ONE

Windows Shell Programming

With each operating system release, service pack, and Internet Explorer upgrade, Microsoft adds more possibilities for vendors to extend and enhance the Windows user interface. This has been the case with earlier versions of Windows through its latest release, Windows 2000. The first version of Windows many of us did any serious development for was Windows 3.x. With the 3.x versions, you could add limited modifications to the shell:

- Control Panel applets
- Screen savers
- File manager extensions

The early file manager extensions allowed developers to only add menu items and toolbar buttons. On any given Windows installation, a computer could have up to five extensions installed. A lot of time has passed since then, and the developers at Microsoft have continued to enhance the extensibility of the shell. With Windows 2000, the shell still allows you to write your own Control Panel applets and screen savers, but you can also do more—much more. For example, you can:

- Customize the Windows taskbar
- Add extra menu options when right-clicking on a file
- Add advanced handling of folders, drives, and printers
- Handle new data formats when a file is copied and pasted
- Allow specialized actions when a file gets dropped on a file type
- Monitor copying of folders, drives, and printers
- Add your own views of data in Windows Explorer

- Add various types of toolbars to Windows and Internet Explorer
- Allow the shell to do work for you in your own applications, such as filename auto-completion

Looking at the list, it should be clear that the shell is no longer a simple windowing environment. We can customize Windows as we see fit and Microsoft gives us this ability at a price—they increase the chance of Windows' instability. This level of customization makes it such that no two installations of Windows ever remain the same for very long. As the user installs more applications on their machine, the shell gains new capabilities and features. A few applications will add items to the system tray (clock area) on the taskbar. Another may add items to a context menu in Explorer. If the machine suddenly (or worse, slowly) becomes unstable, the user will not blame the people at Foobar, Inc. for a bad shell extension. To the user, all of the right-click functionality, drag-and-drop capabilities, and Explorer enhancements are part of the operating system , so Microsoft will be blamed for shoddy workmanship. This means that we have a responsibility to create stable additions so that people continue to trust Microsoft operating systems with their corporate and personal data. Failing to do so may lead Microsoft to remove our ability to customize the shell.

1.1 Goals of this Book

It is my belief that a book on a specific technology should not just explain the technology; it should also make it easier to use that technology. For example, I have a few fairly popular C++ books:

- *Effective C++* by Scott Meyers
- *More Effective C++* by Scott Meyers
- *Large Scale C++ Software Design* by John Lakos
- *C++ FAQs* by Marshall Cline and Greg Lomow
- *The C++ Programming Language* by Bjarne Stroustrup

I have learned more from the first four books than from the fifth, which was written by the inventor of the language! Do not get me wrong, Stroustrup's text has helped me around syntactical errors, but the others have given me tools to write better code faster.[1] I will try to show how to write good code that uses and enhances the Windows shell. In order to accomplish this, I provide a lot of fully functional examples as well as what I hope are production-quality libraries and wizards to make your life easier.

1. Stroustrup has written many articles and USENET posts that will help you use the language better. However, *The C++ Programming Language* is just a language reference, not a "how to do it better" kind of book.

In the following pages I want to accomplish a number of things:

1. **Explain how the shell works and what opportunities for enhancement are available.** The first thing the reader expects from this book is that it will explain the shell in detail. This means going over many of the interfaces, functions, and other items needed to understand the shell. There is a need to explain these things more clearly than the Microsoft Developer Network does.
2. **Show how to use MFC and ATL to enhance the shell.** The book targets a specific group of developers: those who use Visual C++ as their development tool of choice. Consequently, I have a responsibility to show the readers how to develop solutions that leverage what they already know. For example, if something displays a window, the reader wants to know how to use a CWnd to handle the message loop.
3. **Speed up development time.** This book targets programmers who have projects to complete as fast as possible. Many of them will not even read an entire chapter unless they have problems that they cannot figure out. For some readers, the most valuable part of the book will probably be the included CD. To them, the book is nothing more than a user manual for that CD.

