

# Window Programming

Visual C++ MFC Programming

Lecture 02

김예진

Dept. of Game Software

# Notices

---

- 03/08: 501 (29) → 502 (18) 등록

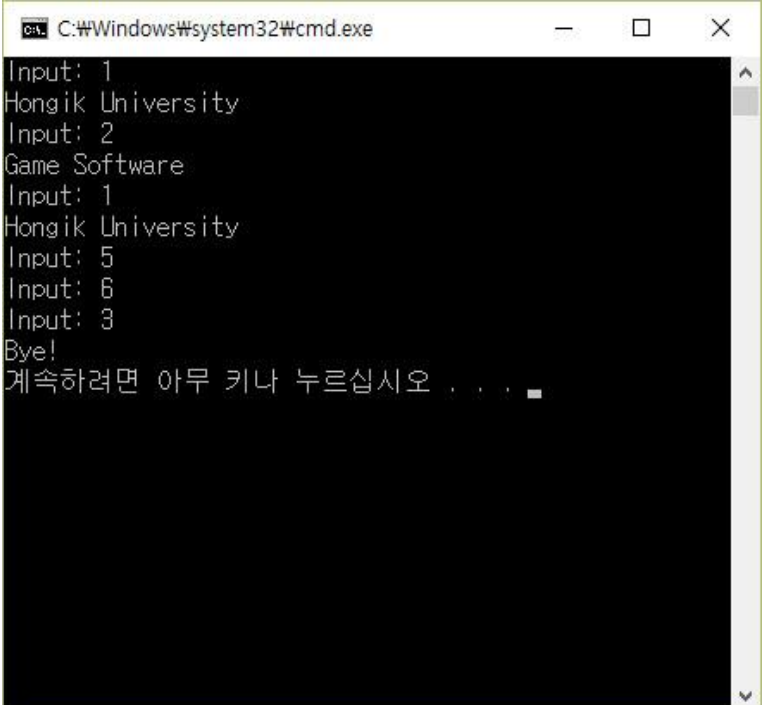
# Plan

---

- 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)  
{  
...  
}
```

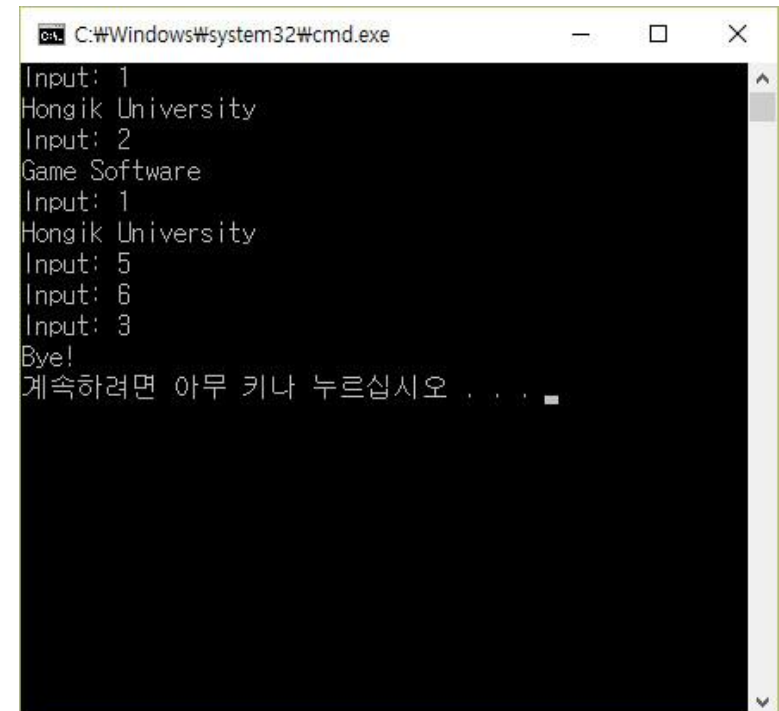
- 선택

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

- \*iostream header file 추가 필요

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

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



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window shows the output of a C++ program. The output is as follows:

```
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)

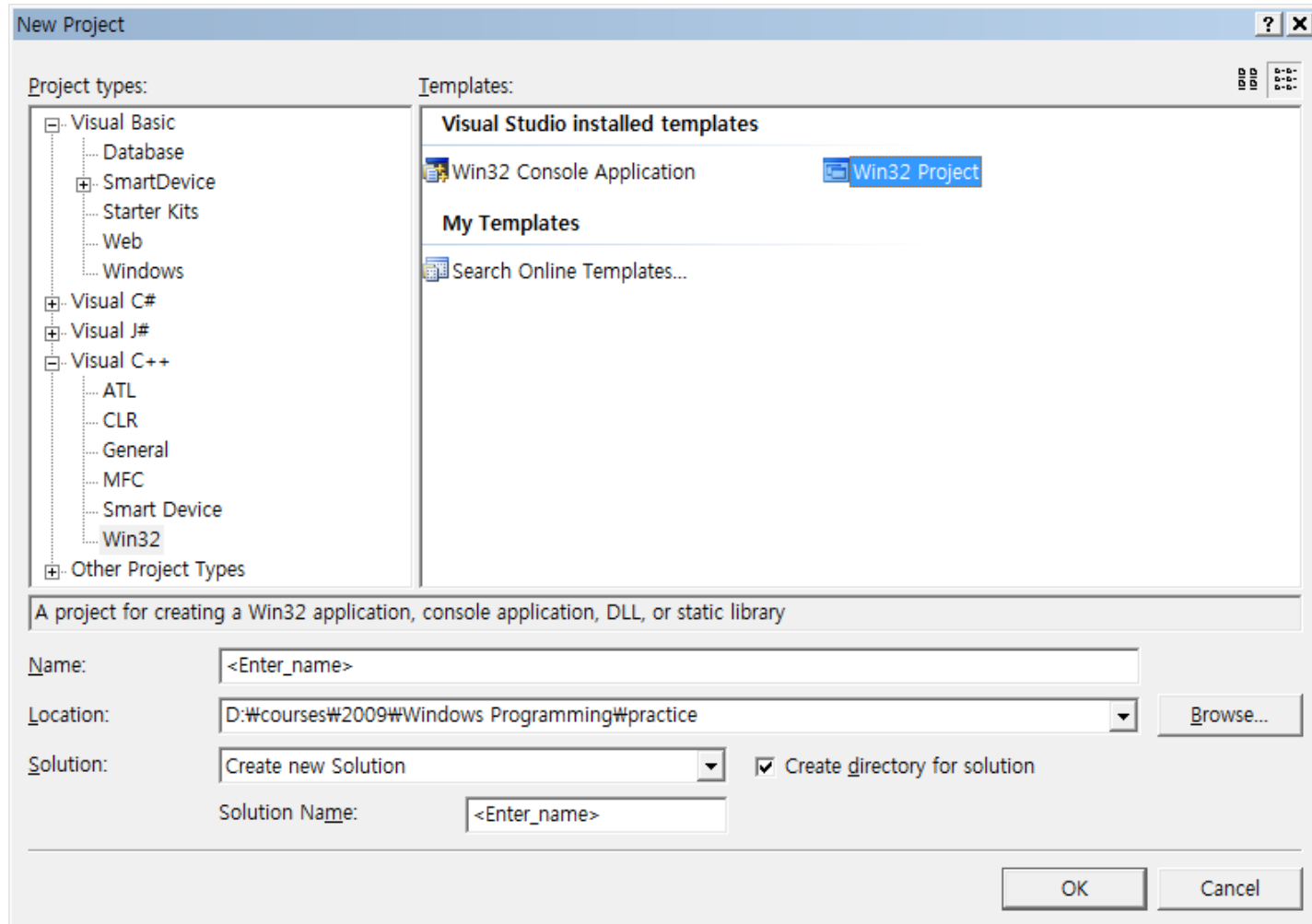
# Win32 ? (= Windows API)

