Creating OLE 2.0 Object Containers, Object Servers and Automation Servers with Visual C++ and MFC

by

Richard Hale Shaw

Richard Hale Shaw is a Contributing Editor to PC Magazine, Windows Tech Journal, MFC Journal, Visual Basic Programmer's Journal and Microsoft Systems Journal. He's also the editor of NT Developer Journal and writing *Visual Programming++* for Addison-Wesley. He can be reached at MCIMAIL 399-8368 or CompuServe 72241,155.

Course Outline

- Goals for this class
- A Review of Visual C++ and MFC
- An Overview of OLE
- The MFC OLE Classes
- Building OLE Object Containers
- Building OLE Object Servers
- The Big Payoff: Building OLE Automation Servers
- The Future of OLE: Chicago and Cairo
- General Q&A

Goals for this Class

- Introduce the essential concepts of OLE
- Avoid overdosing your with the esoteric aspects of each
- Expose you to the core technologies provided by VC++ / MFC
- Show you the resources that are available
- Leave you ready to start building OLE applications of your own

Goals for this Class (2)

- Have Fun!

Visual C++ and MFC: A Review

- Programming Paradigm:
  - Use AppWizard to generate application
  - Use AppStudio to modify/add resources
  - Use ClassWizard to modify application
  - Write a little code
- Using On-Line Help
- Examining the Generated Code
- Documents-Views
- Etc.

An Overview of OLE 2.0

- Benefits
- Terminology
- Features
- Object Functionality
- Compatibility with OLE 1.0
- Multiplatform Support
- OLE Opportunities

OLE Benefits

- Document-centric approach to computing

- Rich documents which provide more information
- Compound documents for organizing information
- Easier applications integration

OLE Terminology (1)

- Compound Documents (container documents)
  - documents that contain data created by multiple applications
- OLE Objects (OLE items, data items)
  - text, graphics, spreadsheets, sound, etc. that are linked/embedded in a compound document

OLE Terminology (2)

- Container (container application, client application)
  - creates/manages a compound document
- Server (server application)
  - creates data items linked/embedded in compound document
- Container-Server
  - container to some, server to others

OLE Terminology (3)

- Full server
  - can be run as a standalone application and store its own documents as disk files
- Mini-server
  - cannot be run standalone
  - can only be run via a container

- cannot store its own files
- only useful with embedded items
- Basis for a control

OLE Terminology (4)

- Object Linking
  - relates data item to compound document via a linkage
  - data item is stored by server
- Object Embedding
  - relates data item to compound document via storage
  - data item is stored by container

OLE Terminology (5)

- Open editing
  - OLE 1.0 method for invoking server to edit OLE object
  - uses separate window
- In-place Activation
  - state of OLE object activated in a container application’s compound document
- Visual Editing
  - OLE 2.0 method for invoking server to edit OLE object
  - uses same window
  - only available for embedded items

OLE Terminology (6)

- OLE automation

- lets one application (server) expose objects to be manipulated
- lets another application (client) manipulate those objects
- exposed objects consist of sets of properties (data members) and methods (member functions)
- basis for custom controls

Automation Server

- application that exposes objects for manipulation

Automation Client

- application that manipulates exposed objects

OLE Features (1)

Visual Editing

- directly activate objects in place within documents without switching to a different window

Nested Objects

- a contained object can contain other objects

Drag-Drop

- drag objects from one application window to another

Cut-Paste

- move/copy objects via the clipboard

OLE Features (2)

Component Object Model

- simplifies linking/embedding, better support for container-servers

Version Management

- objects can contain versioning information

Object Conversion

- objects can be converted for use by different applications

- Adaptable Links
  - maintains links when object is moved/copied

OLE Features (3)

- Storage-Independent links
  - embedded objects can update one another's data regardless of file-system
- Automation
  - run commands/functions in one application from another

OLE 1.0 - 2.0 Compatibility

- OLE 1.0 and OLE 2.0 applications may coexist on same system
- You may mix-match 1.0/2.0 containers/servers
- OLE 2.0 apps default to OLE 1.0 behavior when dealing with OLE 1.0 app

