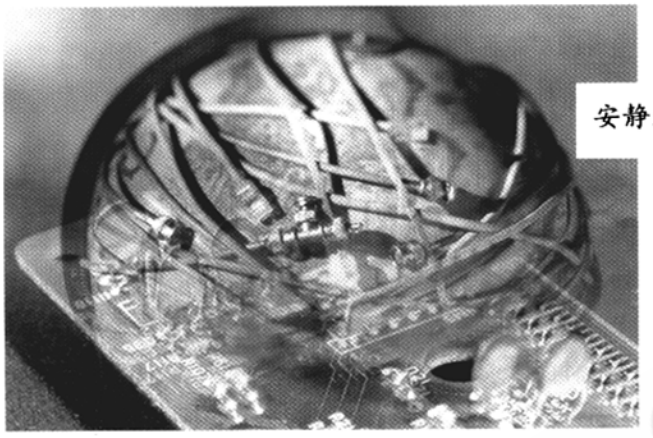
基于COM的Windows Shell扩展及实现



安静斌 李慧 (长沙国防科技大学计算机学院 410073)

摘要：Windows提供了多种扩展或改变其Shell的功能和行为的途径。利用Windows Shell扩展可以很大地改善软件，尤其是桌面工具软件的用户界面。本文介绍了如何基于COM的各种Windows Shell扩展的实现方法及相关问题。

关键词：Windows Shell扩展 用户界面 组件对象模型接口

1 概述

Windows为了实现一个灵活的可定制的命令外壳，提供了多种扩展或改变Shell行为的途径。一个是修改注册表或.ini文件中的相关项，这种方法最简单直接，但所能实现的功能很有限。第二个方法称为Windows Shell扩展处理程序(Windows Shell Extension Handlers)，这是扩展Shell功能的一个更灵活有效的途径，是最常见的一种方式。更高级的方式还有如Shell名字空间扩展(Shell Namespace Extension)、桌面对象(Desktop Objects)等。这里我们着重介绍第二种方法，包括各种Windows Shell扩展处理程序的功能、实现和调试。

2 Winodws Shell扩展处理程序

Windows Shell扩展处理程序分为两类，第一类与不同的Shell文件类似关联，包含下面几种：

(1) 上下文菜单处理程序(Context Menu Handlers)：在显示一个文件对象的上下文菜单时，由Shell调用，用来向文件对象的上下文菜单中添加新的菜单项。例如我们常见的压缩工具Winzip会给压缩文件的上下文菜单添加命令Extract to...，而给其他文件添加Add to Zip。

(2) 图标处理程序（Icon Handlers）：在显示一个文件对象的图标时调用。用来更改不同文件对象或文件类缺省图标。

(3) 数据对象处理程序（Data ObjectHandlers）：当Shell对象被拖放时调用。用来给拖放目标提供其他的剪贴板格式。

(4) 拖放目标处理程序（Drop Target Handlers）：当一个数据对象被拖过或投放到一个文件时调用。用来使一个文件成为拖放的目标。一般情况下文件都不是拖放目标。

(5) 属性表处理程序（Property Sheet Handlers）：在显示一个对象的属性表时调用。用来替换其中的属性页或者添加新的属性页。

第二类则不与固定的文件类相关联，只在Shell进行某项操作(如拷贝、移动、重命名等)之前调用。

(6) 拷贝钩子处理程序（Copy Hook Handlers）：当一个文件夹或打印机对象将要被移动、拷贝、删除或重命名时调用。通过它可以同意或否定正在进行的操作。

(7) 拖放处理程序（Drag-and-Drop Handlers）：当一个文件被用户用鼠标右键拖动时调用。可以通过来修改此时显示的上下文菜单。

(8) 图标覆盖处理程序（Icon Overlay Handlers）：显示一个对象的图标之前调用。可以用来为对象指定图标覆盖(Icon Overlay)，如常见的快捷方式图标右下角的箭头图样。

(9) 列处理程序（Column Handlers）：当资源管理器显示一个文件夹的“详细资料”视图时调用这个处理程序。通过它可以向视图中添加自定义的列。

(10) 搜索处理程序（Search Handlers）：用来实现定制的搜索引擎。用户可以通过“开始”菜单或资源管理器来调用。

3 注册

与所有的COM对象相同，Shell扩展也必须在Windows注册表中进行注册后才能使用。

首先，每个Shell扩展对象应该在HKEY_CLASSES_ROOT/CLSID下面注册自己的GUID(CLSID)，并这个表项下面添加键InProServer32，用来指定对应的Shell扩展DLL的存放位置和线程模型。例如：

```
[HKEY_CLASSES_ROOT/CLSID＼{9a55049f-badd-4e66-a0f8-fe6ecf062dd0}]
    @="TestExtHandler"
[HKEY_CLASSES-ROOT/CLSID/{9a55049f-badd-4e66-a0f8-fe6ecf062dd0}
    ＼InProcServer32]
    @="C://Windows//system//TestShellExt.dll"
"ThreadingModel"="Apartment"
```