---

- 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...

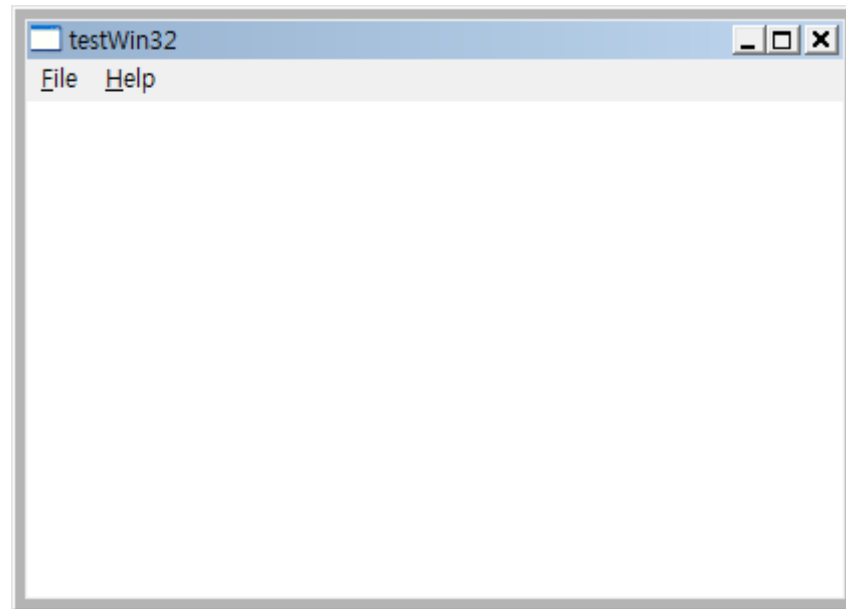
# Win32 Project 만들어 보기

- 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(hInst, 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 = 0;
    wcex.cbWndExtra = 0;
    wcex.hInstance = hInstance;
    wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_SMALL));
    wcex.hCursor = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_ARROW));
    wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW);
    wcex.lpszMenuName = MAKEINTRESOURCE(IDM_ABOUTBOX);
    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 variable hInst.

    hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPED,
        CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance,
        NULL);

    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);
            }
            break;

        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)TRUE;
            }
            break;
    }

    return (INT_PTR)FALSE;
}
```



# 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 메세지 → 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...
```

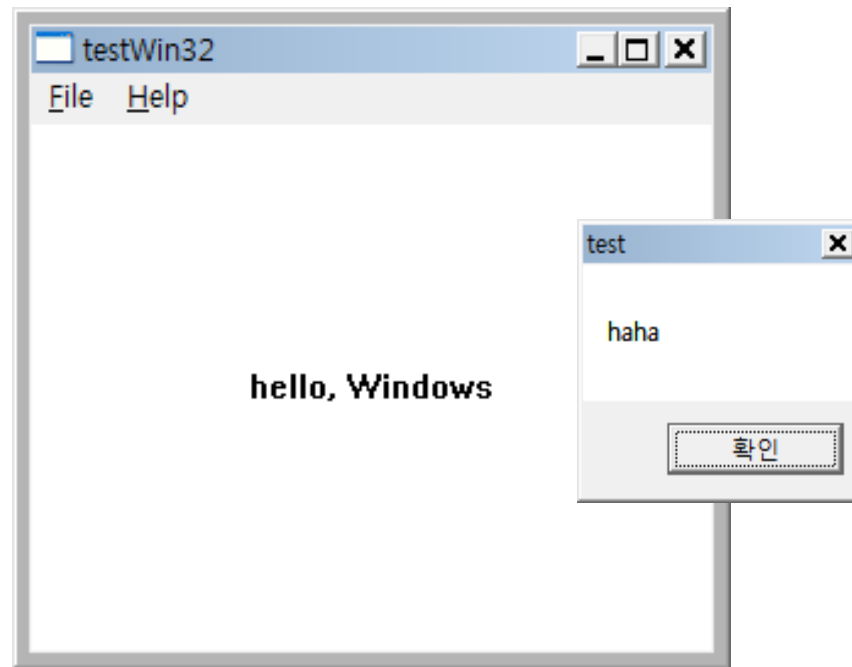
```
    RECT rect;  
    GetClientRect(hWnd, &rect);  
    DrawText(hdc, _T("hello, Windows"), -1, &rect,  
             DT_SINGLELINE|DT_CENTER|DT_VCENTER);
```

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

추가!

# 실행결과

---



# Win32 프로그램 구조

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

← main 함수

← 초기화 (틀을 만들고 보여줌)

← 메시지 루프

← 메시지 처리 (WinProc)

