**演练：Office 编程（C# 和 Visual Basic）**

Visual Studio 2010 在 C# 和 Visual Basic 中引入了改进 Microsoft Office 编程的新功能。每种语言都增加了其他语言中已经存在的功能。

C# 中的新功能包括命名参数和可选参数、具有 **dynamic** 类型的返回值，以及在 COM 编程中忽略 **ref** 关键字和访问索引属性的功能。Visual Basic 中的新功能包括自动实现的属性、Lambda 表达式语句和集合初始值设定项。

两种语言都支持嵌入类型信息，从而允许在不向用户的计算机部署主互操作程序集 (PIA) 的情况下部署与 COM 组件交互的程序集。有关详细信息，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)。

本演练演示 Office 编程环境中的新功能，但这些新功能在常规编程中也极为有用。在本演练中，你将首先使用 Excel 外接应用程序来创建 Excel 工作簿。然后，你将创建包含工作簿链接的 Word 文档。最后，你将看到可以如何开启和关闭 PIA 依赖项。

**系统必备**

若要完成本演练，你的计算机上必须安装有 Microsoft Office Excel 2013（或者 2007 或更高版本）和 Microsoft Office Word 2013（或者 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)。 |

**设置 Excel 外接应用程序**

1. 启动 Visual Studio。
2. 在**“文件”**菜单上，指向**“新建”**，然后单击**“项目”**。
3. 在“安装的模板”窗格中，展开“Visual Basic”或“Visual C#”，再展开“Office”，然后单击“2013”（或“2010”或“2007”）。
4. 在“模板”窗格中，单击“Excel 2013 外接应用程序”（或“Excel 2010 外接应用程序”或“Excel 2007 外接应用程序”）。
5. 查看“模板”窗格的顶部，确保“.NET Framework 4”或更高版本出现在“目标框架”框中。
6. 如果需要，在“名称”框中键入项目的名称。
7. 单击“确定”。
8. 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”选项卡上，在“组件名称”列表中选择“Microsoft.Office.Interop.Excel”版本 15.0.0.0（针对 Excel 2010，选择版本 14.0.0.0；针对 Excel 2007，选择版本 12.0.0.0），然后按住 Ctrl 键并选择“Microsoft.Office.Interop.Word”版本 15.0.0.0（针对 Word 2010，选择版本 14.0.0.0；针对 Word 2007，选择版本 12.0.0.0）。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 Imports 语句或 using 指令**

1. 在“解决方案资源管理器”中，右键单击“ThisAddIn.vb”或“ThisAddIn.cs”文件，然后单击“查看代码”。
2. 将以下 **Imports** 语句 (Visual Basic) 或 **using** 指令 (C#) 添加到代码文件的顶部（如果不存在）。

[VB](https://msdn.microsoft.com/zh-CN/library/ee342218.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

using System.Collections.Generic;

using Excel = Microsoft.Office.Interop.Excel;

using Word = Microsoft.Office.Interop.Word;

**创建银行帐户列表**

1. 在“解决方案资源管理器”中，右键单击你的项目名称，单击“添加”，然后单击“类”。如果使用的是 Visual Basic，则将类命名为 Account.vb；如果使用的是 C#，则将类命名为 Account.cs。单击**“添加”**。
2. 将 Account 类的定义替换为以下代码。类定义使用*“自动实现的属性”*，在 Visual Studio 2010 中，是 Visual Basic 的新功能。有关详细信息，请参阅[自动实现的属性 (Visual Basic)](https://msdn.microsoft.com/zh-CN/library/dd293589.aspx)。

[VB](https://msdn.microsoft.com/zh-CN/library/ee342218.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

class Account

{

public int ID { get; set; }

public double Balance { get; set; }

}

1. 若要创建包含两个帐户的 bankAccounts 列表，请将以下代码添加到 ThisAddIn.vb 或 ThisAddIn.cs 中的 ThisAddIn*Startup 方法。列表声明使用*“集合初始值设定项”\_，在 Visual Studio 2010 中，是 Visual Basic 的新功能。有关详细信息，请参阅[集合初始值设定项 (Visual Basic)](https://msdn.microsoft.com/zh-CN/library/dd293617.aspx)。

[VB](https://msdn.microsoft.com/zh-CN/library/ee342218.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-3)

var bankAccounts = new List&lt;Account&gt;