Because I went the extra step to see how to create generic solutions, I forced myself to understand the technology outside the scope of my current sample project. Many of the libraries and wizards presented in the book take the unfamiliar Windows shell and mold it to the world of the MFC/ATL developer. C++ is an extremely pliant language and will let you do almost anything. Together, MFC, ATL and C++ allow you to do amazing things. Using them along with the libraries and wizards in this book, the reader will realize the benefits of learning the ins and outs of MFC.

1.2 What is the Windows Shell?

The Windows shell is nothing like the UNIX or the DOS shells. With UNIX shells and their command line interfaces (CLI),[2] users have to know that a feature exists before they can use it. Compare this to the Windows shell, which can advertise new features to the user. For example, let us look at how a user would go about opening a JPEG file. To view the picture with a command line interface, users have to know that they need a graphics viewer to

2. If you have no UNIX experience, think back to the days of MS-DOS. Now imagine a lot more expressiveness and power on the command line.

look at the file and how to start a viewer. Windows provides hooks that allow the viewer application to advertise its association with the JPEG files. When users select a file with the right mouse button, Windows will reveal a menu allowing them to view, move, or possibly even translate the file to another graphics format. They discover these capabilities simply by knowing that a right mouse click will tell them what they can do with the file. This circumvents the need to tell the users about all the different programs available for file manipulation.

The Windows shell provides the means to interact with the computer. The shell is composed of the following elements:

- **The Desktop.** When Windows starts up, this is the first thing a user sees.
- **The Taskbar.** The taskbar provides a clock, a way to start applications, and a place for applications to notify the user about program activity. With Active Desktop installed, the taskbar can contain toolbars beyond the standard task list.
- **The Control Panel.** The Control Panel provides a single location to configure devices and programs on the computer. Besides the ability to add applets to the Control Panel, some of the packaged Control Panel applets allow third parties to add extra property pages.[3]
- **Internet Explorer.** Like it or not, Microsoft has made the browser part of the shell. For some time now, we have been able to view drives across the network as if they were on the local machine. It is easy to see the benefits of viewing FTP sites the same way. The only stretch happens when looking at hyperlinks. Hyperlinks and HTML documents allow us to navigate to new directories and files by clicking on links in files. This navigational model seems as valid as the hierarchical file systems we use on a daily basis. Because of the spider web that HTML documents produce, browsers present the best-known way to navigate these documents. Integrating the browser allows Internet Explorer to be the ultimate in common file type navigation.
- **Windows Explorer.** Windows Explorer allows us to navigate whatever information is presented on our machines. It provides the ability to move files around, drop files on other files, and display information regarding files, among other things. If you want to add something that is not file related, you can do so by extending Explorer's capabilities.[4]
- **File Viewers.** The people at Microsoft will cringe when they see this because the feature has been removed from the operating sys-

3. Specifically, these are the display, keyboard, and mouse applets.

4. See Chapter 9 for how to customize Explorer.

tem as of Windows 98 SE. Still, Windows 95, 98, and Windows NT 4.0 all provide hooks that allow vendors to distribute DLLs which present a read-only view of a file. Typically users can distribute the viewer without risk of copyright or licensing violations.

- **Disk Cleanup.** Starting with Windows 98, the shell provides a janitor named CLEANMGR.EXE. Applications can provide the janitor with instructions on how to free up space on the local hard drives. This way, when users need more space, they do not have to start by deciding which files they should get rid of. Instead, the janitor gets rid of all the truly useless stuff first.
- **The Registry.** The entire registry is not a part of the shell, although two parts of the registry do a significant job of customizing the user's interaction with Windows: HKEY_CURRENT_USER (HKCU) and HKEY_LOCAL_MACHINE (HKLM). Because of this, the shell team has created a number of functions that make it easier for programs to interact with those two hives.

1.3 Chapter Summaries

My aim was to present each topic so that it stands on its own. You should be able to go to any one chapter and find all the information you need to get your job done. The only required reading in this book is this chapter and the chapter that covers your topic, unless your topic is namespace extensions. A namespace extension can be a fairly complex beast. As a result, that topic is split into three chapters: one to explain namespace extensions, one to document the library and wizard I wrote, and one to design and create an extension. Furthermore, in each chapter I reference the related material found in the appendices and other chapters as needed. For example, many of the shell customizations require you to implement the COM interface IcontextMenu, so whenever IContextMenu enters the discussion, I reference section 7.2.