MFC 프로그램 만들어 보기

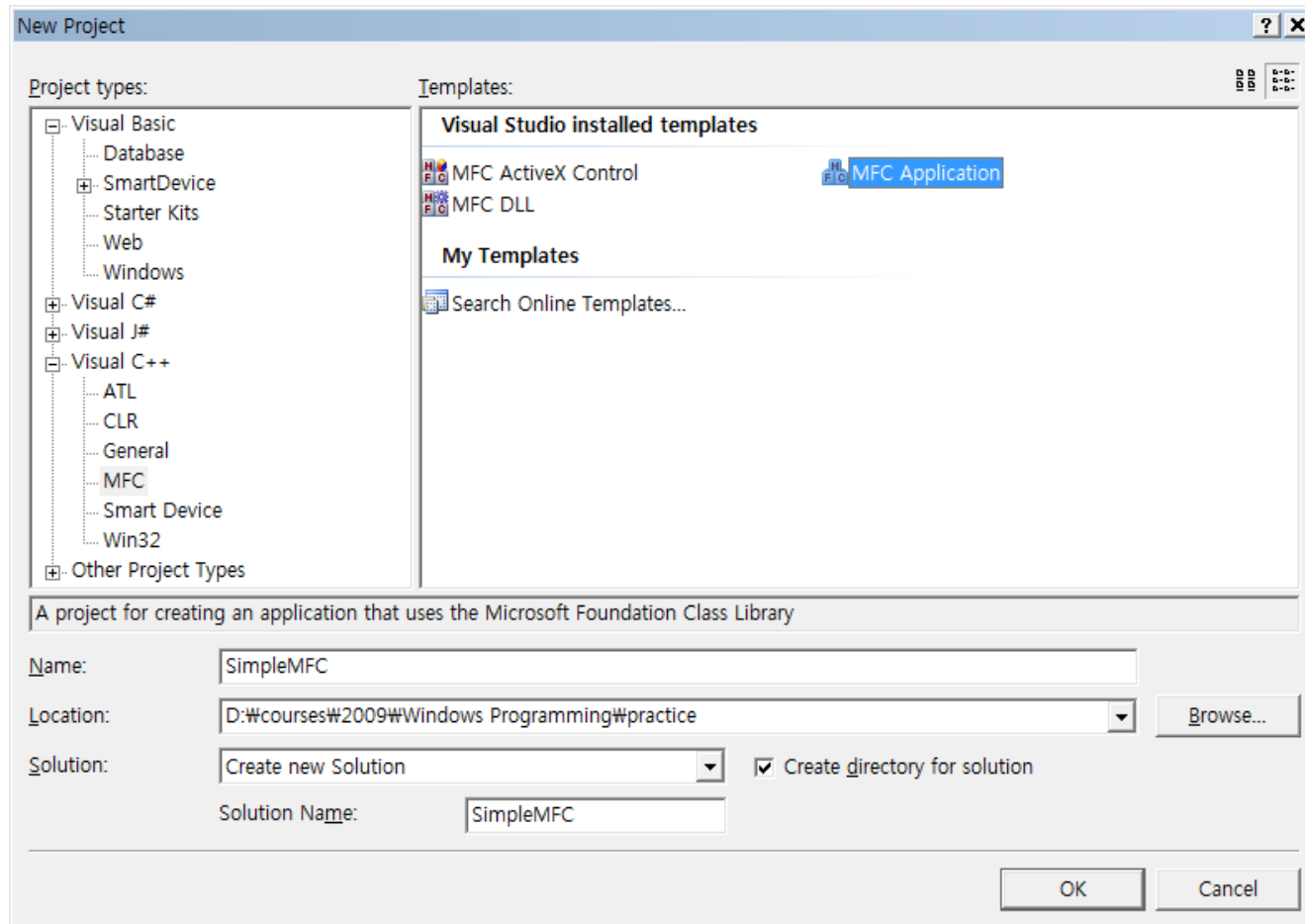
# 간단한 MFC Application

---

- 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)