{

new Account

{

ID = 345,

Balance = 541.27

},

new Account

{

ID = 123,

Balance = -127.44

}

};

**将数据导出到 Excel**

1. 在相同的文件中，将以下方法添加到 ThisAddIn 类。该方法设置 Excel 工作薄并将数据导出到工作簿。

[VB](https://msdn.microsoft.com/zh-CN/library/ee342218.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-4)

void DisplayInExcel(IEnumerable<Account> accounts,

Action<Account, Excel.Range> DisplayFunc)

{

var excelApp = this.Application;

// Add a new Excel workbook.

excelApp.Workbooks.Add();

excelApp.Visible = true;

excelApp.Range["A1"].Value = "ID";

excelApp.Range["B1"].Value = "Balance";

excelApp.Range["A2"].Select();

foreach (var ac in accounts)

{

DisplayFunc(ac, excelApp.ActiveCell);

excelApp.ActiveCell.Offset[1, 0].Select();

}

// Copy the results to the Clipboard.

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

}

此方法使用 C# 的两项新功能。Visual Basic 中已存在这两项功能。

* + 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。由于上一个示例中未发送任何参数，**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).

有关详细信息，请参阅[命名实参和可选实参（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/dd264739.aspx)。

* + [Range](http://go.microsoft.com/fwlink/?LinkId=210911) 对象的 **Range** 和 **Offset** 属性使用*“索引属性”*功能。此功能允许你通过以下典型 C# 语法从 COM 类型使用这些属性。索引属性还允许你使用 **Range** 对象的 **Value** 属性，因此不必使用 **Value2** 属性。 **Value** 属性已编入索引，但索引是可选的。在以下示例中，可选参数和索引属性配合使用。
  + // Visual C# 2010 provides indexed properties for COM programming.
  + excelApp.Range["A1"].Value = "ID";
  + excelApp.ActiveCell.Offset[1, 0].Select();

在早期版本的语言中，需要以下特殊语法。

// In Visual C# 2008, you cannot access the Range, Offset, and Value

// properties directly.

excelApp.get\_Range("A1").Value2 = "ID";

excelApp.ActiveCell.get\_Offset(1, 0).Select();

你不能创建自己的索引属性。该功能仅支持使用现有索引属性。

有关详细信息，请参阅[如何：在 COM 互操作编程中使用索引属性（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/ee310208.aspx)。

1. 在 DisplayInExcel 的末尾添加以下代码以将列宽调整为适合内容。

[VB](https://msdn.microsoft.com/zh-CN/library/ee342218.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-7)

excelApp.Columns[1].AutoFit();

excelApp.Columns[2].AutoFit();

这些添加展示 C# 2010 中的另一项新功能：处理从 COM 主机返回的 **Object** 值（如 Office），就像它们具有 [dynamic](https://msdn.microsoft.com/zh-CN/library/dd264741.aspx) 类型一样。当“嵌入互操作类型”设置为其默认值 **True** 时，或者由 [/link](https://msdn.microsoft.com/zh-CN/library/dd264728.aspx) 编译器选项引用程序集时，自动发生这种情况。键入 **dynamic** 允许后期绑定（Visual Basic 已提供该功能）并可避免 Visual C# 2008 和早期版本的语言中要求的显式强制转换。

例如，excelApp.Columns[1] 返回 **Object**，并且 **AutoFit** 是 Excel 的 [Range](http://go.microsoft.com/fwlink/?LinkId=210911) 方法。如果没有 **dynamic**，你必须将 excelApp.Columns[1] 返回的对象强制转换为 Range 的实例，然后才能调用 AutoFit 方法。

// Casting is required in Visual C# 2008.

