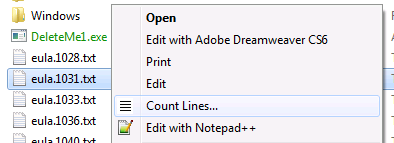
.NET Shell Extensions - Shell Context Menus

.net 外壳扩展之右键菜单

* [**Download SharpShell source code - 1.8 MB**](https://www.codeproject.com/KB/shell/512956/SharpShell_Source_Code.zip)
* [**Download SharpShell Tools - 181.1 KB**](https://www.codeproject.com/KB/shell/512956/SharpShell_Tools.zip)
* [**Download SharpShell core library - 90.9 KB**](https://www.codeproject.com/KB/shell/512956/SharpShell_Core_Library.zip)

# 引言：

一直到.NET 4.0为止，我们都还是不能通过.NET代码完全实现外壳扩展的功能。但是随着framework的不断提高，现在我们已经可以实现这个功能了。在本文中，我将带领你通过C#类文件快速实现右键菜单扩展功能。



（上图是已经实现的功能，“Count Lines”是我自定义的一个右键选项，文章将仔细解释如何实现该功能）

# “.NET外壳扩展”索引：

本文只是“.NET外壳扩展”的一部分，本系列还包括:

1. [.NET Shell Extensions - Shell Context Menus](http://www.codeproject.com/Articles/512956/NET-Shell-Extensions-Shell-Context-Menus)
2. [.NET Shell Extensions - Shell Icon Handlers](http://www.codeproject.com/Articles/522665/NET-Shell-Extensions-Shell-Icon-Handlers)
3. [.NET Shell Extensions - Shell Info Tip Handlers](http://www.codeproject.com/Articles/527058/NET-Shell-Extensions-Shell-Info-Tip-Handlers)
4. [.NET Shell Extensions - Shell Drop Handlers](http://www.codeproject.com/Articles/529515/NET-Shell-Extensions-Shell-Drop-Handlers)
5. [.NET Shell Extensions - Shell Preview Handlers](http://www.codeproject.com/Articles/533948/NET-Shell-Extensions-Shell-Preview-Handlers)
6. [.NET Shell Extensions - Shell Icon Overlay Handlers](http://www.codeproject.com/Articles/545781/NET-Shell-Extensions-Shell-Icon-Overlay-Handlers)
7. [.NET Shell Extensions - Shell Thumbnail Handlers](http://www.codeproject.com/Articles/563114/NET-Shell-Extensions-Shell-Thumbnail-Handlers)
8. [.NET Shell Extensions - Shell Property Sheets](http://www.codeproject.com/Articles/573392/NET-Shell-Extensions-Shell-Property-Sheets)
9. [.NET Shell Extensions - Deploying SharpShell Servers](http://www.codeproject.com/Articles/653780/NET-Shell-Extensions-Deploying-SharpShell-Servers)

# 什么是Shell Context Menus？

Shell Context Menus是在系统中注册，允许扩展shell对象的Context菜单的COM服务器。这里说的对象可以明确到具体的文件类型，例如“.txt”文件，驱动，文件或者其他的文件类型。这时的Context菜单可以通过Windows管理器快速访问到更多的信息。

# 准备：

实现外壳扩展功能有很多工作需要完成：我们需要实现具体的COM接口，需要提供服务，并通过各种方式更新注册表。现在我们可以调用我已经写好的一个叫做“SharpShell”库来完成这些复杂的功能，这样我们的任务就变成了只需要创建一个包含已完成延展功能类的轻量级的类库了。

# 我们的目标：

下面所示代码创建了一个shell扩展，允许你通过右键并选择“Count Lines”来计算出任何文本文件中的行数。另外在文章后半部分我将详细讲解如何创建下面所示的库。我先将代码显示出来的意图是为了强调在调用SharpShell库的情况下如何直截了当的完成库的书写

/// &lt;summary>

/// The CountLinesExtensions is an example shell context menu extension,

/// implemented with SharpShell. It adds the command 'Count Lines' to text

/// files.