要使一个扩展处理程序生效，除了进行上述的注册外，对于文件类，还必须在其ProgID键的子键ShellEx下面建立一个扩展处理程序子键，而对于预定义的Shell对象，则需要在其对应的类型名键的子键ShellEx下建立扩展处理程序子键。

对于下面的几种扩展处理程序，在其扩展处理程序子键下可以建立注册多个处理程序，每个用一个自己表示，子键名为扩展处理程序对象的CLSID。

扩展处理程序类型	扩展处理程序子键名
列处理程序	ColumnHandlers
上下文菜单处理程序	ContextMenuHandlers
拷贝钩子处理程序	CopyHookHandlers
拖放处理程序	DragDropHandlers
属性表处理程序	PropertySheetHandlers

而对于另外几种，在其扩展处理程序子键下只能注册一个处理程序，并用其CLSID作为该子键的缺省值。

扩展处理程序类型	扩展处理程序子键名
数据对象处理程序	DataHandler
拖放目标处理程序	DropHandler
图标处理程序	IconHandler
缩略图处理程序	{BB2E617C-0920-11d1-9A0B-00C04FC2D6C1}
提示信息处理程序	{00021500-0000-0000-C000-000000000046}

下面是一个文件类的上下文菜单和属性表扩展处理程序的注册：

```
[HKEY_CLASSES_ROOT＼]
    .test=TestProgram.1
[HKEY_CLASSES_ROOT＼CLsID＼]
    {00000000-1111-2222-3333-444444444444}
        InProcServer32=D:＼Dir＼TestCommand.dll
            ThreadingModel=Apartment
    {11111111-2222-3333-4444-555555555555}

        InProcServer32="D:＼Dir＼TestProSheet.dll
            ThreadingModel=Apartment
[HKEY_CLASSES_ROOT＼]

    TestProgram.1=TestProgram Application
        Shellex
            COntextMenuHandlers
                {00000000-1111-2222-3333-444444444444}
            PropertySheetHandlers
                {11111111-2222-3333-4444-555555555555}
```

需要指出的是，在Windows NT和Windows 2000下，还必须在注册表键

[HKEY_LOCAL_MACHINE＼Software＼Microsoft＼Windows/CurrentVersion＼Shell Extensions＼Approved＼]

下面注册每一个扩展处理程序对象的CLSID，而且只有管理员才具有访问此键的权限。

4 实现

上面我们介绍的Windows Shell扩展程序，在具体实现时大部分都与具体的类型相关，但还是有一些公共的部分。

首先，所有的Shell扩展处理程序都实现为一个在程(In-process)COM对象，从而必须为其指定一个GUID并在注册表中进行注册。与其他的在程COM对象一样，Shell扩展处理程序也实现为一个动态链接库(DLL)，并输出下面的标准函数：

· DllMain: DLL的标准入口函数。

· DllGetClass Object:输出对象的类工厂。

· DllCanUnloadNow:COM调用此函数来决定该对象是否正在被客户使用，如果不是，系统就可以卸载该DLL并释放内存。

与所有的COM对象一样，Shell扩展处理程序同样必须实现IUnknown接口和一个类工厂(Class Factory)。此外其中大部分还必须实现IPersistFile或IShellExtInit接口，Shell通过它们来对扩展处理程序进行初始化。

下列扩展处理程序必须实现IPersistFile接口：

· 图标处理程序

· 数据对象处理程序

· 拖放目标处理程序

IPersistFile 接口可以使一个对象能够保存到一个磁盘文件或者从磁盘文件装入。除了继承IUnknown的方法外还有六个方法，其中 GetClassID 是从 IPersist 继承的。由于一般情况下，Shell 扩展不需要读写磁盘，所以只有 GetClassID 和 Load 这两个方法需要真正地实现。Shell 首先调用 GetClassID，该函数扩展处理程序对象的 CLSID。接着调用 Load 并传入两个参数：pOleStr 和 dwFlags。前者是一个 Unicode 字符，包含 Shell 将要进行操作的文件或文件夹的名字；后者则指出了访问文件的方式，由于通常不需要访问文件，一般为0，为了便于以后引用，这里只要简单地保存起来就可以了。