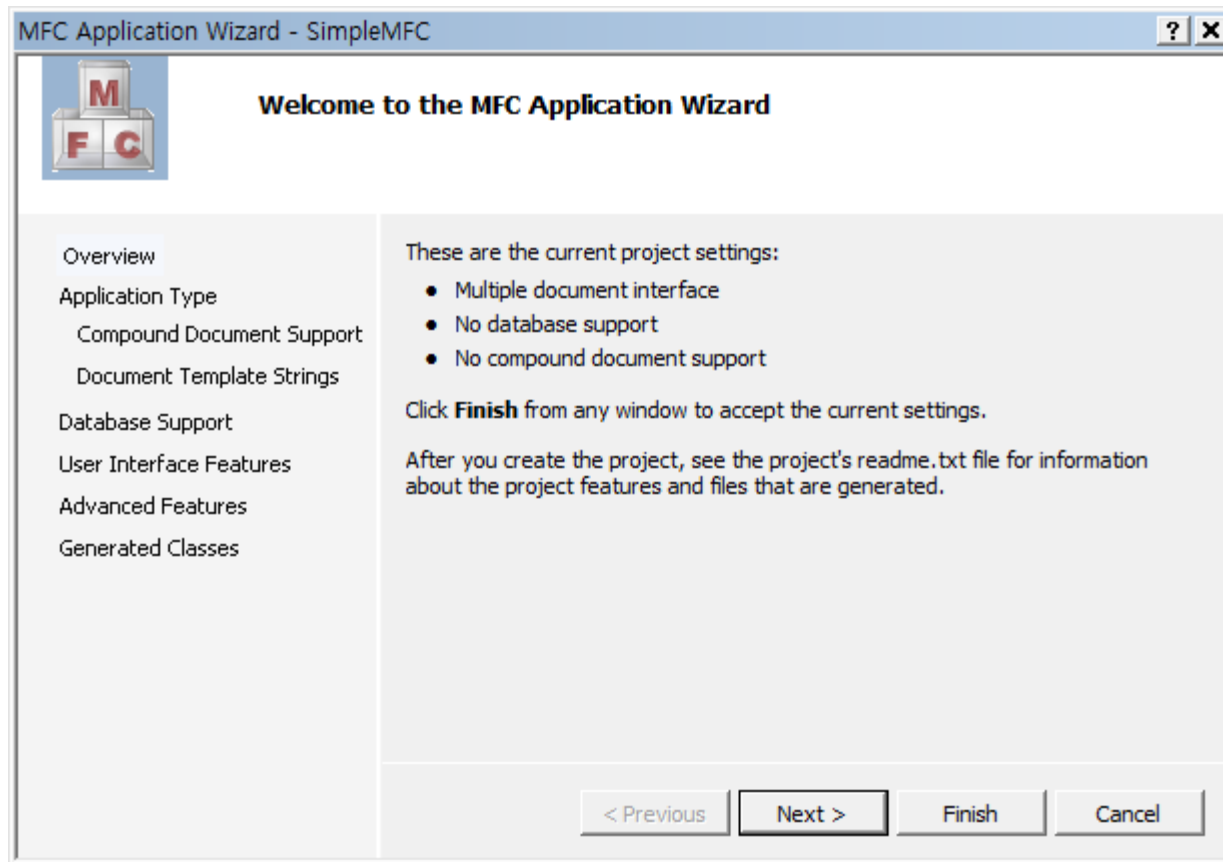
- 프로젝트 종류 선택





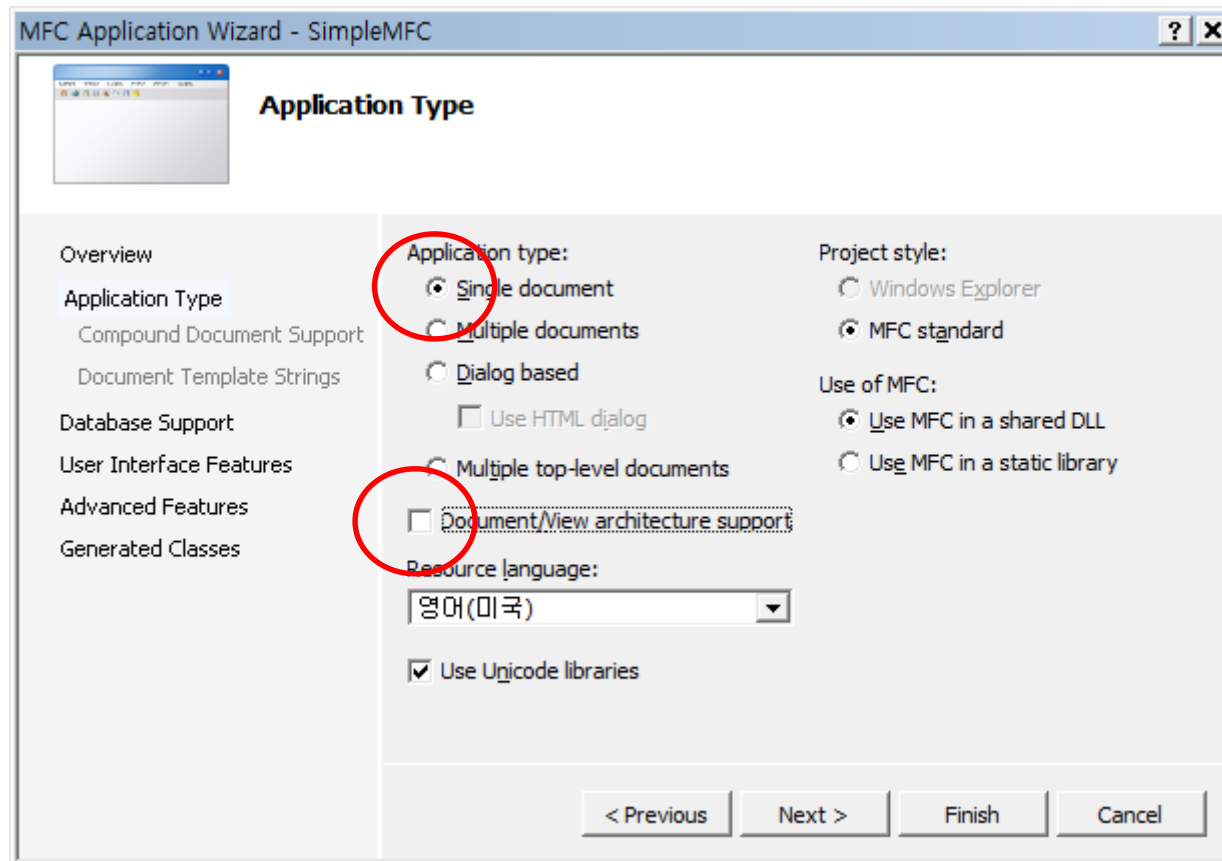
# MFC 응용 프로그램 생성 (2/8)

- AppWizard 1단계



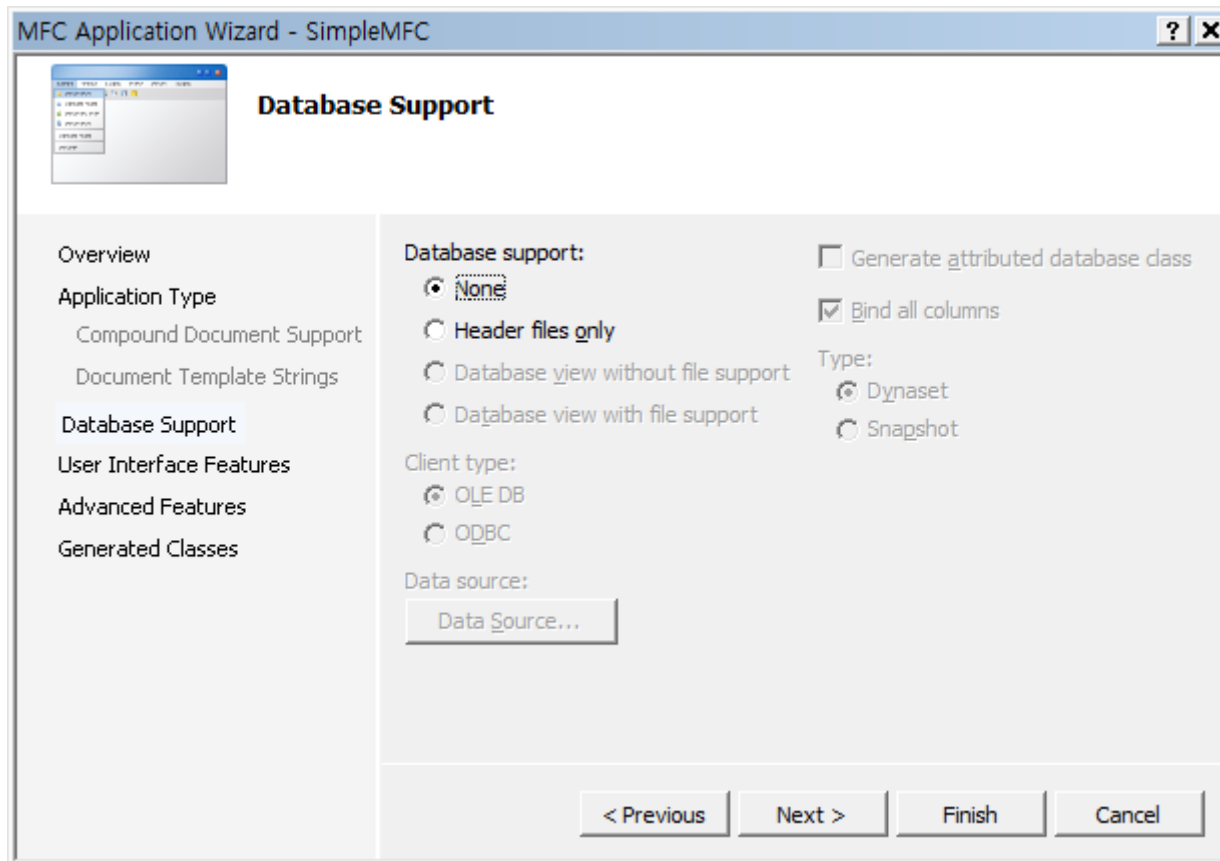
# MFC 응용 프로그램 생성 (3/8)