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

// Casting is not required in Visual C# 2010.

excelApp.Columns[1].AutoFit();

有关嵌入互操作类型的详细信息，请参阅本主题后面部分的“查找 PIA 引用”和“还原 PIA 依赖项”程序。有关 **dynamic** 的详细信息，请参阅 [dynamic（C# 参考）](https://msdn.microsoft.com/zh-CN/library/dd264741.aspx) 或 [使用类型 dynamic（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/dd264736.aspx)。

**调用 DisplayInExcel**

1. 在 ThisAddIn\_StartUp 方法的末尾添加以下代码。对 DisplayInExcel 的调用包含两个参数。第一个参数是要处理的帐户列表的名称。第二个参数是定义如何处理数据的多行 lambda 表达式。每个帐户的 ID 和 balance 值都显示在相邻的单元格中，如果余额小于零，则相应的行显示为红色。多行 lambda 表达式是 Visual Basic 2010 中的新功能。有关详细信息，请参阅[Lambda 表达式 (Visual Basic)](https://msdn.microsoft.com/zh-CN/library/bb531253.aspx)。

[VB](https://msdn.microsoft.com/zh-CN/library/ee342218.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-9)

DisplayInExcel(bankAccounts, (account, cell) =>

// This multiline lambda expression sets custom processing rules

// for the bankAccounts.

{

cell.Value = account.ID;

cell.Offset[0, 1].Value = account.Balance;

if (account.Balance > 0)

{

cell.Interior.Color = 255;

cell.Offset[0, 1].Interior.Color = 255;

}

});

1. 若要运行程序，请按 F5。出现包含帐户数据的 Excel 工作表。

**添加 Word 文档**

1. 在 ThisAddIn\_StartUp 方法末尾添加以下代码，以创建包含指向 Excel 工作簿的链接的 Word 文档。

[VB](https://msdn.microsoft.com/zh-CN/library/ee342218.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-10)

var wordApp = new Word.Application();

wordApp.Visible = true;

wordApp.Documents.Add();

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

此代码展示 C# 中的几项新功能：在 COM 编程中忽略 **ref** 的功能、命名参数以及可选参数。Visual Basic 中已存在这些功能。 [PasteSpecial](http://go.microsoft.com/fwlink/?LinkId=147099) 方法有七个参数，每个参数都是可选引用参数。在 Visual C# 2010 之前，你必须为这七个形参定义用作实参的对象变量，即使你没有有意义的值发送。通过命名实参和可选实参，你可以指定希望按名称访问的形参并仅将实参发送到这些形参。在本示例中，发送实参以指示应创建指向剪贴板上工作簿的链接（形参 Link）并指示该链接应在 Word 文档中显示为图标（形参 DisplayAsIcon）。Visual C# 2010 还允许你忽略这些参数的 **ref** 关键字。将 Visual C# 2008 的以下代码段与 Visual C# 2010 中需要的单行进行比较：

// Call to PasteSpecial in Visual C# 2008.

object iconIndex = Type.Missing;

object link = true;

object placement = Type.Missing;

object displayAsIcon = true;

object dataType = Type.Missing;

object iconFileName = Type.Missing;

object iconLabel = Type.Missing;

wordApp.Selection.PasteSpecial(ref iconIndex,

ref link,

ref placement,

ref displayAsIcon,

ref dataType,

ref iconFileName,

ref iconLabel);

// Call to PasteSpecial in Visual C# 2010.

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

**运行应用程序**

1. 按 F5 运行该应用程序。Excel 启动并显示包含 bankAccounts 中两个帐户的信息的表。然后，出现包含指向 Excel 表的 Word 文档。

**清理已完成的项目**

1. 在 Visual Studio 中，单击“生成”菜单上的“清理解决方案”。否则，每次在计算机上打开 Excel 时都会运行外接应用程序。

**查找 PIA 引用**

1. 再次运行应用程序，但不单击“清理解决方案”。
2. 在“开始”菜单上，单击“所有程序”。接下来依次单击“Microsoft Visual Studio 2013”、“Visual Studio 工具”、“Visual Studio 命令提示符 (2013)”。
3. 在 Visual Studio 命令提示符 (2013) 窗口中键入 **ildasm**，然后按 Enter。此时将出现 IL DASM 窗口。
4. 在 IL DASM 窗口的“文件”菜单上，单击“打开”。双击“Visual Studio 2013”，然后双击“项目”。打开项目的文件夹，在 bin/Debug 文件夹中查找 *项目名称*.dll。双击 *项目名称*.dll。新窗口将显示项目的属性以及对其他模块和程序集的引用。注意，命名空间 **Microsoft.Office.Interop.Excel** 和 **Microsoft.Office.Interop.Word** 包含在程序集中。在 Visual Studio 2013 中，编译器默认将你需要的类型从引用的 PIA 导入程序集。