OLE Multiplatform Support

- Win16
- Win32
  - Windows NT Daytona 2Q-3Q '94 (OLE 2.1)
- Apple Macintosh System 7
  - Complete compatibility for compound documents to/from Win16/Win32
  - Uses AppleEvents protocol
- RISC
  - MIPS
  - Alpha

OLE Opportunities with VC++ (1)

- Turn your application into OLE container
  - Lets users link/embed server objects in your app's documents
  - Lets users use visual editing to access embedded server objects
  - Open editing still available for OLE 1.0 or linked objects

OLE Opportunities with VC++ (2)

- Turn your application into an OLE server
  - Lets users store your application's data in container documents
  - Lets app integrators combine your app with others

OLE Opportunities with VC++ (3)

- Turn your application into a container-server
  - Get the benefits of both

OLE Opportunities with VC++ (4)

- Turn your VC++ application into an Automation Server
  - Automation clients can drive your application for services
  - VB is a built-in Automation Client

MFC's OLE Support

- Containers
- Servers
  - Full Servers
  - Mini-Servers
- Automation Servers

- Based on Document-View architecture
  - COleDocument
  - CDocItem
    - COleClientItem
    - COleServerItem
  - View class will have CDocItem-derived class pointer

MFC's OLE 2.0 Support: The Good News

- Container and visual editing support
- Server and visual editing support
- OLE Automation support
- Drag-drop, Cut-Paste

MFC's OLE 2.0 Support: The Bad News

- No support for Imoniker
- IUnknown interface implemented but not exposed
- IMarshall not implemented, but used internally
- Partial Compound file support

The MFC OLE Classes

Building OLE Applications

- Building OLE Containers
- Building OLE Servers
- Building OLE Automation Servers

Building OLE Containers

- The Contain application
- Invoke AppWizard
  - Use OLE Options dialog to select “Container” option
  - Use Classes dialog to override generated class names, set document type and extension
  - Generate the application
- Try out the container

Ideas for extending the Container application

- Additional Visual Editing support
- Drag-drop support
- Embedded links

Building OLE Servers

- The OText application
- Invoking AppWizard
  - Using the OLE dialog to select Server
  - Using Classes dialog to derive view from CEditView
  - Generate the application
- Add code to serialize the contents of the view
- Add code to turn on word-wrap
- Build OText

Building OLE Automation Servers

- What is Automation?

- Benefits of Automation
- History of Automation
- Automation is like making an API Call
- Automation is *not* like making an API Call
- How Automation works under-the-hood
- OLE's IDispatch Interface
- OLE Automation the Easy Way
- Building an Automation Server with VC++: 4 Easy Steps
- Building an Automation Client with VB: 3 Easy Steps

What is Automation?

- Solution to cross-application language support for systems and application programming
- Lets applications expose functions that can be called by other applications
  - Exposed functions are 'wrappers' for variables and functions in your application
  - Exposed application variables are called properties
  - Exposed application functions are called methods
  - Application exposing automation functions is an automation server (VC++, Excel, etc.)
  - Application accessing exposed automation functions is an automation client (VB, Excel, etc.)
- Automation clients extend their own functionality by automating the functionality of the server

Benefits of Automation

- End-users can use a single macro language (VBA)
- End-users can use the same interface in disparate applications
- Developers can use their own tools
  - (provided the language/tool supports Automation)
- One application can drive another

- You can automate tasks that use multiple applications
- Anyone can write a new macro language and, as long as it supports automation, use the new language to drive automation servers
- Object-oriented: reusable code, easy integration, encapsulation

The History of Automation

- Users want a common macro language
- Microsoft originally planned to define a language and a programming environment
  - BAD IDEA!
  - You'd be restricted to one choice of language
  - You'd be restricted to one choice of tool
- Automation lets you define the commands
  - GOOD IDEA!
  - Each server exposes its functionality
  - Any client can invoke exposed functions
  - End users get their choice of tool/language (VBA, others)

Automation Is Like Making An API Call:

- Client just makes a function call
- Similar to calling an exported function
- Don't statically link to automation methods, but dynamically link to them at runtime

Automation Is *Not* Like Making An API Call:

- DLLs / APIs don't provide direct access to owner's properties or variables
- Application calling DLL must know names of DLL functions in advance
- Automation client can dynamically query server to discover methods / properties, data types and parameters

Automation Under-the-Hood

- Automation Server exposes end-user level functions through OLE interface known as IDispatch
  - IDispatch can be implemented on any OLE object
  - IDispatch is, for the most part, independent of the rest of OLE
- Automation Client uses IDispatch to:
  - Learn names of functions
  - Retrieve and check function parameters
  - Invoke functions
- IDispatch assumes each function has a unique ID

OLE's IDispatch Interface

- GetTypeInfoCount
  - Retrieves number of functions and parameters
- GetTypeInfo
  - Retrieves function and parameter names
- GetIDsOfNames
  - Maps function names to function IDs
- Invoke
  - Invokes function with a given ID

OLE Automation the Easy Way

- Create Automation Servers with Visual C++
  - Why VC++?
    - VC++ is a built-in OLE automation server

- VC++ can expose the variables/functions of any CCmdTarget-derived class

- Create Automation Clients with Visual Basic (or VBA-based application)
  - Why VB?
    - VB is a built-in OLE automation client
    - VB requires only 3 lines of code to initialize, create and invoke an object

Building an Automation Server with VC++: 4 Easy Steps

- Select "Automation Support" in AppWizard when you create your application
  - Adds OLE automation derivation and dispatch table to document class
- Build your application
- Use ClassWizard to expose any variables (properties) and functions (methods)
- Run the application once
  - Registers the exposed object(s) with Windows

Building an Automation Client with VB: 3 Easy Steps

- Add an object variable to your application
- Initialize the object via a call to CreateObject
  - Pass CreateObject the object name
- Call functions exposed by the object (go wild!)

Building an OLE Automation Server

- The AutoServ Application
- Generate the initial application
  - Invoke AppWizard
  - Use OLE Options dialog to check “Automation support”

- Use Classes dialog to override class names and set file extension
- Generate the application

Building an OLE Automation Server (2)

- Add a dialog to prompt for an initial string, position
  - Use AppStudio to create the new dialog
  - Use ClassWizard to add a new dialog class
  - Add code to create the new dialog when the document class is initialized
  - Use ClassWizard to add a WM_INITDIALOG handler and set the focus to a control
- Add a new dialog for editing the string
  - Use AppStudio to create new dialog
  - Use ClassWizard to add a new dialog class

Building OLE an Automation Server (3)

- Add a menu item to invoke the editing dialog
  - Use AppStudio to add the menu item
  - Use ClassWizard to tie menu item to a message handler and invoke the new dialog
- Add data members for the string and position to the document class
  - Modify the Serialize function in the document class to serialize the data
  - Modify the view class' OnDraw function to display the string

Building an OLE Automation Server (4)

- Trap left mouse button messages
  - Use ClassWizard to trap the WM_LBUTTONDOWN message for the view
- Add a Refresh function to the document class
- Run AutoServ

Building an OLE Automation Server (5)

- Expose the position variables as properties
- Expose the string variable as a property
- Expose the Refresh function as a method
- Create/Expose a ShowWindow function, if needed by the client

Build the OLE Automation Client in VB

- Add the object variable
- Initialize the object variable
- Call the exposed functions
- Run the VB Automation Client

The Future of Windows: Chicago and Cairo

- No Program Manager or File Manager: Explorer
- Document-centric, query-based approach
- No program groups, files or directories
- Instead: program icons, documents, folders

The Future of OLE: Cairo

- OLE-aware from the ground-up
- Every folder, document, pane, control is an OLE object
- Globally available, built-in OLE objects
- Every OLE object has exposed properties / methods
- Heavy use of OLE and OLE Automation

- OLE forms instead of dialogs
- OLE distributed object support via DCE RPC

Pointers to Sources: OLE

- Visual Basic Programmer's Journal, March-April Issue
- OLE 2.0 SDK - Microsoft
- MS VC++ 1.5 "Books On-Line"
- OLE JumpStart CD - Microsoft
- "Inside OLE 2.0" by Kraig Brockschmidt - MS Press
- Win32 Professional Developer's Conference

General Q&A