而对于下列扩展处理程序必须实现 IShellExtInit 接口：

· 上下文菜单处理程序

· 拖放处理程序

· 属性表处理程序

与 IPersistFile 一样，IShellExtInit 接口同样用来对扩展处理程序进行初始化。IShellExtInit 只有一个方法：Initialize，它有三个参数。

· pIDFolder：包含一个文件夹的PIDL，对于属性表扩展此项为NULL，对于上下文菜单扩展，为该上下文菜单所属的项所在的文件夹的 PIDL，对于拖放扩展处理程序为目标文件夹的 PIDL。

· pDataObject: 包含一个指向一个数据对象的 IDataObject 接口的指针。该数据对象包含一个或多个 CF_HDROP 格式的文件名。

· hRegKey：包含文件或文件夹对象的注册表键。

Initialize 方法的实现应该保存这三个参数以便接下来使用。

下面讨论各种与具体的Shell扩展类型相关联的接口及其实现。

(1) 上下文菜单处理程序和拖放处理程序。上下文菜单处理程序除了 IUnkown 外，还必须输出 IShellExtInit 和 IContextMenu 接口。对于 IContextMenu 也可以使用 IContextMenu2 和 IContextMenu3。

IContextMenu 有下面三个方法：

① GetCommandString：返回菜单命令的名称或者描述。当用户将鼠标移到一个命令上时，Shell 会调用此函数，并将返回的字符串显示在资源管理器的状态条上。返回的字符串可以是 ANSI 或 Unicode。

② InvokeCommand：菜单命令的处理函数。

③ QueryContextMenu: Shell 在显示菜单之前调用此方法，并传入菜单的句柄，所以在此可以很方便使用 InsertMenu 和 InsertMenuItem 来添加菜单项甚至修改已有的项(不提倡)。这里要保存所添加的每一项的标识，用来在随后的 InvokeCommand 和 QueryContextMenu 中区别不同的菜单项。

可用 IContextMenu2 或 IContextMenu3 替换 IContextMenu，来实现自画(OwnerDraw)菜单项。

拖放处理程序的实现与上下文菜单相同，但其注册位置一般为：

[HKEY_CLASSES_ROOT＼Directory＼Shellex＼DragDropHandlers＼]

(2) 属性表处理程序。属性表处理程序输出 IShellExtInit 和 IShellProp-SheetExt 接口，对于后者只需要实现 IShellPropSheetExt::AddPages 方法：

① 为结构PROPSHEETPAGE的各项赋正确的值。其中，pcRefParent的值应设为扩展处理程序的引用计数，这样就可以防止扩展处理程序在属性表还在显示的时候被卸载；pfnCallBack 指向一个自定义的回调函数，该函数在属性页被创建和销毁时调用，可以用来处理所有与这个属性页相关的消息。

② 将赋好值的PROPSHEETPAGE结构作为参数调用 CreatePropertySheetPage，返回句柄 HPAGE。

③ 调用作为 AddPages 的参数传入的指针 IpfnAddPage 所指向的函数。

(3)数据对象处理程序。数据对象处理程序输出 IPersistFile 和 IDataObject 接口。Shell 在调用 IPersistFile 进行初始化后，调用 IDataObject 来访问处理程序自定义格式的数据。

(4) 拖放目标处理程序。输出 IPersistFile 和 IDropTarget 接口。Shell 通过前者来获取扩展处理程序的 CLSID 并传入被拖动的文件名字。初始化之后，Shell 会调用 IDropTarget 的相应方法，具体实现可参考 IDropTarget 的说明和有关 OLE 数据传输的内容。

(5) 图标处理程序。图标处理程序输出 IPersistFile 和 IExtractIcon 接口。Shell 调用 IExtractIcon::GetIconLocation 来获取图标的位置信息，图标的位置用一个文件和一个索引来标识，如(Shell32.dll,3)，这里，szIconFile 返回文件名，piIndex 返回索引值。

(6) 图标覆盖处理程序。与其他处理程序不同，所有图标覆盖处理程序均由 Shell 在启动时进行初始化，除了 IUnknown 外只输出 IShellIconOverlayIdentifier 接口，Shell

