© 2014 IJIRT | Volume 1 Issue 5 | ISSN : 2349-6002

MICROSOFT FOUNDATION CLASS LIBRARY

Manisha Yadav, Nisha Thakran
IT DEPARTMENT
DCE ,GURGAON

***Abstract* - The paper is associated with the MICROSOFT Foundation CLASS LIBRARY. It is a library that wraps portions of the Windows API in C++ classes, including functionality that enables them to use a default application framework. MFC was introduced with Microsoft's C/C++ 7.0 compiler for use with 16-bit versions of Windows as an extremely thin object-oriented C++ wrapper for the Windows API.This paper also depicts the features ofhe visual c++ such as it provides C++ macros for Windows messages, exceptions, run-time type dentification (RTTI), serialization and dynamic class instantiation. This paper also shows the versions of the visual c++.**

I. INTRODUCTION

The Microsoft Foundation Class Library is an application framework for programming in Microsoft Windows. Written in C++, MFC provides much of the code necessary for managing windows, menus, and dialog boxes; performing basic input/output; storing collections of data objects; and so on. All you need to do is add your application-specific code into this framework. Given the nature of C++ class programming, it is easy to extend or override the basic functionality that the MFC framework supplies.

The MFC framework is a powerful approach that lets you build upon the work of expert programmers for Windows. MFC shortens development time; makes code more portable; provides tremendous support without reducing programming freedom and flexibility; and gives easy access to "hard to program" user-interface elements and technologies, like Active technology, OLE, and Internet programming. Furthermore, MFC simplifies database programming through Data Access Objects (DAO) and Open Database Connectivity (ODBC), and network programming through Windows Sockets. MFC makes it easy to program features like property sheets ("tab dialogs"), print preview, and floating, customizable toolbars.

II. FEATURES

MFC provided C++ macros for Windows message-handling, exceptions, run-time type identification (RTTI), serialization and dynamic class instantiation.

The macros for message-handling aimed to reduce memory consumption by avoiding gratuitous virtual table use and also to provide a more concrete structure for various Visual C++-supplied tools to edit and manipulate code without parsing the full language. The message-handling macros replaced the virtual functionmechanism provided by C++.

The macros for serialization, exceptions, and RTTI predated availability of these features in Microsoft C++ by a number of years. 32-bit versions of MFC, forWindows NT 3.1 and later Windows operating systems, used compilers that implemented the language features and updated the macros to simply wrap the language features instead of providing customized implementations, realizing upward compatibility.

III. MFC DESKTOP APPLICATIONS

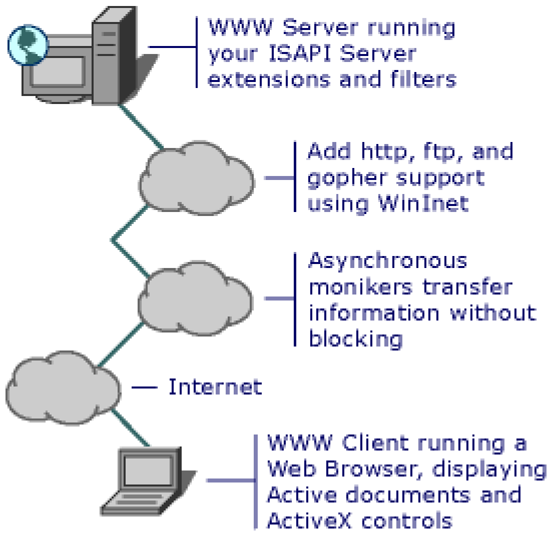
The Microsoft Foundation Class (MFC) Library provides an object-oriented wrapper over much of the Win32 and COM APIs. Although it can be used to create very simple desktop applications, it is most useful when you need to develop more complex user interfaces with multiple controls. The individual hierarchy charts included with each class are useful for locating base classes. The MFC Reference usually does not describe inherited member functions or inherited operators. For information on these functions, refer to the base classes depicted in the hierarchy diagrams.

The documentation for each class includes a class overview, a member summary by category, and topics for the member functions, overloaded operators, and data members.

Public and protected class members are documented only when they are normally used in application programs or derived classes. See the class header files for a complete listing of class members.

IV. MFC INTERNET PROGRAMMING BASICS

Microsoft provides many APIs for programming both client and server applications. Many new applications are being written for the Internet, and as technologies, browser capabilities, and security options change, new types of applications will be written. Browsers run on client computers, providing access to the World Wide Web and displaying HTML pages that contain text, graphics, ActiveX controls, and documents. Servers provide FTP, HTTP, and gopher services, and run server extension applications using CGI. Your custom application can retrieve information and provide data on the Internet**.**


WWW Server running your ISAPI Server extensions and filters
Add http, ftp, and gopher support using WinInet
Asynchronous monikers transfer information without blocking
Internet
WWW Client running a Web Browser, displaying Active documents and ActiveX controls

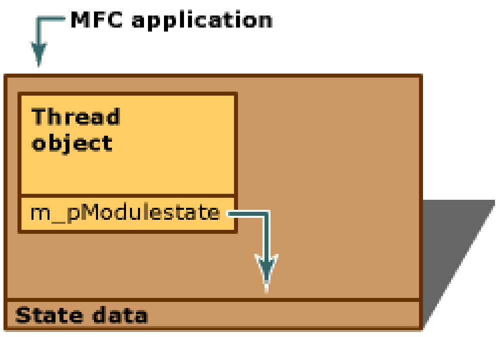
MFC provides classes that support Internet programming. You can use COleControl and CDocObjectServer and related MFC classes to write ActiveX controls and Active documents. You can use MFC classes such as CInternetSession, CFtpConnection, and CAsyncMonikerFile to retrieve files and information using Internet protocols such as FTP, HTTP, and gopher

V. MANAGING THE STATE DATA OF MFC MODULES

The state data of MFC modules and how this state is updated when the flow of execution (the path code takes through an application when executing) enters and leaves a module. Switching module states with the **AFX_MANAGE_STATE** and **METHOD_PROLOGUE** macros is also discussed.