Each of the following sections describes a chapter in the book and what extending the shell in that area can do.

1.3.1 Chapter 2: The Taskbar

This chapter explains how to manipulate the taskbar. It teaches how to do the following:

- Get information about the taskbar (location, size, auto-hide, always-on-top)
- Add and remove taskbar buttons, which can either increase or decrease the number of applications that appear to be running
- Add and remove links on the Start menu

- Add icons to the system tray.[5] It also shows how to animate an icon in the tray

1.3.2 Chapter 3: Application Desktop Toolbars

The best-known application desktop toolbar, or *appbar*, is the taskbar. The first runner up in popularity is the Microsoft Office Shortcut Bar. Appbars provide a nice way to present information without getting in the user's way. They usually dock to one of the edges of the desktop and sit there. The user can even make them automatically hide themselves so that they take up almost no space on the screen. Chapter 3 covers the following topics:

- Guidelines for creating appbars
- Explanation of how appbars work
- Explanation of AppBarLib and the MFC Application Desktop Toolbar AppWizard
- How to use the tools presented to build an appbar of your own

1.3.3 Chapter 4: Control Panel Applets

The Control Panel provides a place to put any utilities for configuring hardware or software. For example, you would place applets to configure a service or fax machine there. People expect to find the configuration utility for background processes and hardware in the Control Panel. Microsoft also has a new utility out: the Microsoft Management Console. If your configuration user interface works best in a dialog, write an applet. Otherwise, write an MMC snap-in.[6] Chapter 4 covers the following items:

- How to decide if the Control Panel is an appropriate place to put your applet
- Ways of packaging Control Panel applets
- Control Panel basics
- Motivation and design of a Control Panel applet wizard
- Building an applet using the wizard

1.3.4 Chapter 5: Screen Savers

Screen savers do so much more than entertain and delight bored workers. They also help extend the life of a monitor, "lock up" a computer when the user is away, and hide what one was working on when called away from the machine. This chapter presents a library that is feature-compatible with

5. This is the little window on the taskbar that the clock lives in.

6. MMC Snap-ins are not part of the shell. Thankfully, wizards and libraries are provided for them in Visual C++.

SCRNSAVE.LIB with an added benefit: you can use MFC to do all your work. Chapter 5 covers the following topics:

- Screen saver responsibilities
- Screen saver internals
- Benefits of SCRNSAVE.LIB over writing your own
- An MFC Screen Saver App wizard
- Writing a screen saver using the wizard

1.3.5 Chapter 6: File Viewers

A file viewer presents a read-only view of the file. You can look at and sometimes even print the file, but you cannot do anything else to the file. Viewers exist for most Microsoft documents, including Word, Excel, and PowerPoint. You can also find them for viewing other file types, including bitmaps, text files, and executables (DLLs and EXEs). Chapter 6 covers these topics:

- File viewer basics—when to create a viewer, how to invoke one, etc.
- File viewer internals
- A File Viewer library/wizard
- A sample file viewer

1.3.6 Chapter 7: Shell Extensions

If you want to find out how to set what users will see and what they can do within Windows Explorer, check out this chapter. You can also do some interesting things to the folders, printers, and drives attached to the machine. Chapter 7 explains the following items:

Extensions registered by file type (a.k.a. class)

- **Context Menu Handler:** Adds items to the context menu (a.k.a. right-click menu) for a file object. You may add verbs and other actions for a file type. (7.2)
- **Icon Handler:** Typically used to add icons specific to the file object. You can also use this to add icons for all files belonging to the same class. (7.3)
- **Data Handler:** Provides an IDataObject interface for a specific class type. The shell passes this interface to the OLE DoDragDrop function. (7.4)
- **Drop Handler:** Provides drop behavior for files that can accept drag and drop objects. (7.5)
- **Property Sheet Handler:** Adds pages to the property sheet that the shell displays for a given file type. You can also extend items such as the Display Properties dialog using a property sheet handler. (7.6)

Extensions associated with file operations and directories (move, copy, rename, etc.)