- AppWizard 2단계



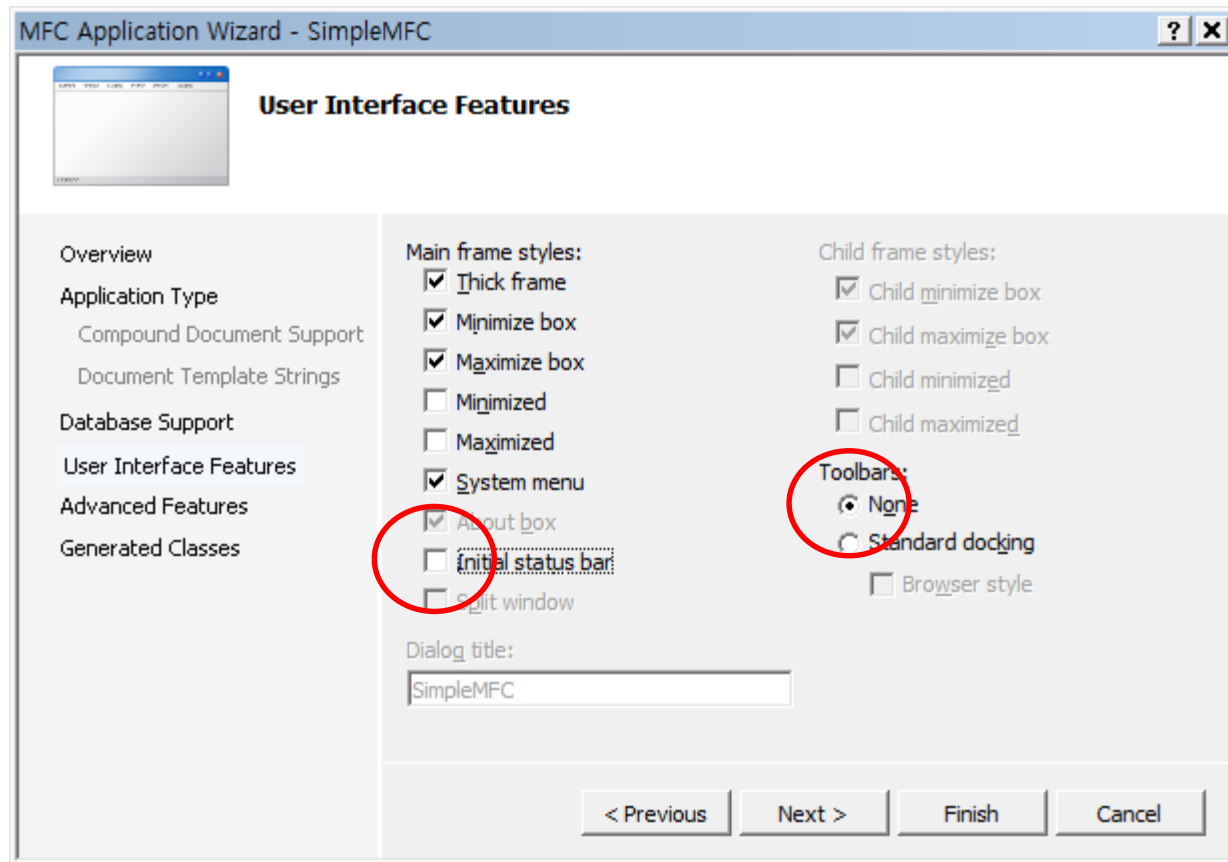
# MFC 응용 프로그램 생성 (4/8)

- AppWizard 3단계



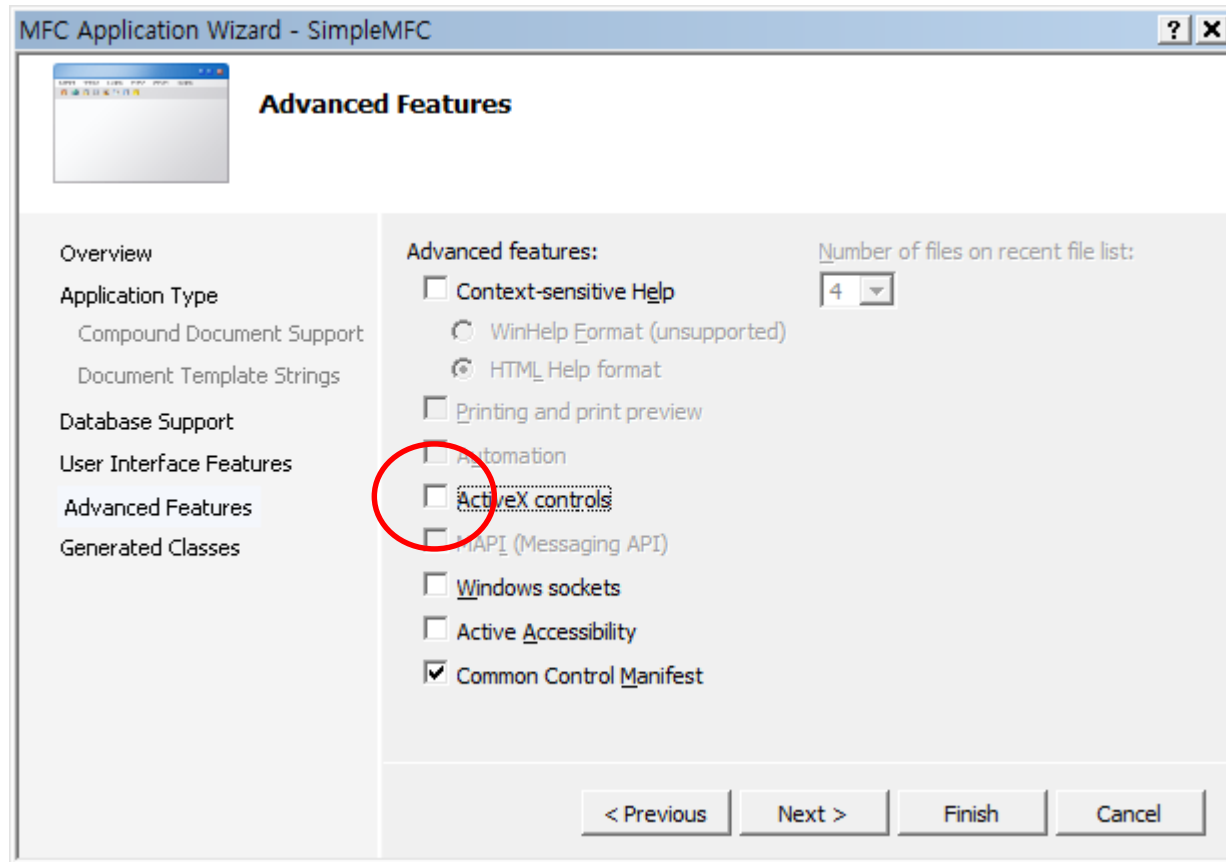
# MFC 응용 프로그램 생성 (5/8)

