.Last.Range.Font.Color = MSWord.WdColor.wdColorBlue;  wordDoc.Paragraphs.Last.Range.Text = strContent;  //写入下划线文本  strContent = "我是下划线文本\n";  wordApp.Selection.EndKey(ref unite, ref Nothing);  wordDoc.Paragraphs.Last.Range.Font.Underline = MSWord.WdUnderline.wdUnderlineThick;  wordDoc.Paragraphs.Last.Range.Text = strContent;  //写入红色下画线文本  strContent = "我是点线下划线，并且下划线是红色的\n";  wordApp.Selection.EndKey(ref unite, ref Nothing);  wordDoc.Paragraphs.Last.Range.Font.Underline = MSWord.WdUnderline.wdUnderlineDottedHeavy;  wordDoc.Paragraphs.Last.Range.Font.UnderlineColor = MSWord.WdColor.wdColorRed;  wordDoc.Paragraphs.Last.Range.Text = strContent;  //取消下划线，并且将字号调整为12号  strContent = "我不要下划线了，并且设置字号为12号，黑色不要斜体\n";  wordApp.Selection.EndKey(ref unite, ref Nothing);  wordDoc.Paragraphs.Last.Range.Font.Size = 12;  wordDoc.Paragraphs.Last.Range.Font.Underline = MSWord.WdUnderline.wdUnderlineNone;  wordDoc.Paragraphs.Last.Range.Font.Color = MSWord.WdColor.wdColorBlack;  wordDoc.Paragraphs.Last.Range.Font.Italic = 0;  wordDoc.Paragraphs.Last.Range.Text = strContent;  #endregion  #region 插入图片、居中显示，设置图片的绝对尺寸和缩放尺寸，并给图片添加标题  wordApp.Selection.EndKey(ref unite, ref Nothing); //将光标移动到文档末尾  //图片文件的路径  string filename = Environment.CurrentDirectory + "\\6.png";  //要向Word文档中插入图片的位置  Object range = wordDoc.Paragraphs.Last.Range;  //定义该插入的图片是否为外部链接  Object linkToFile = false; //默认,这里貌似设置为bool类型更清晰一些  //定义要插入的图片是否随Word文档一起保存  Object saveWithDocument = true; //默认  //使用InlineShapes.AddPicture方法(【即“嵌入型”】)插入图片  wordDoc.InlineShapes.AddPicture(filename, ref linkToFile, ref saveWithDocument, ref range);  wordApp.Selection.ParagraphFormat.Alignment = MSWord.WdParagraphAlignment.wdAlignParagraphCenter;//居中显示图片  //设置图片宽高的绝对大小  //wordDoc.InlineShapes[1].Width = 200;  //wordDoc.InlineShapes[1].Height = 150;  //按比例缩放大小  wordDoc.InlineShapes[1].ScaleWidth = 20;//缩小到20% ？  wordDoc.InlineShapes[1].ScaleHeight = 20;  //在图下方居中添加图片标题  wordDoc.Content.InsertAfter("\n");//这一句与下一句的顺序不能颠倒，原因还没搞透  wordApp.Selection.EndKey(ref unite, ref Nothing);  wordApp.Selection.ParagraphFormat.Alignment = MSWord.WdParagraphAlignment.wdAlignParagraphCenter;  wordApp.Selection.Font.Size = 10;//字体大小  wordApp.Selection.TypeText("图1 测试图片\n");  #endregion  #region 添加表格、填充数据、设置表格行列宽高、合并单元格、添加表头斜线、给单元格添加图片  wordDoc.Content.InsertAfter("\n");//这一句与下一句的顺序不能颠倒，原因还没搞透  wordApp.Selection.EndKey(ref unite, ref Nothing); //将光标移动到文档末尾  wordApp.Selection.ParagraphFormat.Alignment = MSWord.WdParagraphAlignment.wdAlignParagraphLeft;  //object WdLine2 = MSWord.WdUnits.wdLine;//换一行;  //wordApp.Selection.MoveDown(ref WdLine2, 6, ref Nothing);//向下跨15行输入表格，这样表格就在文字下方了，不过这是非主流的方法  //设置表格的行数和列数  int tableRow = 6;  int tableColumn = 6;  //定义一个Word中的表格对象  MSWord.Table table = wordDoc.Tables.Add(wordApp.Selection.Range,  tableRow, tableColumn, ref Nothing, ref Nothing);  //默认创建的表格没有边框，这里修改其属性，使得创建的表格带有边框  table.Borders.Enable = 1;//这个值可以设置得很大，例如5、13等等  //表格的索引是从1开始的。  