- **Copy Hook Handler:** These get called whenever a folder object is about to be copied, moved, deleted, or renamed. The handler can allow or prevent the operation. (7.7)
- **Drag-and-Drop Handler:** A context menu handler that the shell calls when the user drops an object after dragging it to a new position. (7.8)

1.3.7 Chapter 8: Disk Cleanup Handlers

Today's large hard drives allow us to install many programs and store thousands of files. Because of all this space, most of us do not actively clean up anymore. If an application leaves temp files strewn about our machines, we will not notice the decline in space for months. Other programs, such as web browsers, cache web pages to speed up perceived download times. As a result, the task of maintaining one's hard drive has become very difficult. To address the problem, Microsoft introduced disk cleanup handlers with Windows 98. As a developer you have a responsibility to provide a handler for any application you create that leaves behind temporary or unnecessary files. A handler also comes in handy when an application that you think is well-behaved uses temporary files. Many applications will leave these behind if the computer loses power. A handler can clean up part of the resulting mess. Most sizable applications need a cleanup handler. On any non-trivial project, make sure you include development time for one of these.[7] Chapter 8 covers the following topics:

- The Disk Cleanup Utility and its relationship to disk cleanup handlers
- The various interfaces employed by disk cleanup handlers and how they work
- An example program

1.3.8 Chapter 9: Namespace Extensions

Starting with this chapter and continuing through Chapter 11, I departed from the rule of one topic per chapter. Developing a namespace extension can be as complex as developing a full-scale application. As a result, I chose to separate the subject matter into distinct chapters. When a namespace extension is activated, it assumes a lot of control over Explorer's menus, toolbars, and right-hand pane. You have to make a lot of design decisions and understand user expectations. This chapter goes into detail explaining the interaction between Explorer and an extension. It then explains what a user will expect from a full-featured namespace.

7. I would really appreciate a cleanup handler from the Visual Studio team that would delete all the PCH, SBR, OBJ, APS, PLG, and OPT files from the hard drive.

1.3.9 Chapter 10: Tools to Build a Namespace Extension

Once the interaction and design of a namespace extension has been explained (Chapter 9), we need to make the whole experience of building a namespace something easier to do. For example, I have no desire to build menus the way I would for context menu handlers. I would rather handle these by building them using the Visual Studio menu editor. This chapter explains a library and wizard that allow quick creation of a namespace extension. Along the way, I explain why I chose one design over another so that you have more insight as you debug your own namespaces.

1.3.10 Chapter 11: Namespace Extension Example: The Registry

This chapter covers the design and construction of a namespace extension that contains many of the capabilities found in REGEDIT.[8] It also covers all the decisions I had to make:

- What should I put into the Explorer menu?
- What buttons should show up in the toolbars?
- What should the context menus look like?
- What data should I display?

1.3.11 Chapter 12: Explorer Bars and Desktop Bands

Way back in Chapter 2, I explained how to manipulate the taskbar but avoided the topic of adding extra band objects. The topic really deserves separate treatment because of the breadth of things you can do. Using band objects, you can add the following types of toolbars:

- **Desk bands:** These augment the toolbars available in the taskbar. They are only available when Active Desktop has been installed. This feature is included with Windows 98, 2000, and courtesy of Internet Explorer, version 4.*x* and Active Desktop.[9]
- **Comm Bands:** These display information at the bottom of Internet Explorer and Windows Explorer. Only one comm band can display at any given time.
- **Explorer Bands:** These display on the left hand side of Internet Explorer and Windows Explorer. Only one explorer band can display at any given time.

8. This example has a few more capabilities than the SDK registry namespace extension example and takes nothing from the SDK version.

9. The desktop update did not ship with Internet Explorer 5.0 as an installable component.

- **Radio Bars:** You can add extra toolbars to the top of Internet Explorer and Windows Explorer to do whatever you want them to do.
- **HTML Based Bands:** Microsoft implemented an HTML-capable band object. This band object allows you to display HTML by simply writing a REG script and some HTML.