/// &lt;/summary>

[ComVisible(true)]

[COMServerAssociation(AssociationType.ClassOfExtension, ".txt")]

public class CountLinesExtension : SharpContextMenu

{

/// &lt;summary>

/// Determines whether this instance can a shell

/// context show menu, given the specified selected file list.

/// &lt;/summary>

/// &lt;returns>

/// &lt;c>true&lt;/c> if this instance should show a shell context

/// menu for the specified file list; otherwise, &lt;c>false&lt;/c>.

/// &lt;/returns>

protected override bool CanShowMenu()

{

// We always show the menu.

return true;

}

/// &lt;summary>

/// Creates the context menu. This can be a single menu item or a tree of them.

/// &lt;/summary>

/// &lt;returns>

/// The context menu for the shell context menu.

/// &lt;/returns>

protected override ContextMenuStrip CreateMenu()

{

// Create the menu strip.

var menu = new ContextMenuStrip();

// Create a 'count lines' item.

var itemCountLines = new ToolStripMenuItem

{

Text = "Count Lines...",

Image = Properties.Resources.CountLines

};

// When we click, we'll count the lines.

itemCountLines.Click += (sender, args) => CountLines();

// Add the item to the context menu.

menu.Items.Add(itemCountLines);

// Return the menu.

return menu;

}

/// &lt;summary>

/// Counts the lines in the selected files.

/// &lt;/summary>

private void CountLines()

{

// Builder for the output.

var builder = new StringBuilder();

// Go through each file.

foreach (var filePath in SelectedItemPaths)

{

// Count the lines.

builder.AppendLine(string.Format("{0} - {1} Lines",

Path.GetFileName(filePath), File.ReadAllLines(filePath).Length));

}

// Show the ouput.

MessageBox.Show(builder.ToString());

}

}

代码可以说简单清晰了，现在就让我们开始仔细研究具体的实现过程吧。

# 第一步：新建项目

首先，我们需要开始一个新的C#类库项目。

提示：你也可以除了C#，还可以使用VB，在本文中虽然所有的代码都是C#，但是方法原理均同理可得。

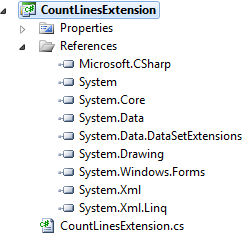
在本例中，我们的项目名称是” CountLinesExtension”.

现在我们需要添加一下的引用：

* System.Windows.Forms
* System.Drawing

之所以需要System.Windows.Forms，是因为我们需要通过WinForms ContextMenuStrip来定义context菜单。而需要调用System.Drawing的原因在于我们想要用到Icons。

将'Class1.cs'重命名为'CountLinesExtension.cs'.那么目前我们项目的就变成了这样了：



# 第二步：引用SharpShell

现在我们需要添加对SharpShell的引用。你可以通过以下这些不同的方法：

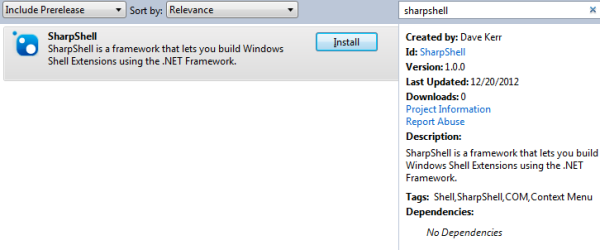
## 添加引用：

在文章最开始的地方下载'SharpShell Library'压缩文件，然后直接对SharpShell.dll文件添加引用即可。

提示：文章开始的下载文件是在此文写的时候的文件，如果你想要获得最新的版本，请通过Nuget或者从[sharpshell.codeplex.com](http://sharpshell.codeplex.com/" \o "SharpShell on CodePlex" \t "_blank)获取。