- AppWizard 4단계



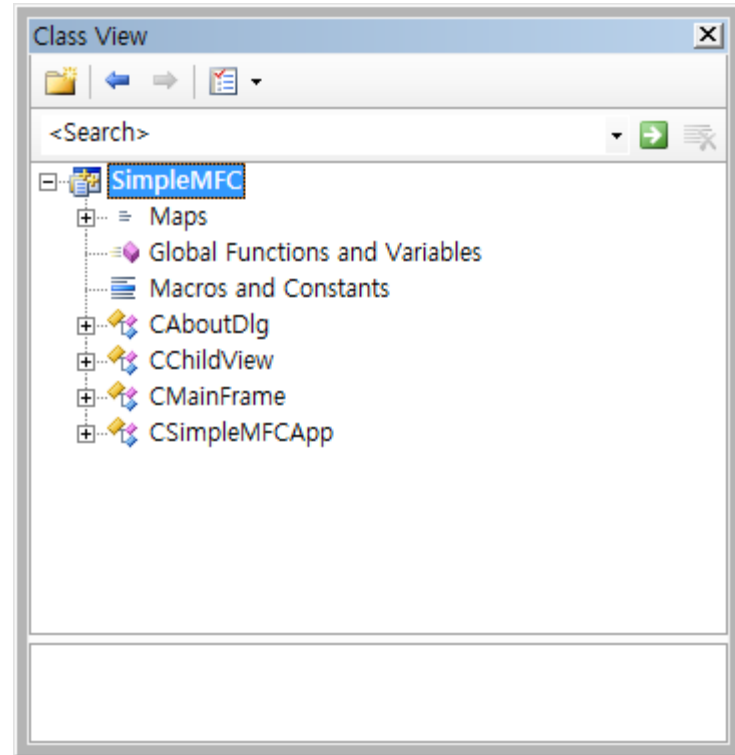
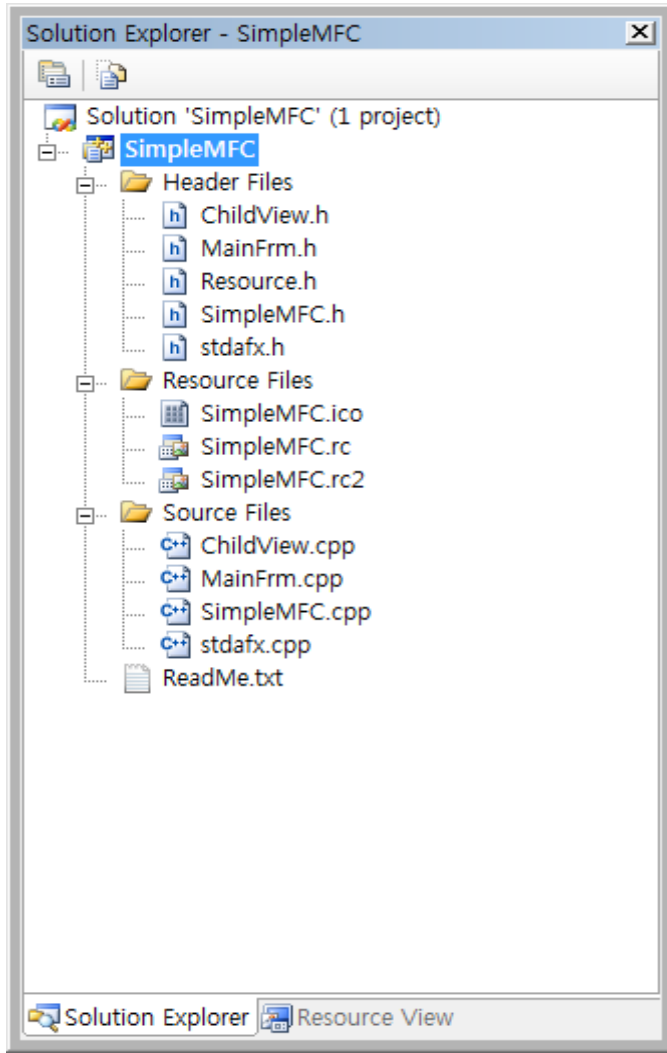
# MFC 응용 프로그램 생성 (6/8)

- AppWizard 5단계



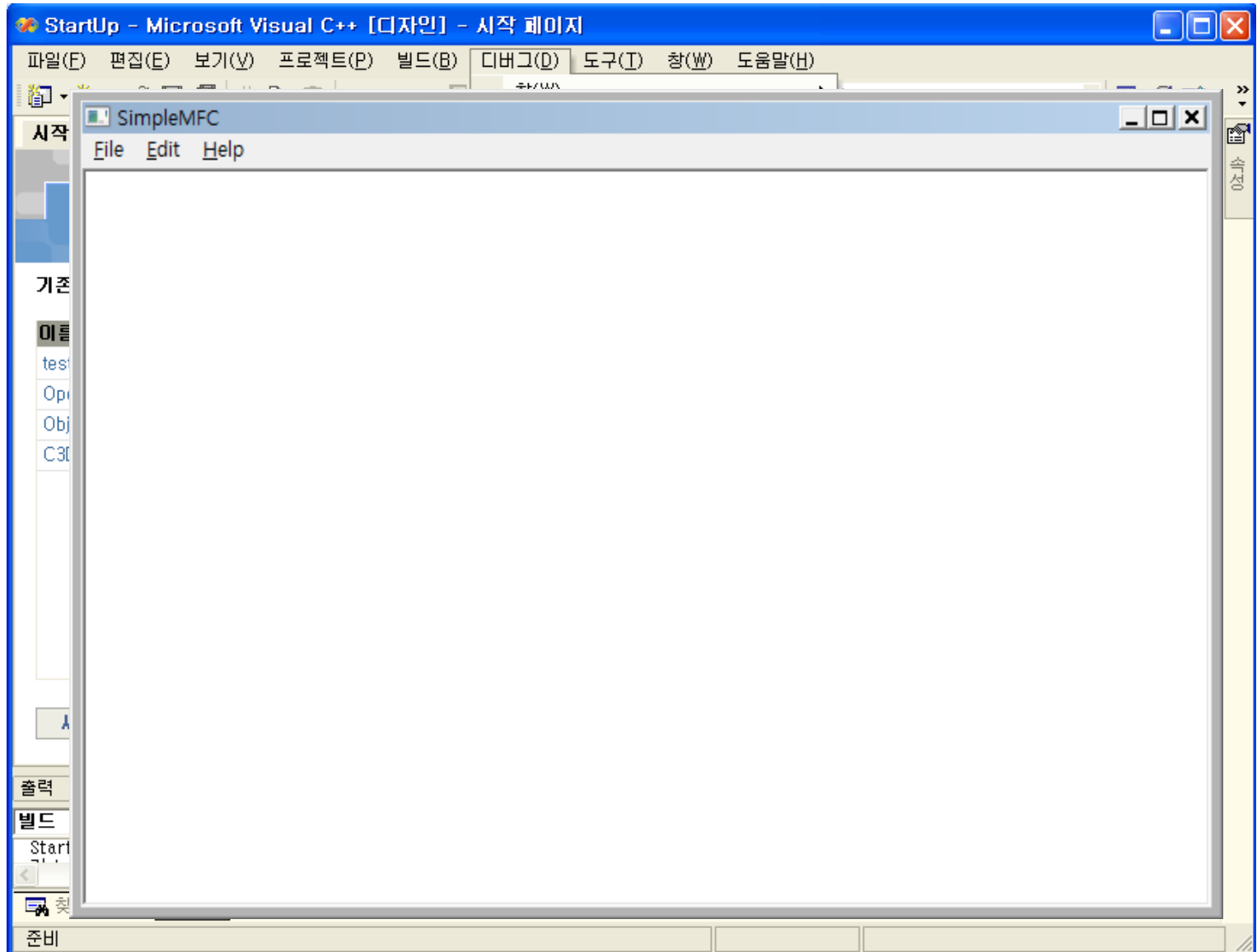
# MFC 응용 프로그램 생성 (7/8)

