

Window Programming

Visual C++ MFC Programming

Lecture 02

김예진

Dept. of Game Software

Announcement

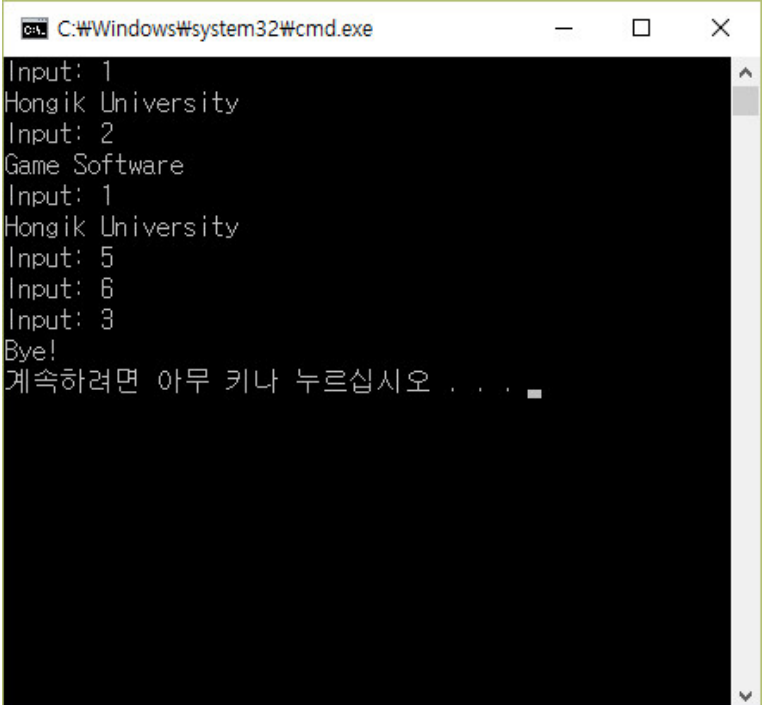
- 03/02: 501 → 502 등록

Plan

- 윈도우 콘솔(C++) 프로그램 소개
- Win32 프로그램 구조 및 소개
- MFC 프로그램 구조 및 소개

간단한 코딩 연습

- C++를 사용하여, 사용자의 입력에 따라
 - '1'을 입력하면 "Hongik University"를 출력
 - '2'를 입력하면 "Game Software"를 출력
 - '3'을 입력하면 "Bye~"를 출력하고 종료
 - 위의 과정을 무한 반복



```
C:\Windows\system32\cmd.exe
Input: 1
Hongik University
Input: 2
Game Software
Input: 1
Hongik University
Input: 5
Input: 6
Input: 3
Bye!
계속하려면 아무 키나 누르십시오 . . .
```

간단한 코딩 연습

- C++ Hints

- 출력: printf 대신 std::cout 사용

```
std::cout << "Software" << std::endl;
```

- 입력: scanf 대신 std::cin 사용

```
int i;  
std::cin >> i;
```

- 무한 반복

```
while (true)  
{  
...  
}
```

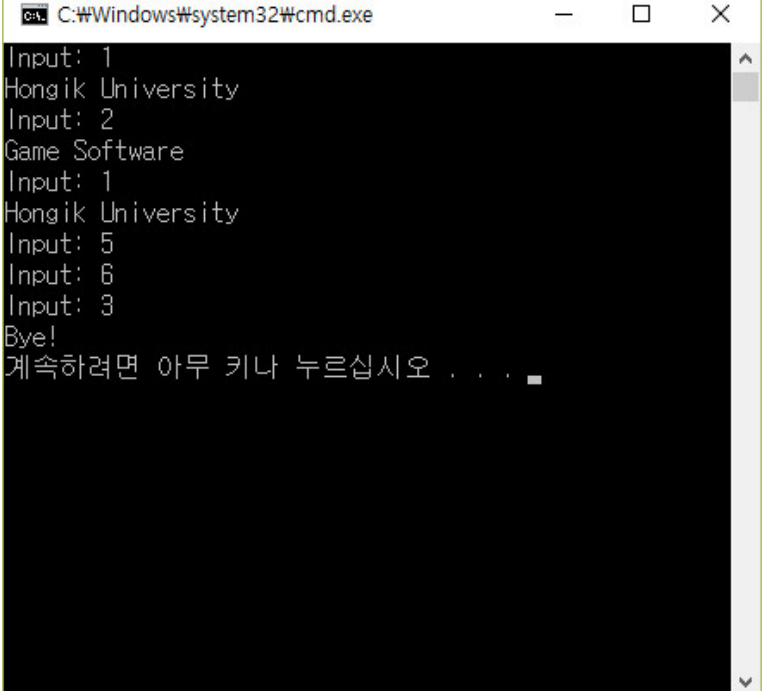
- 선택

```
switch (i)  
{  
    case 1:  
        ...  
        break;  
  
    default:  
        ...  
        Break;  
}
```

- *isostream header file 추가 필요

```
// std::cin, std::cout, std::endl 사용시  
#include <iostream>
```

.h 확장자가 없는 file은 1998년 제정된 C++ 표준을 준수하는 file임



```
C:\Windows\system32\cmd.exe  
Input: 1  
Hongik University  
Input: 2  
Game Software  
Input: 1  
Hongik University  
Input: 5  
Input: 6  
Input: 3  
Bye!  
계속하려면 아무 키나 누르십시오 . . .
```

코딩 예

```
#include <iostream>

int main()
{
    int i;
    while (true){
        std::cout << "Input: ";
        std::cin >> i;

        switch (i){
            case 1:
                std::cout << "Hongik University" << std::endl;
                break;
            case 2:
                std::cout << "Game Software" << std::endl;
                break;
            case 3:
                std::cout << "Bye!" << std::endl;
                return 0;
            default:
                break;
        }
    }
    return 0;
}
```

코딩 예

```
#include <iostream>
using namespace std;

int main()
{
    int i;
    while (true){
        cout << "Input: ";
        cin >> i;

        switch (i){
            case 1:
                cout << "Hongik University" << endl;
                break;
            case 2:
                cout << "Game Software" << endl;
                break;
            case 3:
                cout << "Bye!" << endl;
                return 0;
            default:
                break;
        }
    }
    return 0;
}
```

간단한 코딩 연습

Message (Event)

사용하여, Message (Event) Handler 따라

Windows
Procedure

- '1'을 입력하면 "Hongik University"를 출력
- '2'를 입력하면 "Game Software"를 출력
- '3'을 입력하면 "Bye~"를 출력하고 종료

- 위의 과정을 무한 반복

Message Loop

```
C:\Windows\system32\cmd.exe
Input: 1
Hongik University
Input: 2
Game Software
Input: 1
Hongik University
Input: 5
Input: 6
Input: 3
Bye!
계속하려면 아무 키나 누르십시오 . . .
```


코딩 예

```
#include <iostream>
using namespace std;

int main()
{
    int i;
    while (true){
        cout << "Input: ";
        cin >> i;
        switch (i){
            case 1:
                cout << "Hongik University" << endl;
                break;
            case 2:
                cout << "Game Software" << endl;
                break;
            case 3:
                cout << "Bye!" << endl;
                return 0;
            default:
                break;
        }
    }
    return 0;
}
```

Message
Loop

Message
Handler

Message

좀 더 멋있게...

```
int main()
{
    int i;

    while(true)
    {
        cout << "Input: ";
        cin >> i;
        procedure(i);
    }
    return 0;
}
```

```
void procedure(int msg)
{
    switch(msg)
    {
        case 1:
            cout << "Hongik University" << endl;
            break;
        case 2:
            cout << "Game Software" << endl;
            break;
        case 3:
            cout << "Bye!" << endl;
            exit(0);
            break;
        default:
            break;
    }
}
```

Win32 Program의 구조

Win32 ? (= Windows API)

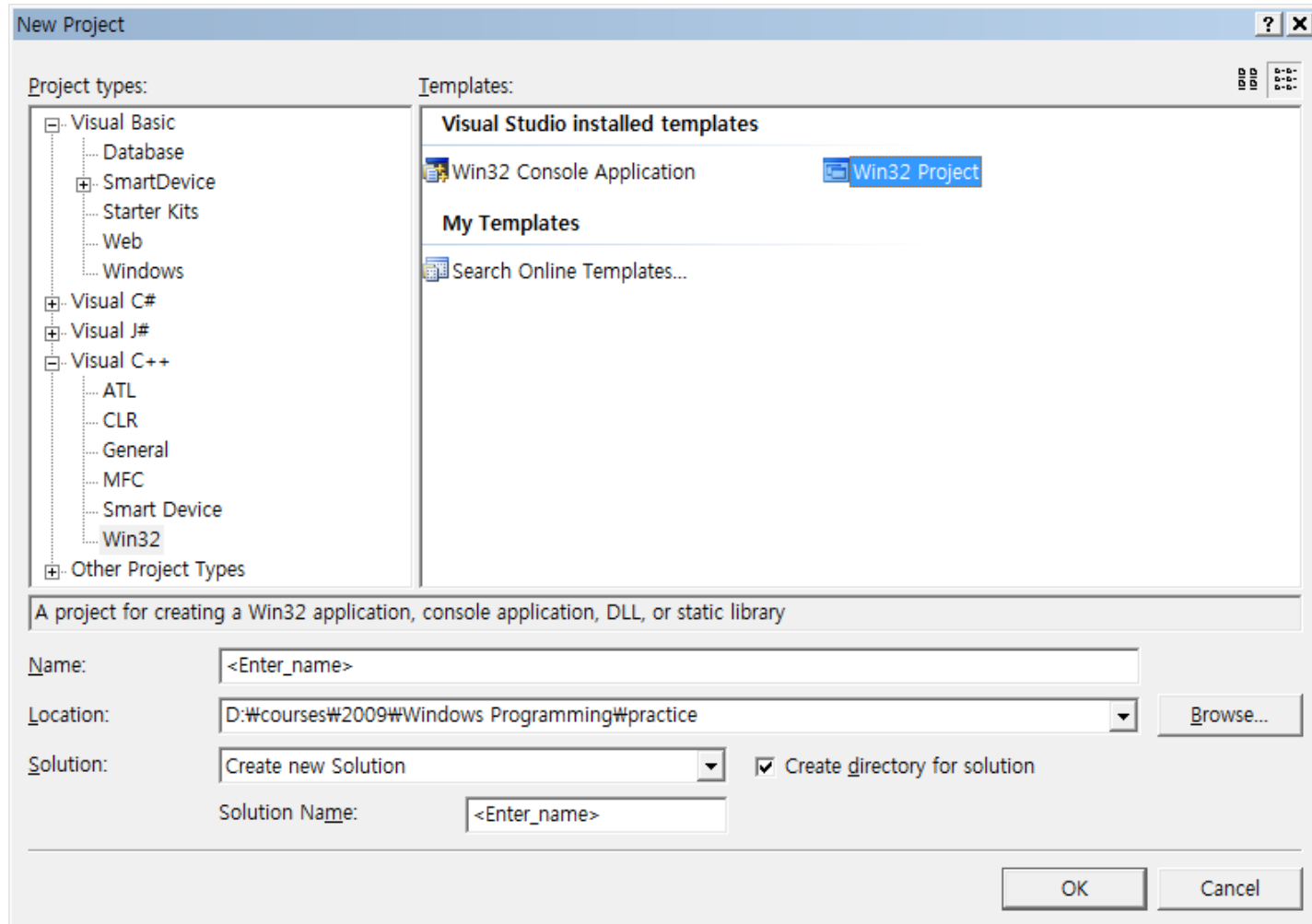
- 프로그래밍의 과정은 無에서 시작하지 않고, 다른 사람들이 잘 만들어 놓은 확장된 기능들을 이용하여 원하는 기능을 구현
- 확장된 기능(데이터 타입, 구조체, 함수들)을 모아 놓은 것을 library라고 함
 - Ex) 그림을 화면에 표시하기 위한 함수들(Library)
소리를 내기 위한 함수들(Library)