## Nuget

如果你电脑有Nuget，那么就只需要简单检索SharpShell并安装即可，或者直接访问：<https://www.nuget.org/packages/SharpShell>



## CodePlex

从**CodePlex** 的里面专门的SharpShell主页[sharpshell.codeplex.com](https://www.codeproject.com/Articles/512956/sharpshell.codeplex.com" \o "SharpShell on CodePlex" \t "_blank)中你直接可以得到最新的版本。Nuget上虽然也有最新的可靠的版本，但是CodePlex可以还有可用的已写好的betas，而且那上面的文章里面的版本基本都是可用的。

# 第三步：SharpContextMenu

现在开始就开始真正的研究了，注意了哦。我们的CountLinesExtension继承于SharpContextMenu:

/// &lt;summary>

/// The Count Lines Context Menu Extension

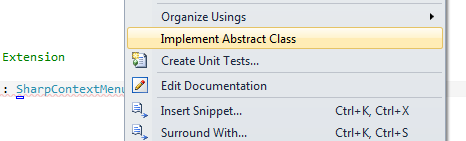
/// &lt;/summary>

public class CountLinesExtension : SharpContextMenu

{

}

现在我们既然继承于SharpContextMenu这个父类，那我们就要来实现这个抽象类。右键选择SharpContextMenu，然后选择'Implement Abstract Class'：



这个操作将会自动生成SharpContextMenu的两个函数-CanShowMenu 和 CreateMenu:

/// &lt;summary>

/// The Count Lines Context Menu Extension

/// &lt;/summary>

public class CountLinesExtension : SharpContextMenu

{

protected override bool CanShowMenu()

{

throw new NotImplementedException();

}

protected override ContextMenuStrip CreateMenu()

{

throw new NotImplementedException();

}

}

通过实现这两个函数没我们可以实现我们所需的所有功能。两个函数功能具体介绍：

## CanShowMenu ？？？？？？好像有点不对

这个函数来决定当前的文件是否是我们需要用来显示扩展菜单的文件。被选中的文件类型是SelectedItemPaths的属性。我们可以通过检查来判断当前的文件类型是不是符合要求的文件类型。如果需要显示菜单就返回true，否则就返回false。

## CreateMenu

这个函数实现主要的功能。我们需要返回WinForms的ContextMenuStrip。

下面是具体的实现：

protected override bool CanShowMenu()

{

// We will always show the menu.

return true;

}

protected override ContextMenuStrip CreateMenu()

{

// Create the menu strip.

var menu = new ContextMenuStrip();

// Create a 'count lines' item.

var itemCountLines = new ToolStripMenuItem

{

Text = "Count Lines"

};

// When we click, we'll call the 'CountLines' function.

itemCountLines.Click += (sender, args) => CountLines();

// Add the item to the context menu.

menu.Items.Add(itemCountLines);

// Return the menu.

return menu;

}

private void CountLines()

{

// Builder for the output.

var builder = new StringBuilder();

// Go through each file.

foreach (var filePath in SelectedItemPaths)

{

// Count the lines.

builder.AppendLine(string.Format("{0} - {1} Lines",

Path.GetFileName(filePath), File.ReadAllLines(filePath).Length));

}

// Show the ouput.

MessageBox.Show(builder.ToString());

}

对于CanShowMenu，我们常常直接返回true（很快我们就会明白为什么我们不需要验证我们的文本文件了）。对于CreateMenu，我们建立了一个显示菜单，里面只有一个选项，也就是'Count Lines'，并实现读出行数的功能。