1.4 Versions of the Shell

In order to use the content in chapters 6 through 12 and the appendices effectively, you must be cognizant of the shell version your application works with. Your user will be able to use pretty much anything you write as long as they are running version 4.72 of the shell, distributed with Internet Explorer 4.01 and Internet Explorer 4.0, SP1. In the past, Microsoft has bundled interim shell updates with Internet Explorer, not as a separate package. The grid below shows the various versions of the shell and gives you an idea of how to upgrade your users to the correct version:

Version	DLL	Distribution Platform
4.00	All	Windows 95/NT 4.0
4.70	All	Internet Explorer 3.x
4.71	All	Internet Explorer 4.0
4.72	All	Internet Explorer 4.01 and Windows 98
5.00	Shlwapi.dll	Internet Explorer 5
5.00	Shell32.dll	Windows 2000
5.80	Comctl32.dll	Internet Explorer 5
5.81	Comctl32.dll	Windows 2000

Along with the preceding table, Microsoft delivers these clarifying notes:[10]

Note 1

The 4.00 versions of Shell32.dll and Comctl32.dll are found on the original versions of Windows 95 and Windows NT 4. New versions of Commctl.dll were shipped with all Internet Explorer releases. Shlwapi.dll first shipped with Internet Explorer 4.0, so its first version number is 4.71. The shell was not updated with the Internet Explorer 3.0 release, so Shell32.dll does not have a version 4.70. While Shell32.dll versions 4.71 and 4.72 were shipped with the corresponding Internet Explorer releases, they were not necessarily installed (see Note 2). For subsequent releases, the version numbers for the three DLLs are not identical. In general, you should assume that all three DLLs may have different version numbers, and test each one separately.

10. From MSDN. Article Title: *Shell and Common Controls Versions.*

Note 2 All systems with Internet Explorer 4.0 or 4.01 will have the associated version of Comctl32.dll and Shlwapi.dll (4.71 or 4.72, respectively). However, for systems prior to Windows 98, Internet Explorer 4.0 and 4.01 can be installed with or without the *integrated shell*. If they are installed with the integrated shell, the associated version of Shell32.dll will be installed. If they are installed without the integrated shell, Shell32.dll is not updated. In other words, the presence of version 4.71 or 4.72 of Comctl32.dll or Shlwapi.dll on a system does not guarantee that Shell32.dll has the same version number. All Windows 98 systems have version 4.72 of Shell32.dll.

Note 3 Version 5.80 of Comctl32.dll and version 5.0 of Shlwapi.dll are distributed with Internet Explorer 5. They will be found on all systems on which Internet Explorer 5 is installed, except Windows 2000. Internet Explorer 5 does not update the shell, so version 5.0 of Shell32.dll will not be found on Windows NT, Windows 95, or Windows 98 systems. Version 5.0 of Shell32.dll will be distributed with Windows 2000, along with version 5.0 of Shlwapi.dll, and version 5.81 of Comctl32.dll.

1.5 Summary

Chapter 1 outlines what the book is about and where to find information on the various extensions. I have tried to make each chapter independent of the others and I have cross-referenced other sections as needed. Chapters 9 through 11 break this rule because namespace extensions make for bigger projects than things like context menu extensions. Section 1.4 outlined the versions of the shell and how to get them to your users.

This book covers a lot of the shell, but you may discover that pieces are missing. Before you send me an e-mail, flaming me for incompetence, poor upbringing, or anything else, check out the Prentice Hall Web site at *www.phptr.com* or visit my site at *www.scottseely.com*. These sites are updated regularly, so if neither contains the new information you have uncovered, e-mail me to let me know what I've missed. The first person to name any missing feature will be named on the Web site and in the acknowledgements section of the next revision of this book. I will list your name (unless you ask me not to) and the item you caught to acknowledge your contribution. I apologize for not covering everything in the shell this first go around. Microsoft has updated the shell seven times in five years. I had to decide to leave out some minor features.

Now, go extend the shell!