Win32 ? (= Windows API)

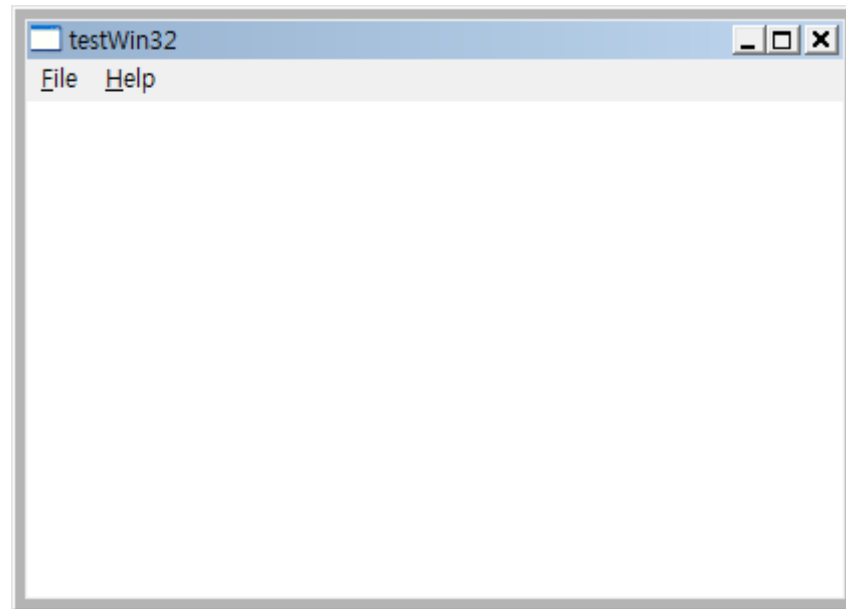
- API (Application Programming Interface)
 - 운영체제가 응용 프로그램을 위해 제공하는 각종 함수의 집합 Library의 일종
 - 주로 C 함수의 형태로 되어 있음
- Win32
 - Windows용 API의 이름
 - 즉, 윈도우에서 돌아가는 프로그램을 만들기 위한 기능들을 모아놓은 가장 기본적인 library
 - Ex) 창만들기, 버튼 달기, 메뉴만들기 등...

Win32 Project 만들어 보기

- File→New→Project → Win32 Project 선택



실행결과



Code....

```
// testWin32.cpp : Defines the entry point for the application.
//
#include "stdafx.h"
#include "testWin32.h"

#define MAX_LOADSTRING 100

// Global Variables:
HINSTANCE hInst;                                // current instance
TCHAR szTitle[MAX_LOADSTRING];                 // The title bar text
TCHAR szWindowClass[MAX_LOADSTRING];           // the main window class name

// Forward declarations of functions included in this module:
ATOM MyRegisterClass(HINSTANCE hInstance);
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow);
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
INT_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

int APIENTRY _tWinMain(HINSTANCE hInstance,
                      HINSTANCE hPrevInstance,
                      LPTSTR lpCmdLine,
                      int nCmdShow)
{
    UNREFERENCED_PARAMETER(hPrevInstance);
    UNREFERENCED_PARAMETER(lpCmdLine);

    // TODO: Place code here.
    MSG msg;
    HACCEL hAccelTable;

    // Initialize global strings
    LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
    LoadString(hInstance, IDC_TESTWIN32, szWindowClass, MAX_LOADSTRING);
    MyRegisterClass(hInstance);

    // Perform application initialization:
    if (!InitInstance(hInstance, nCmdShow))
    {
        return FALSE;
    }

    hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC_TESTWIN32));

    // Main message loop:
    while (GetMessage(&msg, NULL, 0, 0))
    {
        if (!TranslateAccelerator(hAccelTable, &msg))
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
    }

    return (int) msg.wParam;
}
```

```
//
// FUNCTION: MyRegisterClass()
//
// PURPOSE: Registers the window class.
//
// COMMENTS:
//
// This function and its usage are only necessary if you
// to be compatible with Win32 systems prior to the 'R'
// function that was added to Windows 95. It is important
// so that the application will get 'well formed' small
// with it.
//
ATOM MyRegisterClass(HINSTANCE hInstance)
{
    WNDCLASSEX wcex;

    wcex.cbSize = sizeof(WNDCLASSEX);

    wcex.style
        = CS_HREDRAW | CS_VREDRAW;
    wcex.lpfnWndProc = WndProc;
    wcex.cbClsExtra
    wcex.cbWndExtra
    wcex.hInstance
    wcex.hIcon
        = LoadIcon(hInstance, MAKEINTRESOURCE
    wcex.hCursor
    IDC_ARROW);

    wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW);
    wcex.lpszMenuName = MAKEINTRESOURCE
    wcex.lpszClassName = szWindowClass;
    wcex.hIconSm

    LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_SMALL));

    return RegisterClassEx(&wcex);
}

//
// FUNCTION: InitInstance(HINSTANCE, int)
//
// PURPOSE: Saves instance handle and creates main window
//
// COMMENTS:
//
// In this function, we save the instance handle in a global
// and create and display the main program window.
//
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
    HWND hWnd;

    hInst = hInstance; // Store instance handle in our global
    hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAP
        CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInst,
        if (!hWnd)
        {
            return FALSE;
        }

        ShowWindow(hWnd, nCmdShow);
        UpdateWindow(hWnd);

        return TRUE;
}
```

```
//
// FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)
//
// PURPOSE: Processes messages for the main window.
//
// WM_COMMAND - process the application menu
// WM_PAINT - Paint the main window
// WM_DESTROY - post a quit message and return
//
//
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    int wmId, wmEvent;
    PAINTSTRUCT ps;
    HDC hdc;

    switch (message)
    {
        case WM_COMMAND:
            wmId = LOWORD(wParam);
            wmEvent = HIWORD(wParam);
            // Parse the menu selections:
            switch (wmId)
            {
                case IDM_ABOUT:
                    MessageBox(hWnd,
                        _T("haha"), _T("about"), MB_OK);
                    DialogBox(hInst,
                        MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);
                    break;

                case IDM_EXIT:
                    DestroyWindow(hWnd);
                    break;

                default:
                    return DefWindowProc(hWnd, message, wParam, lParam);
            }
        case WM_PAINT:
            hdc = BeginPaint(hWnd, &ps);
            // TODO: Add any drawing code here...
            RECT rect;
            GetClientRect(hWnd, &rect);
            DrawText(hdc, _T("hello, Windows"),
                -1, &rect, DT_SINGLELINE|DT_CENTER|DT_VCENTER);
            EndPaint(hWnd, &ps);
            break;
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}

// Message handler for about box.
INT_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    UNREFERENCED_PARAMETER(lParam);
    switch (message)
    {
        case WM_INITDIALOG:
            return (INT_PTR)TRUE;

        case WM_COMMAND:
            if (LOWORD(wParam) == IDOK ||
                LOWORD(wParam) == IDCANCEL)
            {
                EndDialog(hDlg, LOWORD(wParam));
                return (INT_PTR)FALSE;
            }
            break;
    }
}
```


Code in short

```
int APIENTRY _tWinMain(...)
```

```
{
    // Perform application initialization:
    if (!InitInstance (hInstance, nCmdShow))
    {
        return FALSE;
    }
    // Main message loop: 메시지 큐 → 메시지 가져옴
    while (GetMessage(&msg, NULL, 0, 0))
    {
        if (!TranslateAccelerator(...))
        {
            TranslateMessage(&msg); 키보드 메시지 → 문자
            DispatchMessage(&msg); WndProc()으로 메시지 보냄
        }
    }
    return (int) msg.wParam;
}
```

```
BOOL InitInstance(...)
```

```
{
    hWnd = CreateWindow(...);
    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);

    return TRUE;
}
```

```
LRESULT CALLBACK WndProc(...)
```

```
{
    switch (message)
    {
        case WM_COMMAND:
            break;
        case WM_PAINT:
            break;
        case WM_DESTROY:
            PostQuitMessage(0);
            break;
        default:
            return;
    }
    return 0;
}
```

약간 변경하기... (WinProc)

- switch문 속 변경하기:

```
case IDM_ABOUT:
```

```
    MessageBox(hWnd, _T("haha"), _T("Test!"), MB_OK);
```

```
    DialogBox(hInst, MAKEINTRESOURCE(IDD_ABOUTBOX), hWnd, About);  
    break;
```

추가!

- 또는 switch 문 속에 아래 것 추가:

```
case WM_LBUTTONDOWN:
```

```
    MessageBox(hWnd, _T("haha"), _T("Test!"), MB_OK);
```

```
    break;
```

추가!