- 생성된 프로젝트와 클래스들



# MFC 응용 프로그램 생성 (8/8)

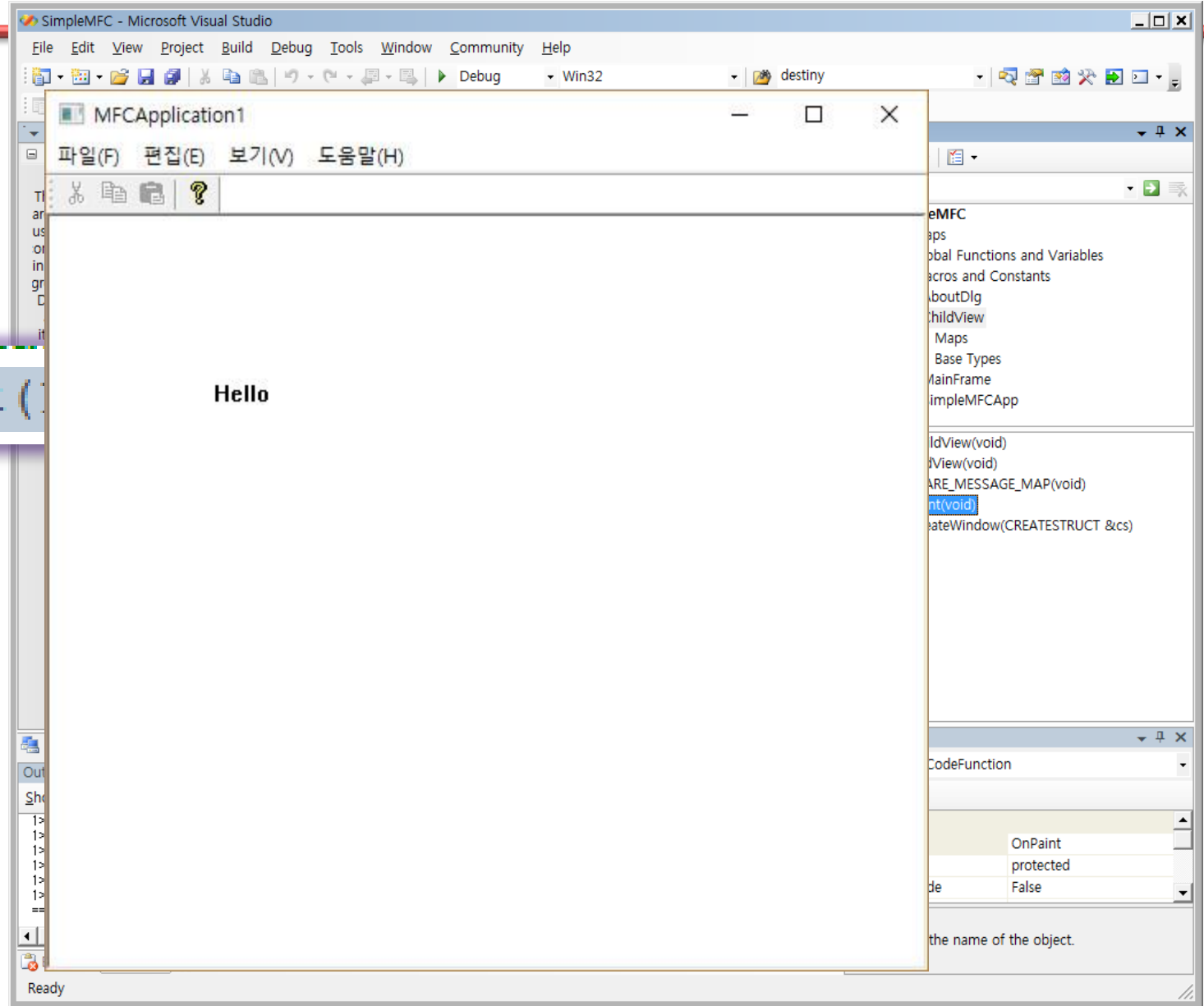
- 실행



# MFC 응용 프로그램 작성

- 코드의 변경

```
dc.TextOut (
```





# 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

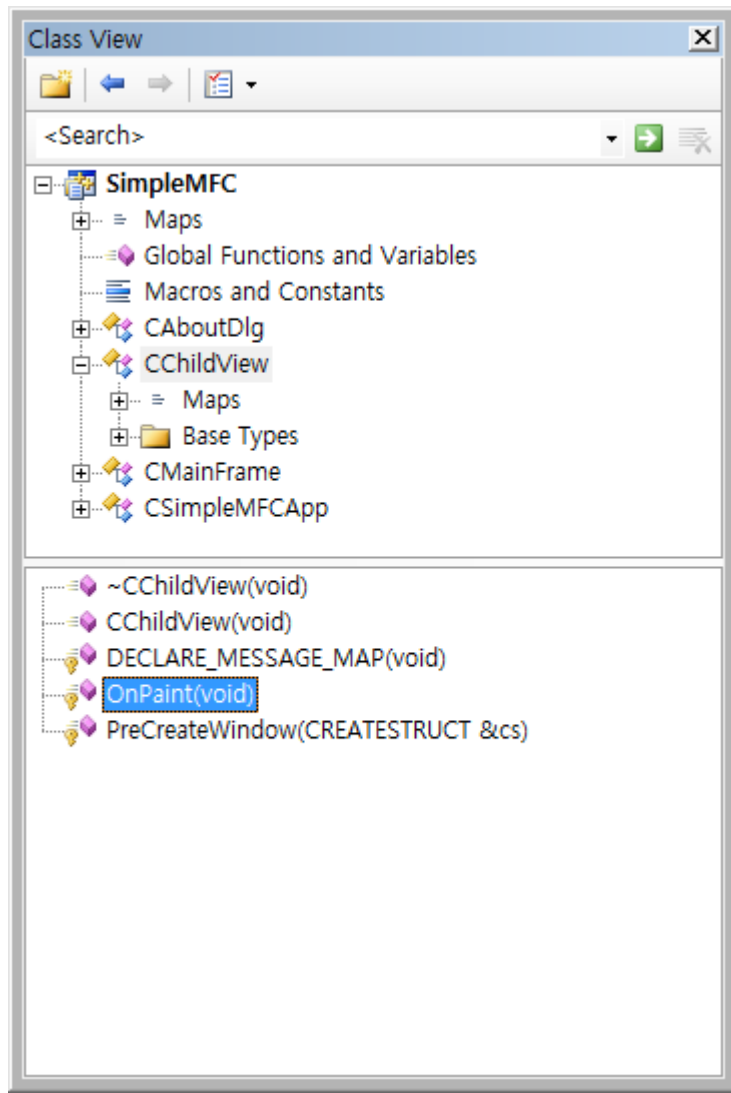
# 정리 - Project Workspace

- 프로젝트 워크스페이스의 항목별 설명

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

# 정리 - Project Workspace

- Class View 화면

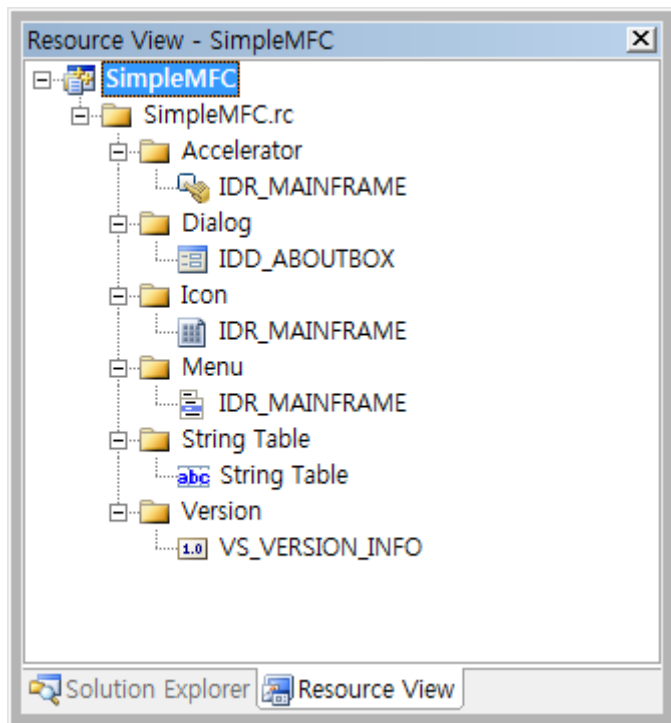


- 해당 항목을 더블 클릭하면 클래스 헤더가 나타나고 우측 버튼을 클릭하면 해당 클래스에 함수나 변수, 이벤트를 설정하도록 메뉴 등장