CountLines这个功能的实现是通过SelectedItemPaths和读取每个文件的函数，然后总结好了显示在弹窗里。

# 第四步：处理COM注册

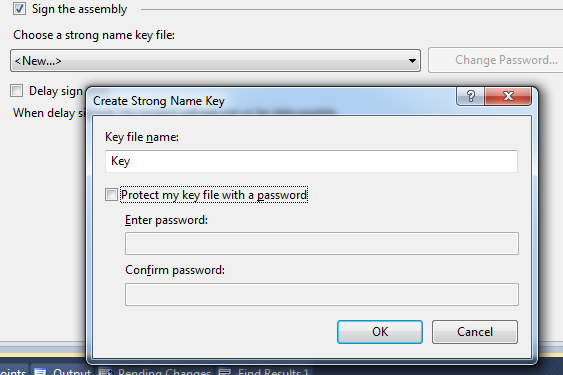
现在所剩的事情不多了。首先我们需要添加COMVisible的引用到我们的类里面。

[ComVisible(true)]

public class CountLinesExtension : SharpContextMenu

为什么呢？因为我们的类虽然看起来不像，但是他本质上也是一个COM Server.如果你了解一些基类，那么你可以看我们刚刚实现的COM接口，例如IShellExtInit, IContextMenu, 和 ISharpShellServer。我们不用了解这里到底做了什么，凡是为了保证系统能够创建我们的扩展，她就必须有这个属性。

接下来，我们需要给程序集一个强名称。这里有很多方法实现这个要求，但是总体来说，这是最好的方法。现在我们单机程序右键，然后选择“属性”。再选择“签名”。选中“为程序集签名”，再选择“选择强名称秘钥文件”里面的“新建”，并输入新建的密钥文件名称。你可以自行选择是否需要使用密码保护密钥文件：



最后一步，我们需要关联我们的扩展到具体的文件类型了。这个的实现可以通过对COMServerAssociation（来源于SharpShell）的引用：

[ComVisible(true)]

[COMServerAssociation(AssociationType.ClassOfExtension, ".txt")]

public class CountLinesExtension : SharpContextMenu

在这里我们做了什么呢？我们告诉了SharpShell，注册服务器时，我们想要关联任何与“.txt”相关的类。这就意味着我们不止是仅限于以.TXT结尾的文件，而是任何同类的文件。更权威的说法是，大多是与txt文件共享同样图标的东西。

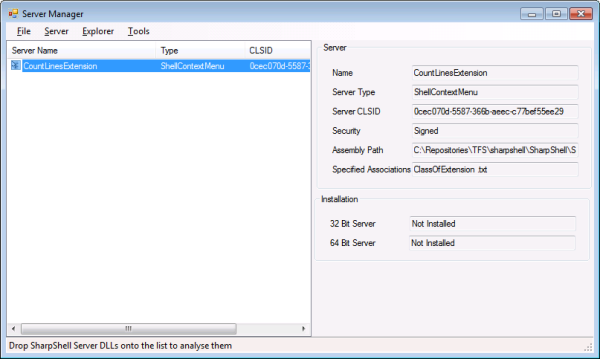
你可以用COMServerAssociation的属性来实现更多的功能，例如你还可以与文件夹，驱动，位置的文件来实现特定的扩展，等等。另外，关于[COM Server Associations](http://sharpshell.codeplex.com/wikipage?title=COM%20Server%20Associations&referringTitle=Documentation" \t "_blank" \o "COM Server Associations Documentation)的详细的文献资料在[SharpShell CodePlex](http://sharpshell.codeplex.com/" \o "SharpShell" \t "_blank)上面。

好了，但目前为止就搞定了！我们已经创建一个可以在作为COM server 在系统中添加自定义菜单的CountLinesExtension程序集项目了。关于注册COM Server和调试部署的任务，我们我们将在下一节，仔细讨论。