약간 변경하기... (WinProc)

- switch문 속 변경하기 2:

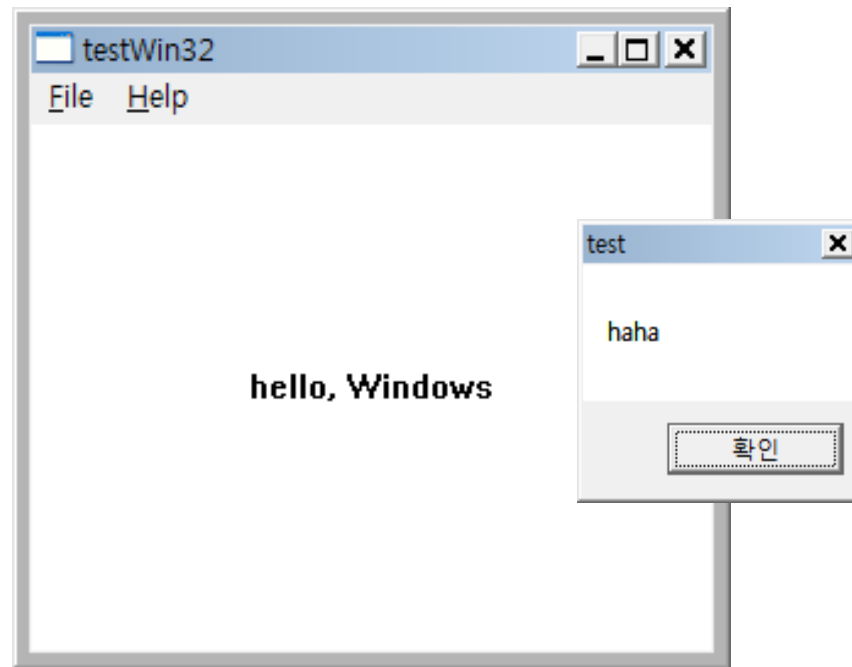
```
case WM_PAINT:  
    hdc = BeginPaint(hWnd, &ps);  
    // TODO: Add any drawing code here...
```

```
    RECT rect;  
    GetClientRect(hWnd, &rect);  
    DrawText(hdc, _T("hello, Windows"), -1, &rect,  
            DT_SINGLELINE | DT_CENTER | DT_VCENTER);
```

추가!

```
    EndPaint(hWnd, &ps);  
    break;
```

실행결과



Win32 프로그램 구조

```
WinMain(...)
{
    InitInstance(...)

    while(GetMessage (...))
    {
        DispatchMessage(...)
    }
}
```

← main 함수

← 초기화 (틀을 만들고 보여줌)

← 메시지 루프

← 메세지처리(WinProc)

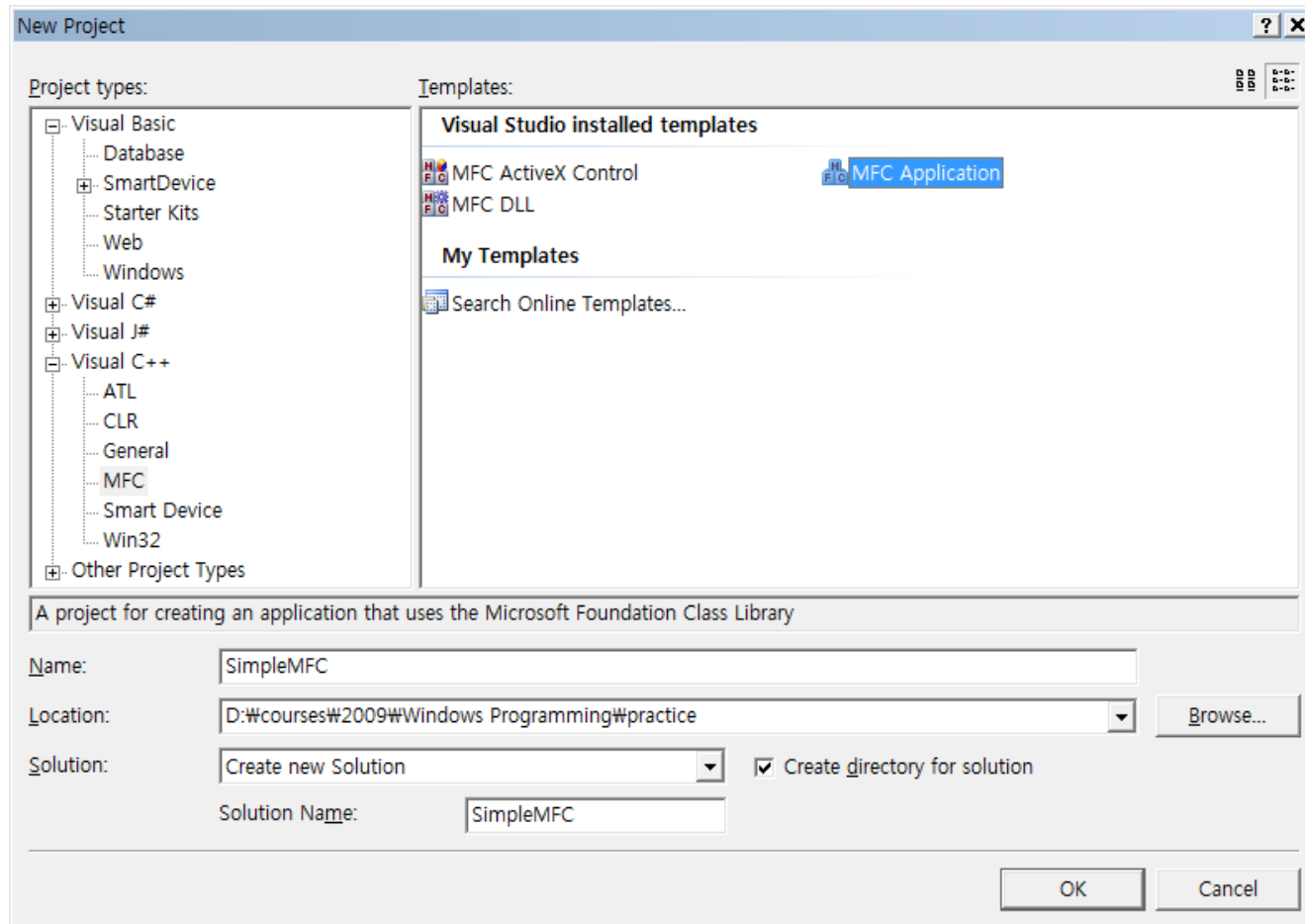
MFC 프로그램 만들어 보기

The simplest MFC application

- Single Document
- No Document/View architecture support
- No database support
- No ActiveX control
- No Docking toolbar
- No Initial status bar

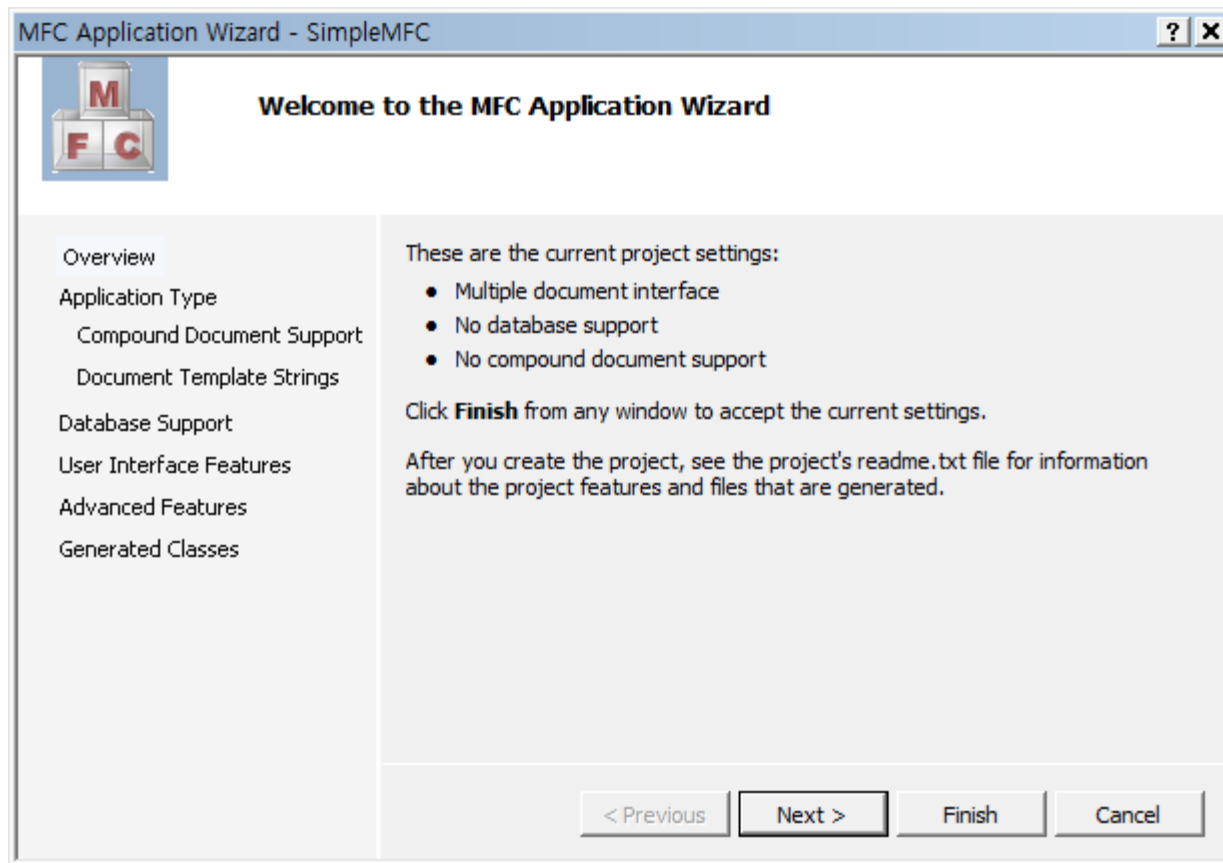
MFC 응용 프로그램 생성 (1/8)