有关详细信息，请参阅[如何：查看程序集内容](https://msdn.microsoft.com/zh-CN/library/ceats605.aspx)。

1. 双击“清单”图标。此时将出现包含程序集列表的窗口，这些程序集包含项目所引用的项。 **Microsoft.Office.Interop.Excel** 和 **Microsoft.Office.Interop.Word** 未包含在列表中。由于项目需要的类型已导入程序集中，因此不需要引用 PIA。这使得部署变得更加容易。用户的计算机上不必存在 PIA，因为应用程序不需要部署特定版本的 PIA，应用程序可设计为与多个版本的 Office 配合使用，前提是所有版本中都存在必要的 API。

由于不再需要部署 PIA，你可以提前创建可与多个版本的 Office（包括之前的版本）配合使用的应用程序。但是，仅当你的代码不使用你当前所使用 Office 版本中不可用的任何 API 时，此情况才适用。特殊 API 在早期版本中是否可用并不始终明确，因此不建议使用早期版本的 Office。

1. 关闭清单窗口和程序集窗口。

| **https://wizardforcel.gitbooks.io/msdn-csharp/content/img/note.jpg 注意** |
| --- |
| 在 Office 2003 以前，Office 并不发布 PIA。因此，生成适用于 Office 2002 或早期版本的互操作程序集的唯一方法是导入 COM 引用。 |

**还原 PIA 依赖项**

1. 在“解决方案资源管理器”中，单击“显示所有文件”按钮。展开“引用”文件夹并选择“Microsoft.Office.Interop.Excel”。按 F4 以显示**“属性”**窗口。
2. 在“属性”窗口中，将“嵌入互操作类型”属性从“True”更改为“False”。
3. 对 **Microsoft.Office.Interop.Word** 重复此程序中的步骤 1 和 2。
4. 在 C# 中，在 DisplayInExcel 方法的末尾注释掉对 Autofit 的两次调用。
5. 按 F5 以验证项目是否仍正确运行。
6. 重复上一个程序的步骤 1-3 以打开程序集窗口。注意，**Microsoft.Office.Interop.Word** 和 **Microsoft.Office.Interop.Excel** 不再位于嵌入程序集列表中。
7. 双击“清单”图标并滚动引用程序集的列表。 **Microsoft.Office.Interop.Word** 和 **Microsoft.Office.Interop.Excel** 均位于列表中。由于应用程序引用 Excel 和 Word PIA 并且“嵌入互操作类型”属性设置为“False”，因此最终用户的计算机上必须存在两个程序集。
8. 在 Visual Studio 中，单击“生成”菜单上的“清理解决方案”以清理完成的项目。

**请参阅**

[自动实现的属性 (Visual Basic)](https://msdn.microsoft.com/zh-CN/library/dd293589.aspx)

[自动实现的属性（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/bb384054.aspx)

[集合初始值设定项 (Visual Basic)](https://msdn.microsoft.com/zh-CN/library/dd293617.aspx)

[对象和集合初始值设定项（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/bb384062.aspx)

[可选参数 (Visual Basic)](https://msdn.microsoft.com/zh-CN/library/f25e2b6b.aspx)

[按位置和名称传递参数 (Visual Basic)](https://msdn.microsoft.com/zh-CN/library/51wfzyw0.aspx)

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

[早期绑定和后期绑定 (Visual Basic)](https://msdn.microsoft.com/zh-CN/library/0tcf61s1.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)

[Lambda 表达式 (Visual Basic)](https://msdn.microsoft.com/zh-CN/library/bb531253.aspx)

[Lambda 表达式（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/bb397687.aspx)

[如何：在 COM 互操作编程中使用索引属性（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/ee310208.aspx)

[演练：嵌入 Microsoft Office 程序集中的类型信息（C# 和 Visual Basic）](https://msdn.microsoft.com/zh-CN/library/ee317478.aspx)

[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)

[演练：创建你的第一个 Excel VSTO 外接程序](https://msdn.microsoft.com/zh-CN/library/cc668205.aspx)

[COM 互操作 (Visual Basic)](https://msdn.microsoft.com/zh-CN/library/6bw51z5z.aspx)

[互操作性（C# 编程指南）](https://msdn.microsoft.com/zh-CN/library/ms173184.aspx)