# 调试Shell 扩展

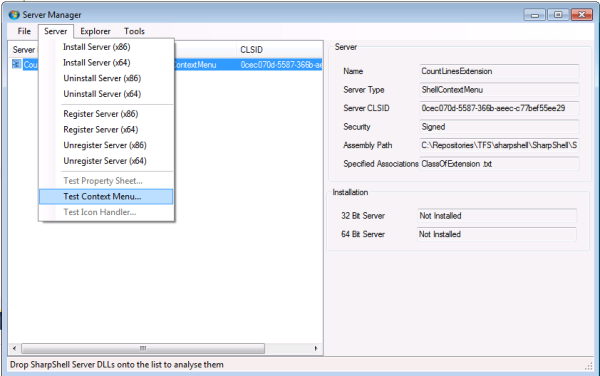
这个Shell扩展功能将被加载到电脑资源管理器时，由于加载.NET COM Server时各种迂回的方式，所以可以说让调试器进入进程并但不执行托管代码几乎可以说是不可能实现的事情。当然，这里还是有一个方法可以快速实现调试。Sharp Shell有一些工具，让实现COM Server变得简单了，其中一个叫做：Server Manager。我们可以用这个工作实现调试。

打开Server Manager工具，然后点击File，选择Load Server，再加载已生成的文件（DLL文件）。你也可以直接拖动文件到主界面中。然后被选中的文件的详细信息就会在旁边显示了。



这个Server Manager非常好用，它可以告诉你服务是否已经安装了和其他各种信息。

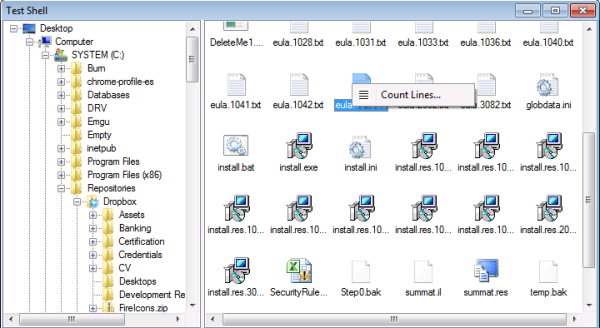
如果你要加载SharpContextMenu服务，那就选中'Tools'，然后你选中再选中'Test Context Menu'。



当你用到了'Test Context Menu'的时候，你会得到一份测试窗口。只是Windows资源管理器应用程序的基本实现。你可以通过右键单击来测试。

提示：不管你的COMServerAssocations已经生成了，测试的窗口都会不断的重新创建。

将调试器附加到servermanager过程将允许你调试你的上下文菜单和测试它的功能，而无需在Windows登记服务器。下面是运行Count Line上下文扩展菜单时测试shell的样子



# 安装和注册Shell扩展

这里又很多方法实现安装和注册Shell扩展。在接下来的这一节里我将仔细介绍这些。

## regasm

你可以通过regasm这个工具来实现安装和祖册shell扩展。当使用regasm的时候shell会被装进注册表中（COM服务器的类ID将放在COM服务器类部分，并与实际服务器文件的路径相关联），也会创建关联。

## Server Manager

在开发过程中，服务器管理工具是我首选的安装/卸载和注册/注销的方法，因为它可以让安装和注册作为单独的步骤处理。它也将让你指定你是安装/卸载在32位还是64位模式中。

## 手动注册表操作

Generally a bad approach, but if you absolutely have to then the MSDN documentation for Shell Extensions describes the changes that must be made to the registry to manually register a COM server, or a Managed COM Server（这句不会翻译）。这个文献资料被放在了下一节。

# 有用的资料

* [在windows中创建快捷菜单处理程序](http://msdn.microsoft.com/en-us/library/windows/desktop/cc144171.aspx)：这是最重要的资料：关于shell菜单扩展如何在windows里具体运行的细节介绍。
* [CodePlex上的SharpShell](http://sharpshell.codeplex.com/)：关于SharpShell项目的专门的主页：其中包罗了各种文献资料，大家的讨论，以及最新的代码研究资源和最新的版本。

# 后记

会逐渐通过SharpShell和.NET的使用来时间实现更多的shell扩展功能，形成一个完整的系列。目前那我正在实现图标处理程序（正在编写文档），关于属性表处理程序问题已经实现了（还有些bug正在解决）。