- 해당 클래스의 멤버 함수와 멤버 변수의 리스트
- +버튼을 클릭한 상태에서 해당 항목(변수)을 클릭하면 해당 항목이 설정되어 있는 소스 파일로 이동
- protected 형태로 설정되어 있을 경우(열쇠)
- protected 형태로 설정되어 있지 않을 경우는 열쇠 아이콘이 나타나지 않음

# 정리 - Project Workspace

- Resource View 화면

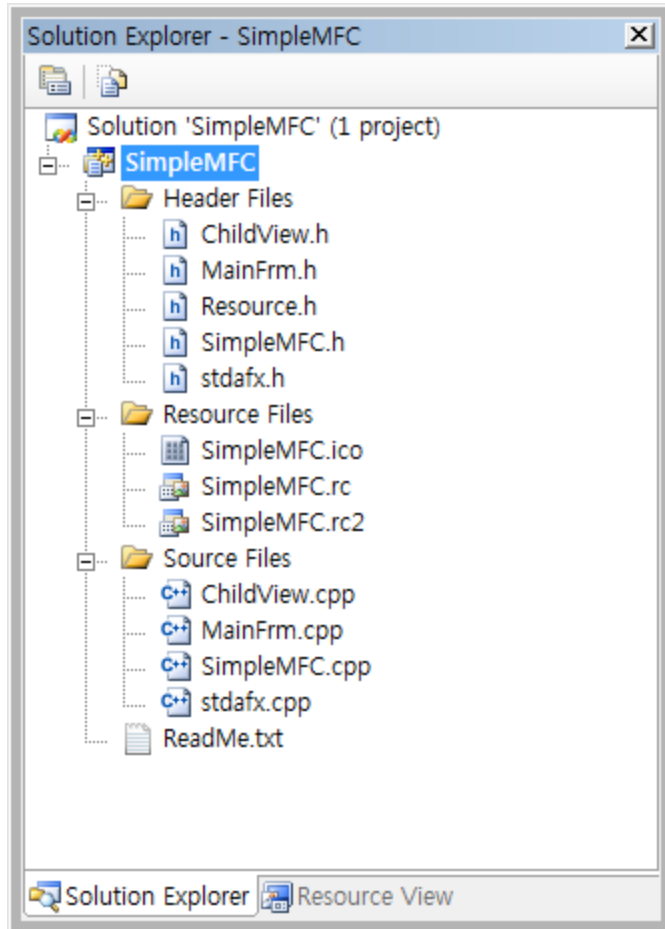
- Resource : 윈도우 프로그램을 만들 때 필요한 여러 자원



- 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



C++?

CLASS?

# 집에 가서...

---

- Class 정의 하는 법 알아오기
- 윈도우 프로그래밍 (개정판, 2014) 1장 읽어보기
  - 특히 page 54~66



---

Q & A