wordDoc.Tables[1].Cell(1, 1).Range.Text = "列\n行";  for (int i = 1; i < tableRow; i++)  {  for (int j = 1; j < tableColumn; j++)  {  if (i == 1)  {  table.Cell(i, j + 1).Range.Text = "Column " + j;//填充每列的标题  }  if (j == 1)  {  table.Cell(i + 1, j).Range.Text = "Row " + i; //填充每行的标题  }  table.Cell(i + 1, j + 1).Range.Text = i + "行 " + j + "列"; //填充表格的各个小格子  }  }  //添加行  table.Rows.Add(ref Nothing);  table.Rows[tableRow + 1].Height = 35;//设置新增加的这行表格的高度  //向新添加的行的单元格中添加图片  string FileName = Environment.CurrentDirectory + "\\6.png";//图片所在路径  object LinkToFile = false;  object SaveWithDocument = true;  object Anchor = table.Cell(tableRow + 1, tableColumn).Range;//选中要添加图片的单元格  wordDoc.Application.ActiveDocument.InlineShapes.AddPicture(FileName, ref LinkToFile, ref SaveWithDocument, ref Anchor);  //由于是本文档的第2张图，所以这里是InlineShapes[2]  wordDoc.Application.ActiveDocument.InlineShapes[2].Width = 50;//图片宽度  wordDoc.Application.ActiveDocument.InlineShapes[2].Height = 35;//图片高度  // 将图片设置为四周环绕型  MSWord.Shape s = wordDoc.Application.ActiveDocument.InlineShapes[2].ConvertToShape();  s.WrapFormat.Type = MSWord.WdWrapType.wdWrapSquare;  //设置table样式  table.Rows.HeightRule = MSWord.WdRowHeightRule.wdRowHeightAtLeast;//高度规则是：行高有最低值下限？  table.Rows.Height = wordApp.CentimetersToPoints(float.Parse("0.8"));//  table.Range.Font.Size = 10.5F;  table.Range.Font.Bold = 0;  table.Range.ParagraphFormat.Alignment = MSWord.WdParagraphAlignment.wdAlignParagraphCenter;//表格文本居中  table.Range.Cells.VerticalAlignment = MSWord.WdCellVerticalAlignment.wdCellAlignVerticalBottom;//文本垂直贴到底部  //设置table边框样式  table.Borders.OutsideLineStyle = MSWord.WdLineStyle.wdLineStyleDouble;//表格外框是双线  table.Borders.InsideLineStyle = MSWord.WdLineStyle.wdLineStyleSingle;//表格内框是单线  table.Rows[1].Range.Font.Bold = 1;//加粗  table.Rows[1].Range.Font.Size = 12F;  table.Cell(1, 1).Range.Font.Size = 10.5F;  wordApp.Selection.Cells.Height = 30;//所有单元格的高度  //除第一行外，其他行的行高都设置为20  for (int i = 2; i <= tableRow; i++)  {  table.Rows[i].Height = 20;  }  //将表格左上角的单元格里的文字（“行” 和 “列”）居右  table.Cell(1, 1).Range.ParagraphFormat.Alignment = MSWord.WdParagraphAlignment.wdAlignParagraphRight;  //将表格左上角的单元格里面下面的“列”字移到左边，相比上一行就是将ParagraphFormat改成了Paragraphs[2].Format  table.Cell(1, 1).Range.Paragraphs[2].Format.Alignment = MSWord.WdParagraphAlignment.wdAlignParagraphLeft;  