- 프로젝트 종류 선택



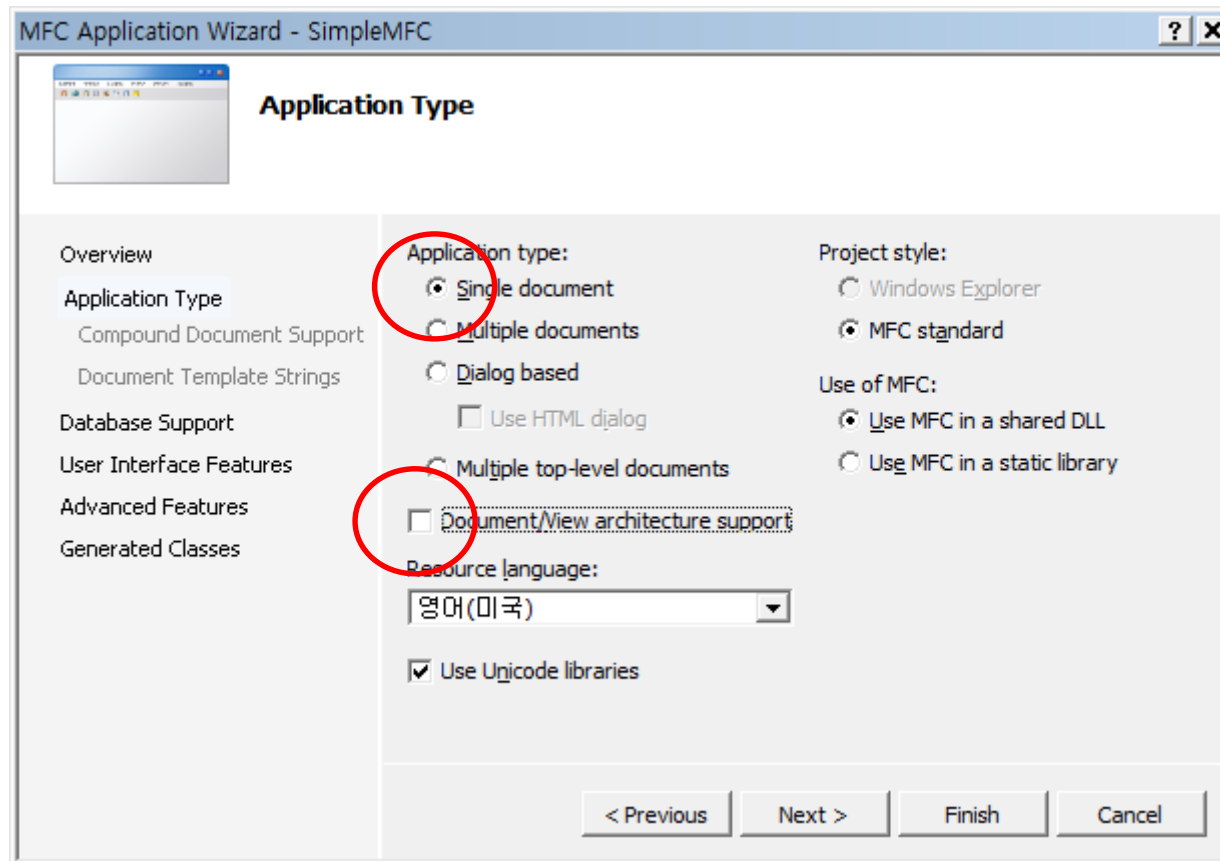
MFC 응용 프로그램 생성 (2/8)

- AppWizard 1단계



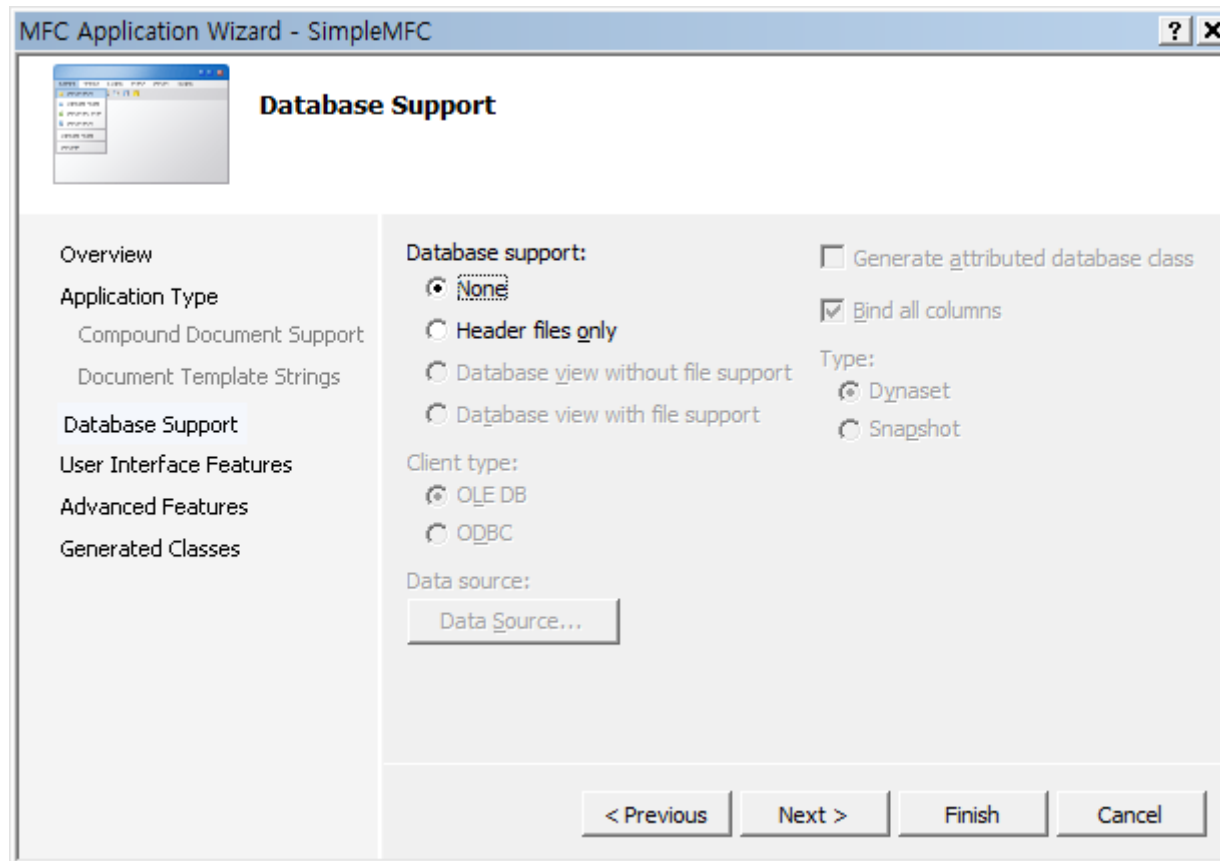
MFC 응용 프로그램 생성 (3/8)

- AppWizard 2단계



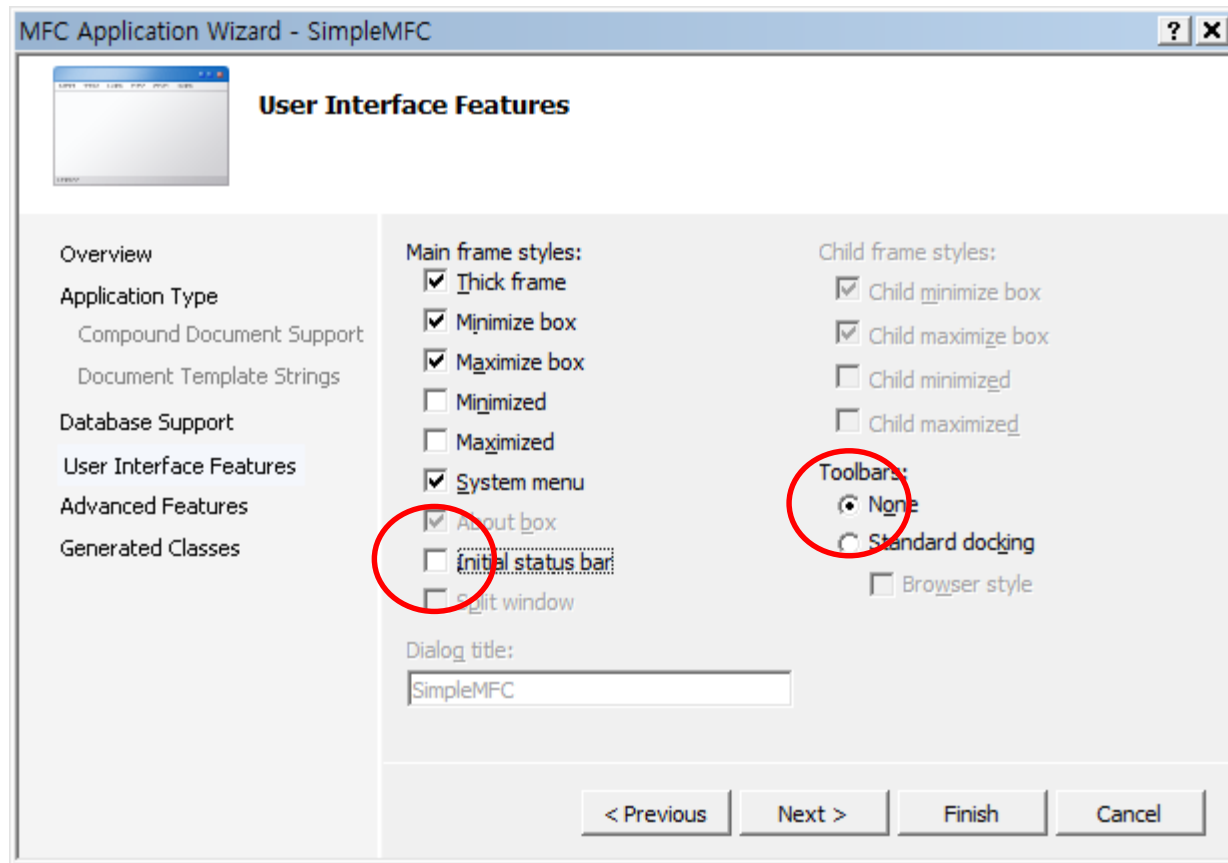
MFC 응용 프로그램 생성 (4/8)