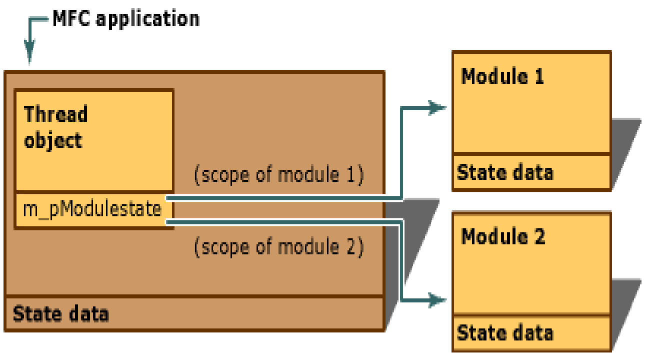
MFC has state data for each module used in an application. Examples of this data include Windows instance handles (used for loading resources), pointers to the current **CWinApp** and **CWinThread** objects of an application, OLE module reference counts, and a variety of maps that maintain the connections between Windows object handles and corresponding instances of MFC objects. However, when an application uses multiple modules, the state data of each module is not application wide. Rather, each module has its own private copy of the MFC's state data.

State Data of a Single Module (Application)



A module's state data is contained in a structure and is always available via a pointer to that structure. When the flow of execution enters a particular module, as shown in the following figure, that module's state must be the "current" or "effective" state. Therefore, each thread object has a pointer to the effective state structure of that application. Keeping this pointer updated at all times is vital to managing the application's global state and maintaining the integrity of each module's state. Incorrect management of the global state can lead to unpredictable application behavior.

State Data of Multiple Modules



In other words, each module is responsible for correctly switching between module states at all of its entry points. An "entry point" is any place where the flow of execution can enter the module's code.

VI. AUTOMATIC LINKING OF MFC LIBRARY VERSION

In versions of MFC before version 3.0 (before Visual C++ version 2.0), you had to manually specify the correct version of the MFC library in the input list of libraries for the linker. With MFC version 3.0 and later, it is no longer necessary to manually specify the version of the MFC library. Instead, the MFC header files automatically determine the correct version of the MFC library, based on values defined with **#define**, such as **_DEBUG** or **_UNICODE**. The MFC header files add **/defaultlib** directives instructing the linker to link in a specific version of the MFC library.

For example, the following code fragment from the AFX.H header file instructs the linker to link in either the NAFXCWD.LIB or NAFXCW.LIB version of MFC, depending on whether you are using the debug version of MFC:

```
#ifndef _UNICODE
#ifdef _DEBUG
#pragma comment(lib, "nafxcwd.lib")
#else
#pragma comment(lib, "nafxcw.lib")
#endif
#else
#ifdef _DEBUG
#pragma comment(lib, "uafxcwd.lib")
#else
#pragma comment(lib, "uafxcw.lib")
#endif
#endif
```

MFC header files also link in all required libraries, including MFC libraries, Win32 libraries, OLE libraries, OLE libraries built from samples, ODBC libraries, and so on. The Win32 libraries include Kernel32.Lib, User32.Lib, and GDI32.Lib.

DLLS IN MFCS

MFC libraries (DLLs) for multibyte character encoding (MBCS) are no longer included in Visual Studio, but are available as an add-on that you can download and install on any machine that has Visual Studio Professional, Visual Studio Premium, or Visual Studio Ultimate. (In Visual Studio, MFC must be enabled.) The installation requires about 440 MB of disk space and includes the English (United States) and localized versions of the DLLs.

You need this download in order to build an MFC project that has the **Character Set** property set to **Use Multi-Byte Character Set** or **Not Set**.

REFERENCES

[1] "Microsoft Visual Studio 2010 Service Pack 1". Microsoft.com. Retrieved 2012-11-19.

[2] "Microsoft Visual C++ 2010 Redistributable Package (x64". Microsoft.com. Retrieved 2012-11-19.

[3] "Microsoft Visual C++ 2010 SP1 Redistributable Package (x86)". Microsoft.com. Retrieved 2012-11-19.

[4] Visual C++ Express Overview

[5] "Visual Studio Express Edition FAQ". Microsoft.com. Retrieved 6 January 2012.

[6] "Microsoft Buys Into Inprise, Settles Disputes". Techweb.com. Retrieved 6 January 2012.

[7] Williams, Mickey; David Bennett. "Creating Your Own Message Maps". Inform IT.

[8] "Visual C++ 2008 Feature Pack shipped". Blogs.msdn.com. Retrieved 26 April 2008.