table.Columns[1].Width = 50;//将第 1列宽度设置为50  //将其他列的宽度都设置为75  for (int i = 2; i <= tableColumn; i++)  {  table.Columns[i].Width = 75;  }  //添加表头斜线,并设置表头的样式  table.Cell(1, 1).Borders[MSWord.WdBorderType.wdBorderDiagonalDown].Visible = true;  table.Cell(1, 1).Borders[MSWord.WdBorderType.wdBorderDiagonalDown].Color = MSWord.WdColor.wdColorRed;  table.Cell(1, 1).Borders[MSWord.WdBorderType.wdBorderDiagonalDown].LineWidth = MSWord.WdLineWidth.wdLineWidth150pt;  //合并单元格  table.Cell(4, 4).Merge(table.Cell(4, 5));//横向合并  table.Cell(2, 3).Merge(table.Cell(4, 3));//纵向合并，合并(2,3),(3,3),(4,3)  #endregion  wordApp.Selection.EndKey(ref unite, ref Nothing); //将光标移动到文档末尾  wordDoc.Content.InsertAfter("\n");  wordDoc.Content.InsertAfter("就写这么多，算了吧！2016.09.27");  //WdSaveFormat为Word 2003文档的保存格式  object format = MSWord.WdSaveFormat.wdFormatDocument;// office 2007就是wdFormatDocumentDefault  //将wordDoc文档对象的内容保存为DOCX文档  wordDoc.SaveAs(ref path, ref format, ref Nothing, ref Nothing, ref Nothing, ref Nothing, ref Nothing, ref Nothing, ref Nothing, ref Nothing, ref Nothing, ref Nothing, ref Nothing, ref Nothing, ref Nothing, ref Nothing);  //关闭wordDoc文档对象  //看是不是要打印  //wordDoc.PrintOut();  wordDoc.Close(ref Nothing, ref Nothing, ref Nothing);  //关闭wordApp组件对象  wordApp.Quit(ref Nothing, ref Nothing, ref Nothing);  Console.WriteLine(path + " 创建完毕！");  Console.ReadKey();  //我还要打开这个文档玩玩  MSWord.Application app = new MSWord.Application();  MSWord.Document doc = null;  try  {  object unknow = Type.Missing;  app.Visible = true;  string str = Environment.CurrentDirectory + "\\TestWord.doc";  object file = str;  doc = app.Documents.Open(ref file,  ref unknow, ref unknow, ref unknow, ref unknow,  ref unknow, ref unknow, ref unknow, ref unknow,  ref unknow, ref unknow, ref unknow, ref unknow,  ref unknow, ref unknow, ref unknow);  string temp = doc.Paragraphs[1].Range.Text.Trim();  Console.WriteLine("你输出temp干嘛？");  }  catch (Exception ex)  {  Console.WriteLine(ex.Message);  }  wordDoc = doc;  wordDoc.Paragraphs.Last.Range.Text += "写这么多吧,祝你好运";  Console.ReadKey();  }  static void Main(string[] args)  {  CreateWordDoc();  }  }  } |

## 还有一个参考网站: <https://www.cnblogs.com/dfcq/p/17835865.html>

**请参阅**

[Type.Missing](https://msdn.microsoft.com/zh-CN/library/system.type.missing.aspx)

[dynamic（C# 参考）](https://msdn.microsoft.com/zh-CN/library/dd264741.aspx)

[使用类型 dynamic（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/dd264736.aspx)

[命名实参和可选实参（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/dd264739.aspx)

[如何：在 Office 编程中使用命名参数和可选参数（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/dd264738.aspx)