- AppWizard 3단계



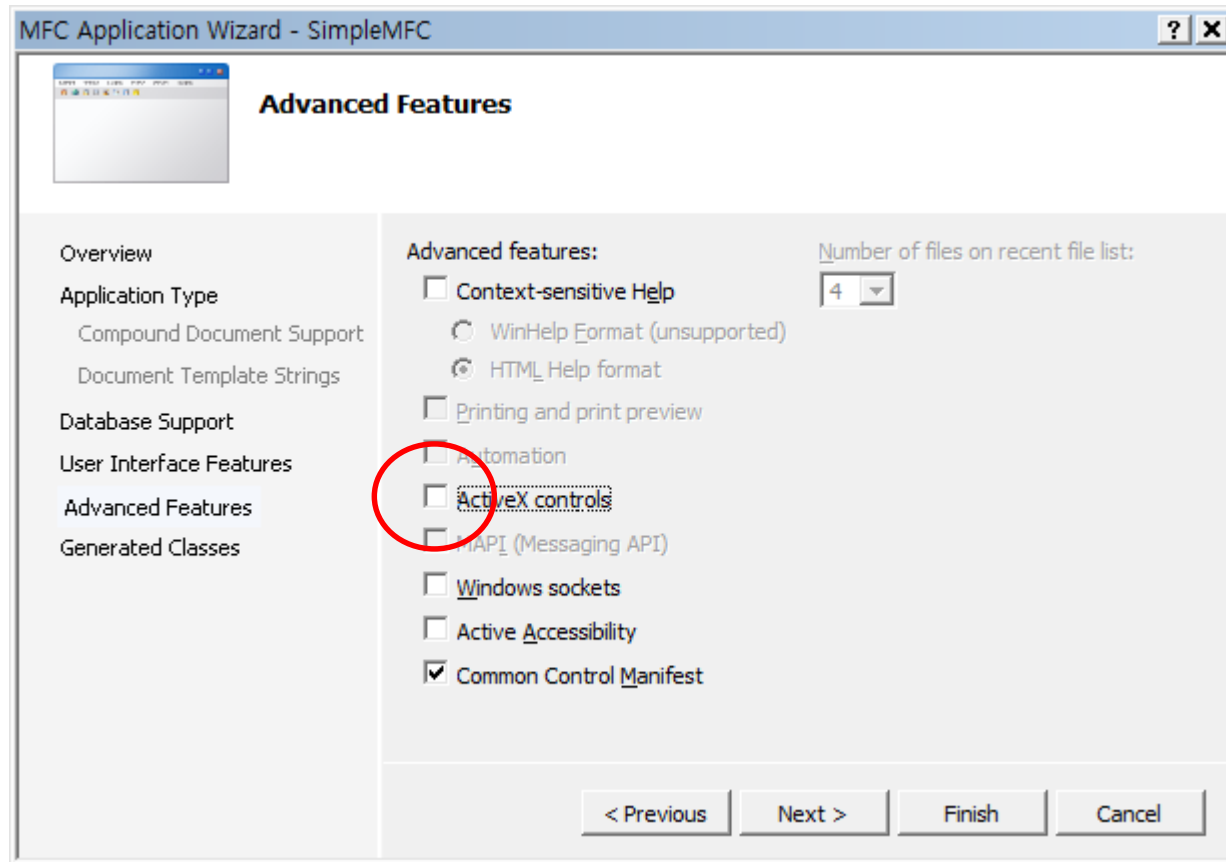
MFC 응용 프로그램 생성 (5/8)

- AppWizard 4단계



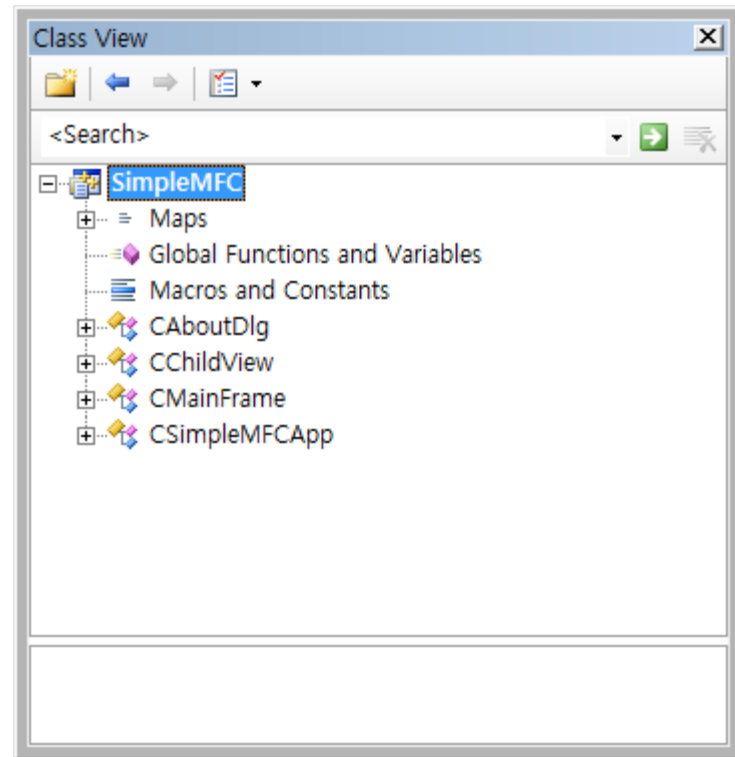
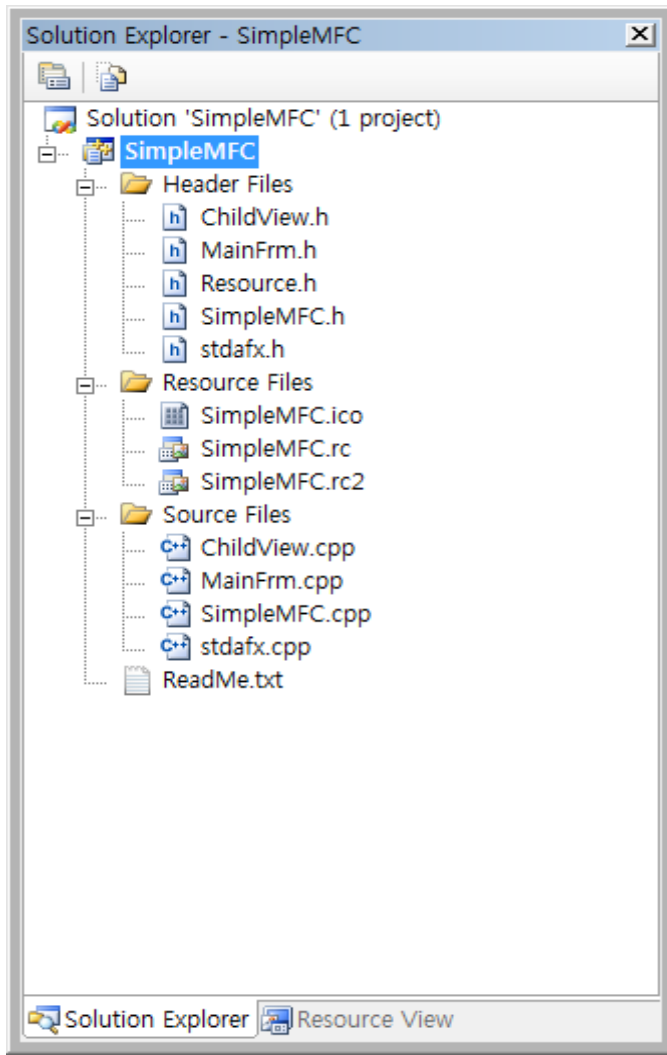
MFC 응용 프로그램 생성 (6/8)

- AppWizard 5단계



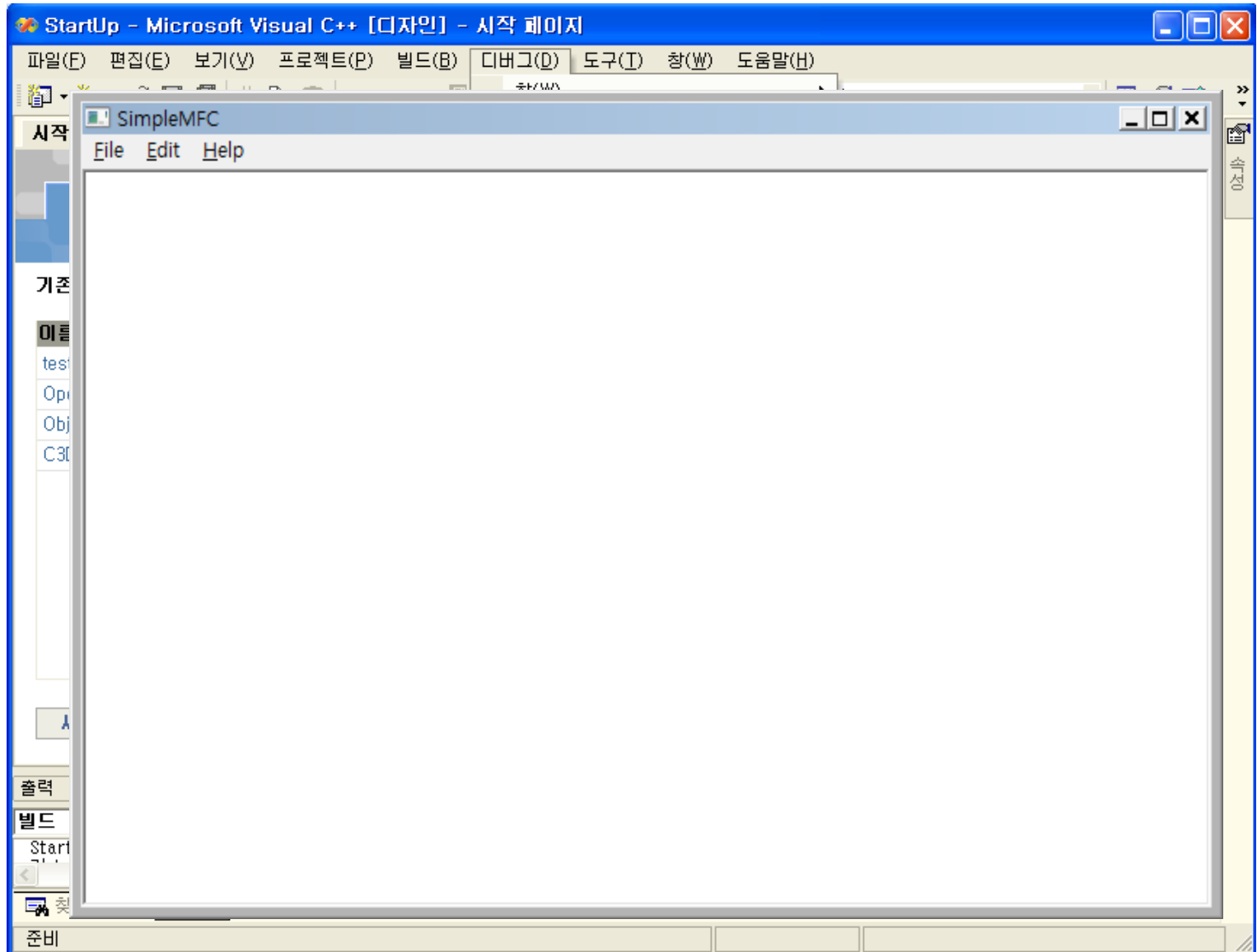
MFC 응용 프로그램 생성 (7/8)