首先调用该接口的方法GetOverlayInfo来获取图标的位置信息，并将该图标加入系统的图像列表。GetPriority方法则用来获取图标覆盖的优先级，为一个0～100之间的数，100最低。当对同一个对象有多个图标覆盖请求时，优先级高的有效。在显示一个对象的图标之前，Shell调用IsMemberOf方法，并传入该对象的名字，如果一个扩展处理程序要显示其图标覆盖就返回S_OK。此时GetOverlayInfo将被再次调用，但这时返回的图标位置信息只作为索引，并不再次从文件读取图标。图标覆盖处理程序的注册位置为：

[HKEY_LOCAL_MACHINE＼Software＼Microsoft＼Windows＼Current Version＼Explorer＼ShellIconOverlayIdentifiers＼]

(7) 拷贝钩子处理程序。这种扩展处理程序除IUnkown外只输出ICopyHook接口，该接口只有一个方法CopyCallBack，当一个文件夹将要被移动时，Shell调用此方法，返回IDYES表示同意正在进行的操作，返回IDNO，Shell将停止对该文件夹的操作，但批处理中的其他操作仍继续，如果返回IDCANCEL，Shell则停止所有的操作。注册位置为：

[HKEY_CLASSES_ROOT/Directory＼Shellex＼CopyHookHandlers＼]

(8) 列处理程序。列处理程序输出IUnknown和IColumnProvider两个接口。IColumnProvider的方法Initialize传入将要显示的文件夹的路径和名字，GetColumnInfo返回列的相关属性如标题、标识符等，GetItemData则返回文件夹中每个文件的相关信息作为该列的每一项。注册位置为：

[HKEY_CLASSES_ROOT/Folder＼ShellEx＼ColumnHandlers＼]

(9) 搜索处理程序。搜索处理程序的实现可分为三种情况：静态、动态和基于DHTML的。基于DHTML的搜索扩展只有Windows 2000及以后版本支持。

搜索处理程序的工作过程和上下文菜单处理程序基本相同，也输出IUnknown、IShellExtInit和IContextMenu三个接口。对于静态扩展，其菜单项根据注册表中的信息创建，所以IShellExtInit::Initialize和IContext/Menu::QueryContextMenu都不用实现。其注册位置为：

[HKEY_LOCAL_MACHINE＼Software＼Microsoft＼Windows＼CurrentVersion＼Explorer＼FindExtensions＼Static＼]

动态扩展则与上下文菜单扩展相同。其注册位置为：

[HKEY_LOCAL_MACHINE＼Software＼Microsoft＼Windows＼CurrentVersion＼Explorer＼FindExtensions＼]

5 调试

在调试一般的程序时，通常只需要在调试工具的Debug模式下设置一些断点，或者单步执行。但由于Windows Shell扩展是由Shell装载并执行的，所以必须想办法从调试器中运行Windows Shell。对于Microsoft Visual C++可以这样做：

(1) 在Visual C++中打开我们要调试的Windows Shell扩展程序的工程。

(2) 在Project菜单中选择Setting，选中Debug属性页，在“Executable for debug session”编辑框中输入Explorer.exe的路径，如：C:＼WINDOWS＼EXPLORER.EXE

(3) 从“开始”菜单选择“关闭系统”，然后在按住Ctrl、Alt和Shift键的同时用鼠标按下“取消”按钮。这时，Shell就会被关闭，桌面上的图标、任务条和开始菜单都会消失。

(4) 使用Alt+Tab组合键回到Visual C++，然后就像调试一般的动态链接库一样，设置断点并运行，这时Shell会重新启动。

对于Windows NT和Windows2000，可以通过将注册表键

[HKEY_CURRENT_USER＼Software＼Microsoft＼Windows＼CurrentVersion＼Explorer]

下的值DesktopProcess置为1，从而可以在一个独立于Shell的Windows资源管理器进程中调试Shell扩展处理程序，这样就不需要重启Shell。

6 结束语

通过Windows Shell扩展可以很大地改善软件的易用性，并体现出以文档为中心的用户界面设计思想，因此，Shell扩展构成了Windows平台软件用户界面设计中的一项重要的内容，尤其是桌面软件的设计中更是如此。我们在FindMedia(Desktop Edition)———一个多媒体资源管理软件的设计和实现中，利用Windows Shell扩展很明显地改善了用户界面功能，使软件的使用方式更为合理，取得了很好的效果。■

参考文献

1 *Windows 2000 Resource Kit Reference. Microsoft Corporation, 2000.*

2 *Windows 2000 SDK Rdference. Microsoft Corporation, 2000.*

3 *Nancy Winnick Cluts. Programming the Windows 95 User Interface. Microsoft Corporation, 1995.*

4 *Don Box. Essential COM. Addison-Wesley Publishing Company, 1998.*