- 생성된 프로젝트와 클래스들



MFC 응용 프로그램 생성 (8/8)

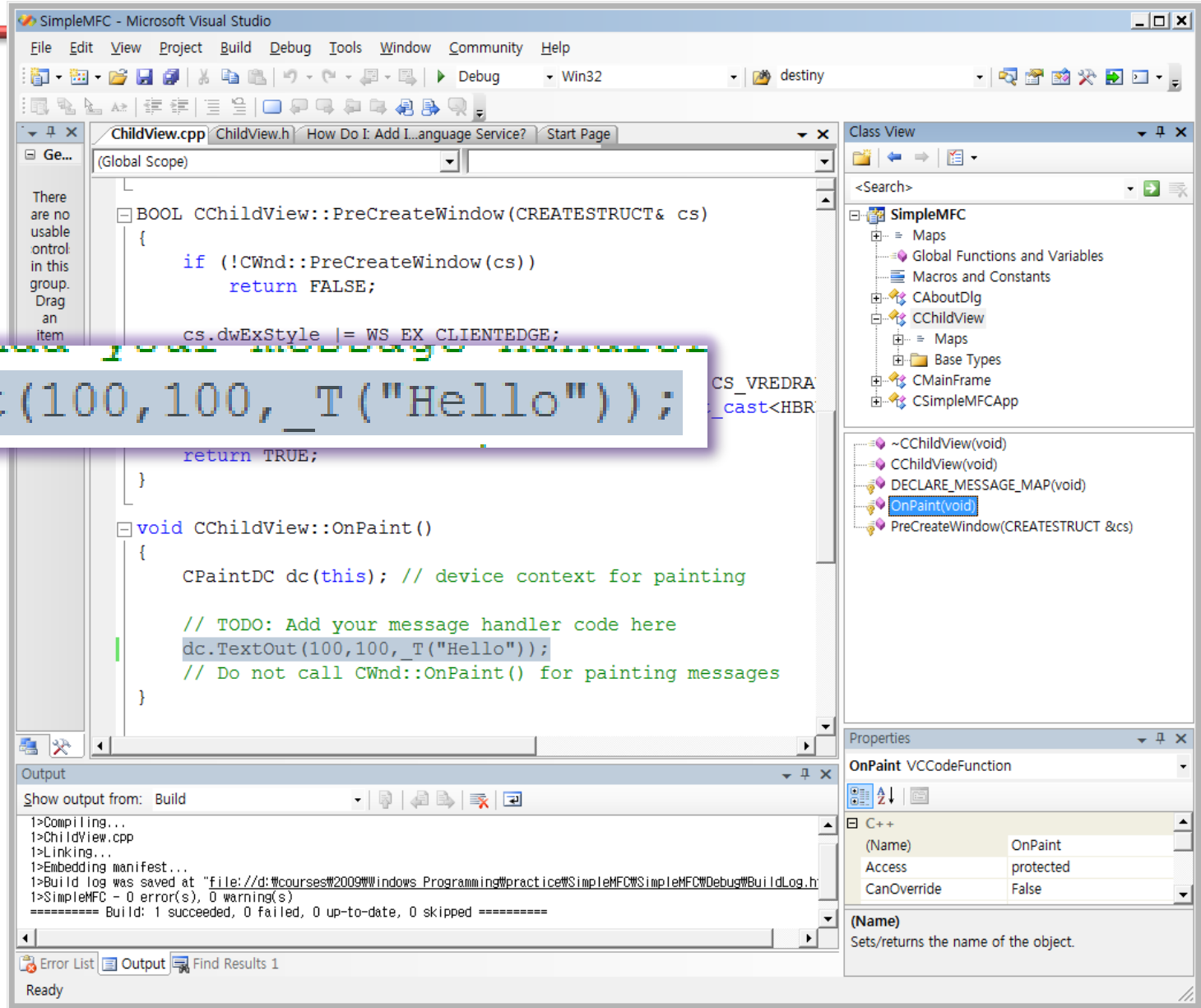
- 실행



MFC 응용 프로그램 작성

- 코드의 변경

```
dc.TextOut(100,100,_T("Hello"));
```



정리 - Application Wizard

- AppWizard 기능
 - 만들고자 하는 기본적인 프로젝트를 생성 해주고 그 안에 필요한 클래스 생성
 - 클래스에 기본적인 내용을 코딩
 - 기본적인 코딩시간을 절약 하므로 빠른 프로젝트 완성
 - AppWizard사용 도중 실수로 옵션을 선택하지 않았을 경우 소스에서 새로 추가할 수 있음

정리 - Project Workspace

- 프로젝트 워크스페이스의 구성
 - MFC의 클래스를 상속 받아 탄생된 새로운 클래스
 - 클래스 소스가 설정되어 있는 파일들
 - 소스파일: *.cpp
 - 헤더 파일: *.h
 - 프로그램에 필요한 메뉴, 아이콘, 문자열, 대화상자 같은 자원

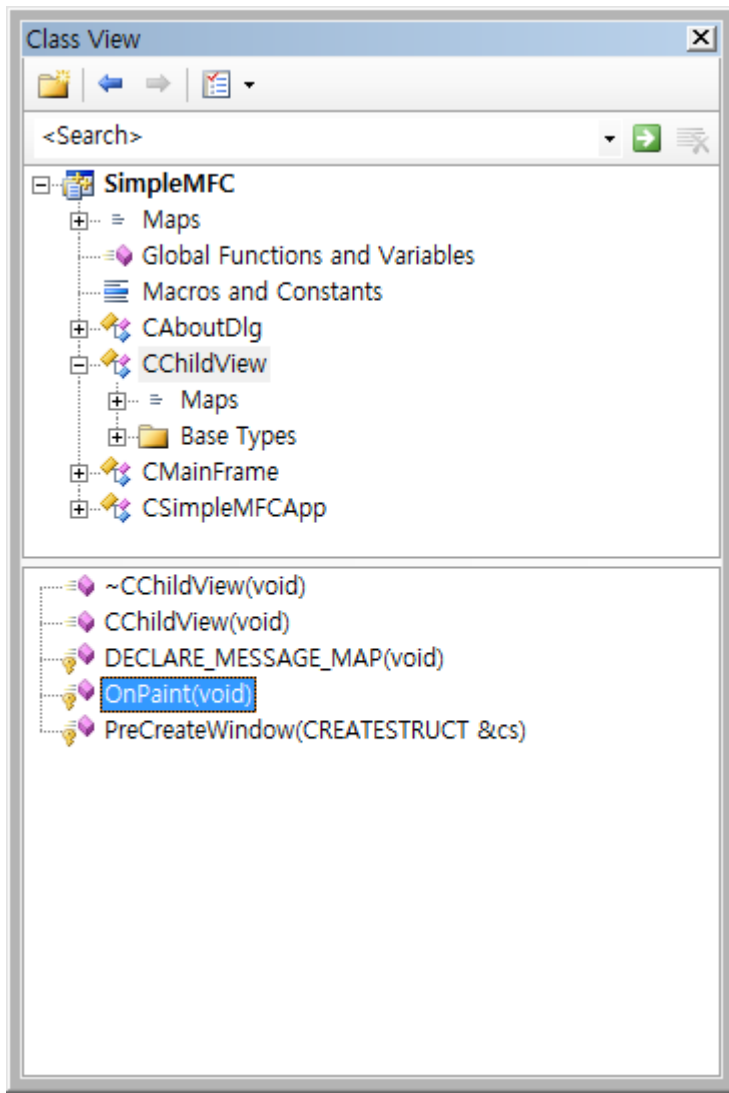
정리 - Project Workspace

- 프로젝트 워크스페이스의 항목별 설명

항 목	내 용
ClassView	프로젝트에 설정되어 있는 클래스별로 출력, 해당 항목을 선택하면 수정 가능
ResourceView	프로젝트에 설정되어 있는 메뉴, 대화 상자, 문자열, 아이콘, 비트맵 등 자원의 리스트 출력, 해당 항목 선택 수정 가능
Solution Explore (=FileView)	프로젝트에 설정되어 있는 파일 리스트 출력, 해당 항목을 선택하여 수정 가능

정리 - Project Workspace

- Class View 화면

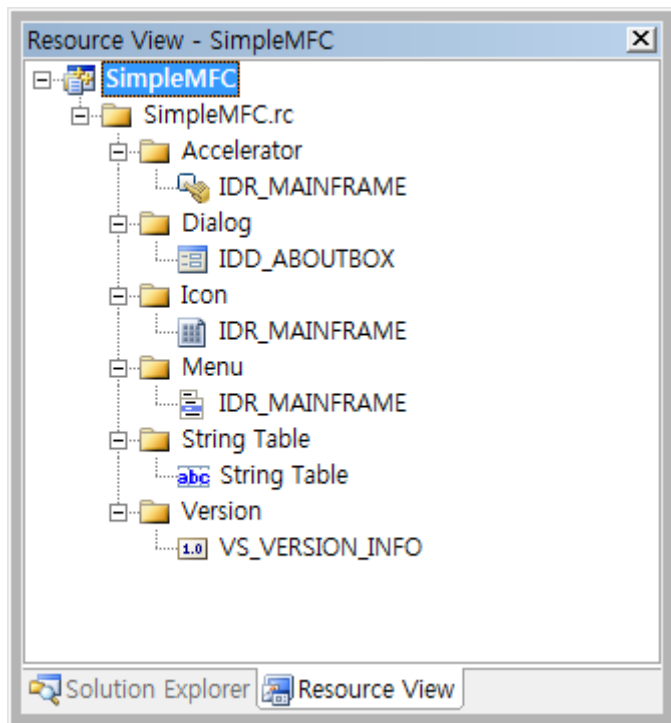


- 해당 항목을 더블 클릭하면 클래스 헤더가 나타나고 우측 버튼을 클릭하면 해당 클래스에 함수나 변수, 이벤트를 설정하도록 메뉴 등장
- 해당 클래스의 멤버 함수와 멤버 변수의 리스트
- +버튼을 클릭한 상태에서 해당 항목(변수)을 클릭하면 해당 항목이 설정되어 있는 소스 파일로 이동
- protected 형태로 설정되어 있을 경우(열쇠)
- protected 형태로 설정되어 있지 않을 경우는 열쇠 아이콘이 나타나지 않음

정리 - Project Workspace

- Resource View 화면

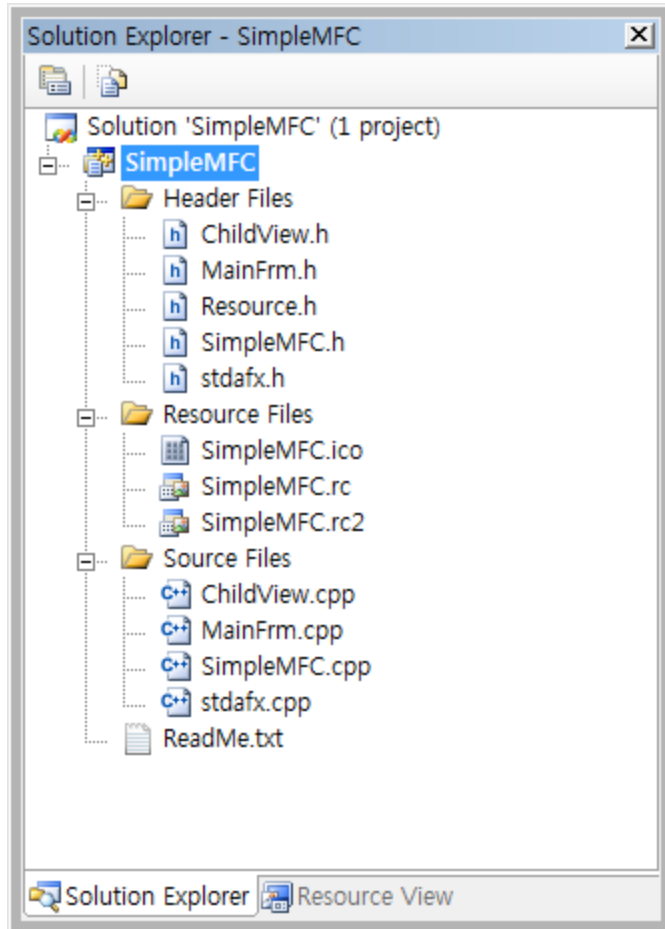
- Resource : 윈도우 프로그램을 만들 때 필요한 여러 자원



- 엑셀레이터(핫키 정의) 키값을 정의하는 항목
- 대화 상자(어떤 형태의 대화 상자의 출력할 품을 만들어서 저장) 자원들
- 아이콘 자원
- 메뉴 자원
- 문자열 테이블
- 툴바

정리 - Project Workspace

- 솔루션 탐색기(File View) 화면

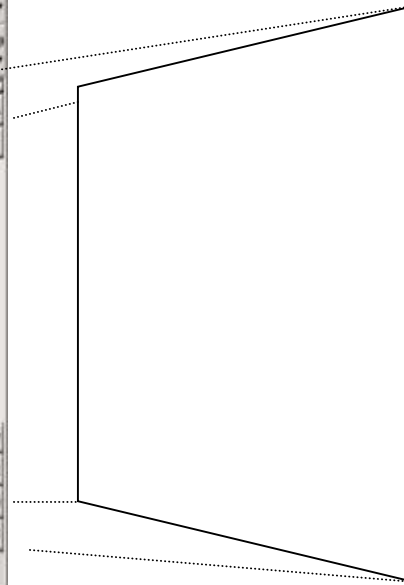
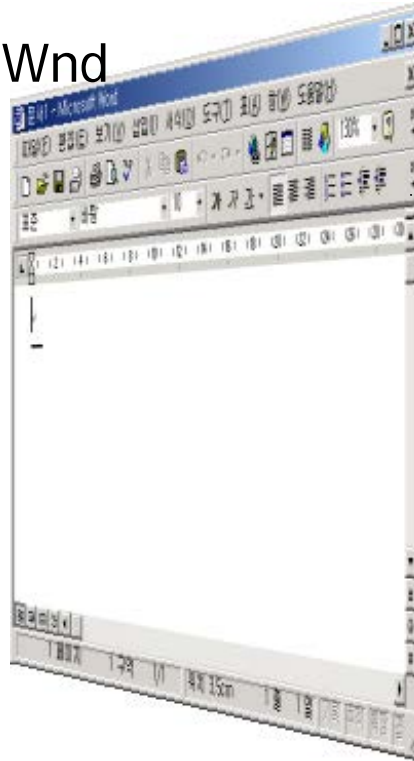


- 소스 파일: *.c, *.cpp
- 헤더 파일: *.h
- 자원 파일: *.ico, *.rc

VC++ Framework

CFrameWnd

CChildView



윈도우의 프레임(틀)을 관리

데이터를 보여주는 윈도우

CWinApp: 위의 두 오브젝트를 묶어 주고, 프로그램을 구동 시킴(눈에 안보임)
메시지 루프를 돌림

프로그램 내부 구조

theApp

(CSimpleMFC : CWinApp)

m_pMainFrame

(CMainFrame : CFrameWnd)

m_wndView

(CChildView : CWnd)

응용 프로그램 실행순서

```
CSimpleMFC theApp;
```

```
WinMain()  
{
```

// MFC 내부에 숨겨짐

```
    theApp.InitInstance();
```

// 초기화

```
    theApp.Run();
```

// 메시지 루프

```
    theApp.ExitInstance();
```

// 종료

```
}
```

응용 프로그램 클래스 (1/4)

// SimpleMFC.h

```
class CSimpleMFC : public CWinApp
{
public:
    CSimpleMFC();                // 클래스 생성자
    virtual BOOL InitInstance(); // 초기화
    afx_msg void OnAppAbout();   // About 메시지 핸들러
    DECLARE_MESSAGE_MAP()        // 1개이상의 메시지 핸들러
};
```

응용 프로그램 클래스 (2/4)

```
// SimpleMFC.cpp
```

```
#include "stdafx.h"  
#include "simpleMFC.h"  
#include "MainFrm.h"
```

MFC 기본 header file을
모아놓음



```
BEGIN_MESSAGE_MAP(CSimpleMFC, CWinApp)  
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)  
END_MESSAGE_MAP()
```

```
CSimpleMFC::CSimpleMFC()  
{  
}
```

```
CSimpleApp theApp; // 응용프로그램 자신에 해당하는 전역객체
```

응용 프로그램 클래스 (3/4)

```
BOOL CSimpleMFC::InitInstance()
{
    SetRegistryKey(_T("Local AppWizard-Generated Applications"));

    CMainFrame* pFrame = new CMainFrame;
    m_pMainWnd = pFrame;

    pFrame->LoadFrame(IDR_MAINFRAME,
        WS_OVERLAPPEDWINDOW | FWS_ADDTOTITLE, NULL,
        NULL);

    pFrame->ShowWindow(SW_SHOW);
    pFrame->UpdateWindow();

    return TRUE;
}
```

응용 프로그램 클래스 (3/4)

CFRAMEWND::LOADFRAME (MFC) - Windows Internet Explorer

http://msdn2.microsoft.com/en-us/library/e3h089yt(VS.80).aspx

파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말(H)

msdn Visual C++ Developer Center

Search MSDN with Live Search

Home Library Learn Downloads Support Community

Printer Friendly Version Add To Favorites Send Add Content...

Click to Rate and Give Feedback

MFC Library Reference

CFRAMEWND::LOADFRAME

Call to dynamically create a frame window from resource information.

This page is specific to **Microsoft Visual Studio 2005/.NET Framework 2.0**

Other versions are also available for the following:

- Microsoft Visual Studio 2003/.NET Framework 1.1
- Microsoft Visual Studio 2008/.NET Framework 3.5

```
virtual BOOL LoadFrame(
    UINT nIDResource,
    DWORD dwDefaultStyle = WS_OVERLAPPEDWINDOW | FWS_ADDTOTITLE,
    CWnd* pParentWnd = NULL,
    CCreateContext* pContext = NULL
);
```

Parameters

nIDResource

완료 인터넷 100%

응용 프로그램 실행순서

```
CSimpleApp theApp;
```

```
WinMain()  
{
```

```
// MFC 내부에 숨겨짐
```

```
    theApp.InitInstance();
```

```
// 초기화
```

```
    theApp.Run();
```

```
// 메시지 루프
```

```
    theApp.ExitInstance();
```

```
// 종료
```

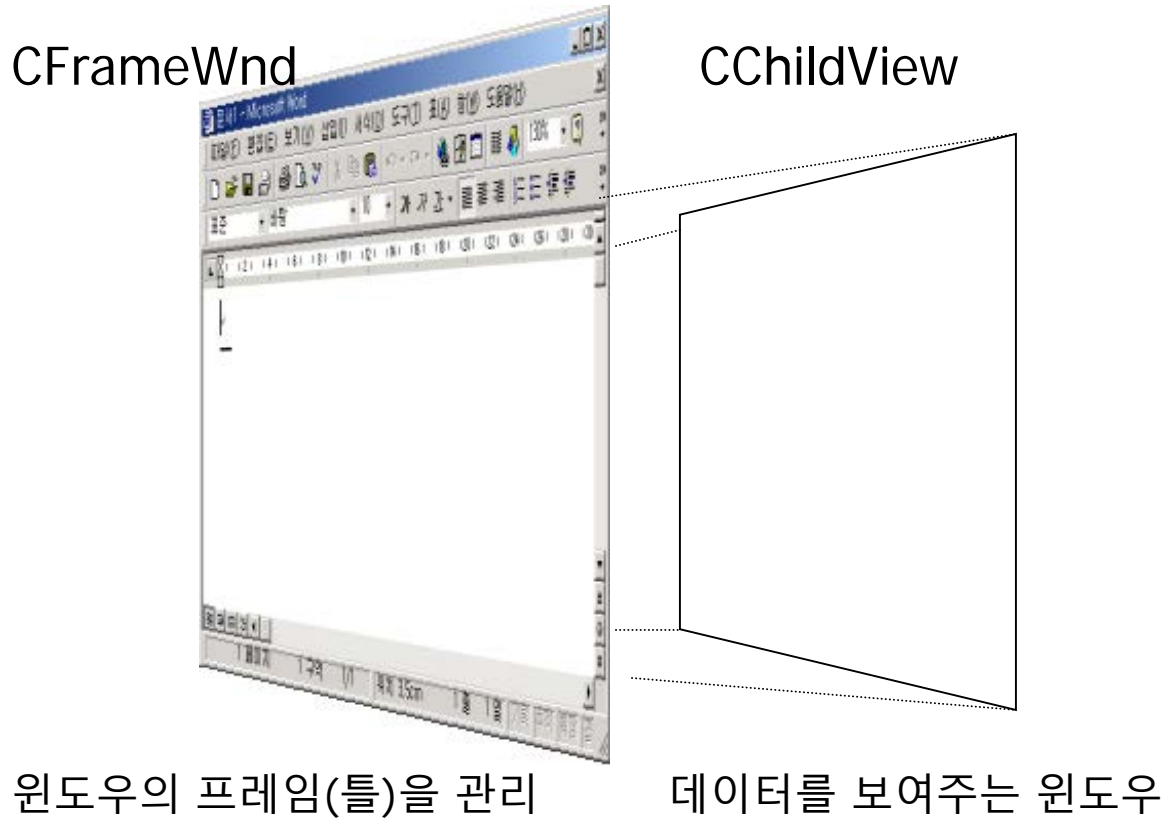
```
}
```

응용 프로그램 클래스 (4/4)

```
// 대화상자 관련 클래스 선언 및 정의 부분 - 생략  
// ...
```

```
void CSimpleApp::OnAppAbout()  
{  
    CAboutDlg aboutDlg;  
    aboutDlg.DoModal();  
}
```

VC++ Framework



CWinApp : 위의 두 오브젝트를 묶어 주고, 프로그램을 구동 시킴(눈에 안보임)
메시지 루프를 돌림

프레임 윈도우 클래스 (1/5)

// MainFrm.h

```
class CMainFrame : public CFrameWnd
{
public:
    CMainFrame();
protected:
    DECLARE_DYNAMIC(CMainFrame)
public:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    virtual BOOL OnCmdMsg(UINT nID, int nCode, void* pExtra,
        AFX_CMDHANDLERINFO* pHandlerInfo);
    virtual ~CMainFrame();
    CChildView m_wndView;
protected:
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnSetFocus(CWnd *pOldWnd);
    DECLARE_MESSAGE_MAP()
};
```

프레임 윈도우 클래스 (2/5)

```
// MainFrm.cpp
IMPLEMENT_DYNAMIC(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    ON_WM_CREATE()
    ON_WM_SETFOCUS()
END_MESSAGE_MAP()

CMainFrame::CMainFrame()
{
}

CMainFrame::~CMainFrame()
{
}
```

프레임 윈도우 클래스 (3/5)

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;
    if (!m_wndView.Create(NULL, NULL, AFX_WS_DEFAULT_VIEW,
        CRect(0, 0, 0, 0), this, AFX_IDW_PANE_FIRST, NULL))
    {
        TRACE0("Failed to create view window\n");
        return -1;
    }

    return 0;
}
```

프레임 윈도우 클래스 (4/5)

```
BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    cs.dwExStyle &= ~WS_EX_CLIENTEDGE;
    cs.lpszClass = AfxRegisterWndClass(0);
    return TRUE;
}
```

프레임 윈도우 클래스 (4/5)

```
struct CREATESTRUCT{
    LPVOID lpCreateParams;
    HINSTANCE hInstance;
    HMENU hMenu;
    HWND hwndParent;
    int cy;
    int cx;
    int y;
    int x;
    LONG style;
    LPCTSTR lpszName;
    LPCTSTR lpszClass;
    DWORD dwExStyle;
};
```

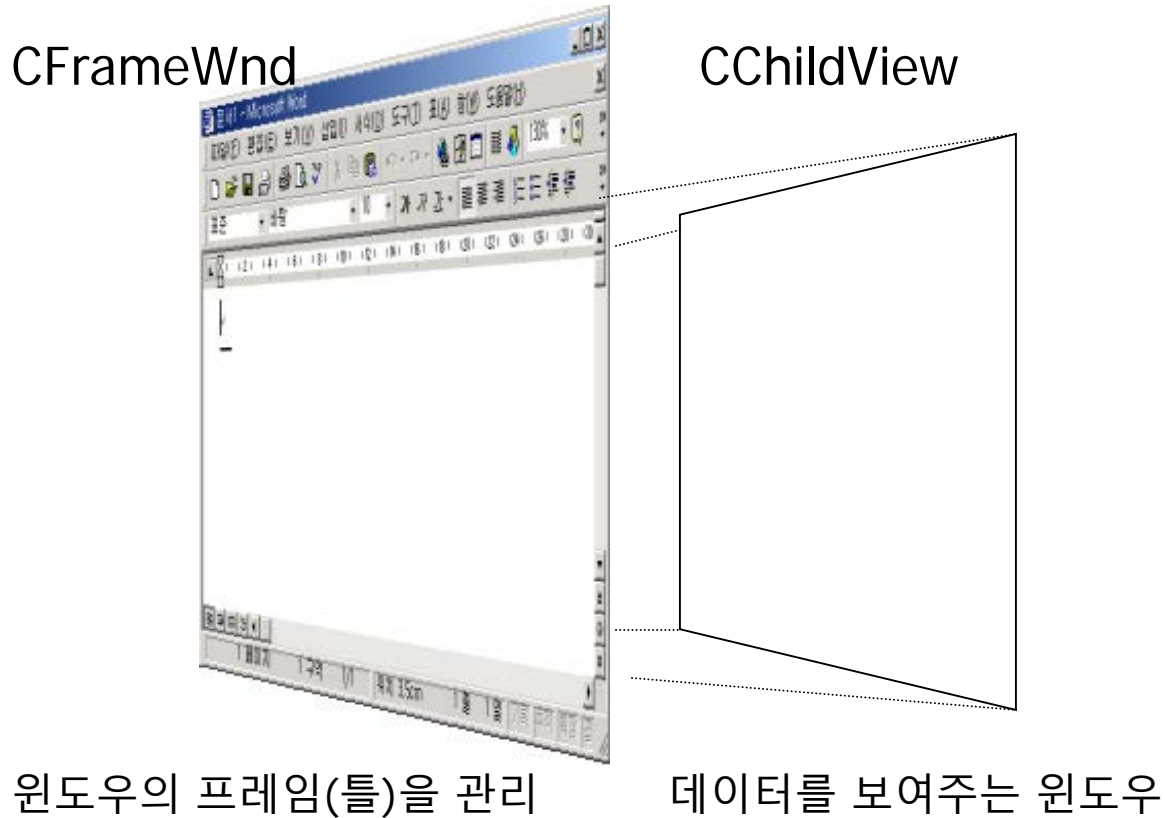
프레임 윈도우 클래스 (5/5)

```
void CMainFrame::OnSetFocus(CWnd* pOldWnd)
{
    m_wndView.SetFocus();
}
```

```
BOOL CMainFrame::OnCmdMsg(UINT nID, int nCode, void*
    pExtra, AFX_CMDHANDLERINFO* pHandlerInfo)
{
    if (m_wndView.OnCmdMsg(nID, nCode, pExtra,
        pHandlerInfo))
        return TRUE;

    return CFrameWnd::OnCmdMsg(nID, nCode, pExtra,
        pHandlerInfo);
}
```

VC++ Framework



CWinApp : 위의 두 오브젝트를 묶어 주고, 프로그램을 구동 시킴(눈에 안보임)
메시지 루프를 돌림

뷰 클래스 (1/4)

// ChildView.h

```
class CChildView : public CWnd
{
public:
    CChildView();
protected:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
public:
    virtual ~CChildView();
protected:
    afx_msg void OnPaint();
    DECLARE_MESSAGE_MAP()
};
```


뷰 클래스 (2/4)

```
// ChildView.cpp
CChildView::CChildView()
{
}

CChildView::~~CChildView()
{
}

BEGIN_MESSAGE_MAP(CChildView,CWnd )
    ON_WM_PAINT()
END_MESSAGE_MAP()
```

뷰 클래스 (3/4)

```
BOOL CChildView::PreCreateWindow(CREATESTRUCT& cs)
{
    if (!CWnd::PreCreateWindow(cs))
        return FALSE;

    cs.dwExStyle |= WS_EX_CLIENTEDGE;
    cs.style &= ~WS_BORDER;
    cs.lpszClass = AfxRegisterWndClass (
        CS_HREDRAW|CS_VREDRAW|CS_DBLCLKS,
        ::LoadCursor(NULL, IDC_ARROW),
        HBRUSH(COLOR_WINDOW+1), NULL);

    return TRUE;
}
```

Style: [http://msdn2.microsoft.com/en-us/library/ms632600\(VS.85\).aspx](http://msdn2.microsoft.com/en-us/library/ms632600(VS.85).aspx)

ExStyle: [http://msdn2.microsoft.com/en-us/library/ms632680\(VS.85\).aspx](http://msdn2.microsoft.com/en-us/library/ms632680(VS.85).aspx)

System Color: <http://msdn2.microsoft.com/en-us/library/ms724371.aspx>

뷰 클래스 (4/4)

```
void CChildView::OnPaint()
{
    CPaintDC dc(this);
    dc.TextOut(100, 100, _T("안녕하세요."));
}
```

요약

클래스 종류	베이스 클래스 이름	핵심 함수 - 주 역할
응용 프로그램 클래스	CWinApp	InitInstance() - 프레임 윈도우를 생성한다. Run() - 메시지 루프를 제공한다.
프레임 윈도우 클래스	CFrameWnd	OnCreate() - 뷰를 생성한다.
뷰 클래스	CWnd	OnPaint() - 화면에 출력한다.

집에 가서...

- 윈도우 프로그래밍 (개정판, 2014) 1장 읽어보기
- 특히 page 54~66

Q & A