# 쉽게 배우는 MFC 윈도우 프로그래밍

Visual C++ 2019

2015·2017 버전 사용 가능

## 3장. MFC 프로그램 구조

# 목차


01 MFC 개요

02 MFC 구조

03 MFC 응용 프로그램 구조

# MFC 발전 과정

**표 3-1** MFC 발전 과정

| 연도 | 개발 도구 | MFC 버전 | 주요 특징 |
|---|---|---|---|
| 1992 | MS C/C++ 7.0 | 1.0 | 16비트 윈도우 API를 클래스화, OLE 1.0 지원 |
| 1993 | 비주얼 C++ 1.0 | 2.0 | 도큐먼트/뷰 구조 도입, DDX/DDV 지원<br>AppWizard/ClassWizard를 이용한 코드 생성 지원 |
| 1993 | 비주얼 C++ 1.5x | 2.5x | OLE 2.0 지원, ODBC 클래스 추가<br>속성 페이지/시트 추가 |
| 1994 | 비주얼 C++ 2.x | 3.x | 32비트 윈도우 API로 전환, 멀티스레드 지원<br>소켓/MAPI 클래스 추가 |
| 1995 | 비주얼 C++ 4.x<br>(mfc40.dll) | 4.x | 컨트롤 클래스 추가, DAO와 32비트 ODBC 지원<br>인터넷 관련 클래스 추가 |
| 1997 | 비주얼 C++ 5.0<br>(mfc42.dll) | 4.21 | ATL 추가, 액티브 도큐먼트 지원<br>DAO 3.5/ODBC 3.0 지원, 인터넷 프로그래밍 지원 향상 |
| 1998 | 비주얼 C++ 6.0<br>(mfc42.dll) | 6.0 | ATL 업그레이드, 동적 HTML 지원(CHtmlView)<br>OLE DB 클래스 추가, ADO 지원, 자잘한 클래스 추가/갱신 |

# MFC 발전 과정

| | | | |
|---|---|---|---|
| 2002 | 비주얼 C++ .NET<br>(mfc70.dll) | 7.0 | MFC와 ATL의 통합 강화<br>CException 클래스를 추상 클래스로 변경<br>CString과 BSTR 사이의 변환 방법 변경<br>CFile 클래스 변경(64비트 옵셋 지원)<br>CTime 클래스 변경(64비트 시간 지원)<br>부울식Boolean Expression 의미 변경(BOOL → bool)<br>ON_MESSAGE 매크로의 매개변수 타입 변경<br>OLE DB 템플릿 변경 |
| 2003 | 비주얼 C++ .NET 2003<br>(mfc71.dll) | 7.1 | ATL 일부 변경 |
| 2005 | 비주얼 C++ 2005<br>(mfc80.dll) | 8.0 | ATL 업그레이드<br>Windows Forms 지원<br>CTime 클래스 변경(1970년 이후 → 1900년 이후 지원) |
| 2007 | 비주얼 C++ 2008<br>(mfc90.dll) | 9.0 | 윈도우 비스타 공용 컨트롤 지원 |

# MFC 발전 과정

| | | | |
|---|---|---|---|
| 2008 | 비주얼 C++ 2008 Feature Pack(mfc90.dll) | 9.0 | 현대적 UI 지원을 위한 다수의 클래스 추가 |
| 2010 | 비주얼 C++ 2010 (mfc100.dll) | 10.0 | 윈도우 7에 새로 도입된 UI 지원을 위한 클래스 추가<br>멀티터치와 고해상도 인식 기능 지원<br>다시 시작 관리자Restart Manager 추가<br>AfxMessageBox()를 대체할 CTaskDialog 클래스 추가<br>애니메이션과 Direct2D 그래픽스 지원 |
| 2012 | 비주얼 C++ 2012 (mfc110.dll) | 11.0 | MFC 라이브러리와 정적 링크 시 실행 파일 크기 축소<br>버그 수정, 윈도우 XP/서버 2003 지원 중단<br>(SP1 설치 시 윈도우 XP/서버 2003용 코드 생성 지원) |
| 2013 | 비주얼 C++ 2013 (mfc120.dll) | 12.0 | 버그 수정, 윈도우 XP/서버 2003용 코드 생성 지원 |
| 2015 | 비주얼 C++ 2015 (mfc140.dll) | 14.0 | |
| 2017 | 비주얼 C++ 2017 (mfc140.dll) | 14.1x | 버그 수정<br>MFC DLL 이름 통합 |

# MFC 발전 과정

| | | |
|---|---|---|
| 2019 | 비주얼 C++ 2019<br>(mfc140.dll) | 14.2x |

# MFC 주요 특징

## MFC 주요 특징

- 윈도우 응용 프로그램 작성에 드는 수고 ↓
- API 기반인 SDK 프로그램과 대등한 속도
- 코드 크기 증가 최소화
- API 함수 직접 호출 가능
- C 언어로 작성된 윈도우 응용 프로그램을 쉽게 C++ 언어로 변경 가능
- SDK 프로그래밍에 대한 기반 지식 재활용
- C++ 언어 이용, C 언어에 비해 API를 편하게 사용 가능
- API를 직접 사용해서 구현하면 복잡도가 높은 부분을 MFC를 이용하면 간단히 구현 가능

# MFC 주요 특징

- 스레드를 생성하는 API 함수

```
HANDLE CreateThread (
    LPSECURITY_ATTRIBUTES lpThreadAttributes,
    SIZE_T dwStackSize,
    LPTHREAD_START_ROUTINE lpStartAddress,
    LPVOID lpParameter,
    DWORD dwCreationFlags,
    LPDWORD lpThreadId
);
```

# MFC 주요 특징

- 스레드를 생성하는 API 함수

```
CWinThread* AfxBeginThread (
    AFX_THREADPROC pfnThreadProc,
    LPVOID pParam,
    int nPriority = THREAD_PRIORITY_NORMAL,
    UINT nStackSize = 0,
    DWORD dwCreateFlags = 0,
    LPSECURITY_ATTRIBUTES lpSecurityAttrs = NULL
);
```

# MFC 구성 요소

- MFC 구성 요소

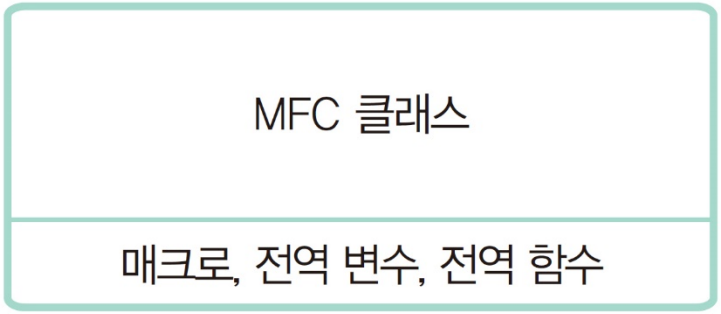



그림 3-1 MFC 구성 요소

# MFC 구성 요소

## MFC 클래스 계층도

- CObject 클래스에서 파생된 부류
- 독립적으로 존재하는 부류

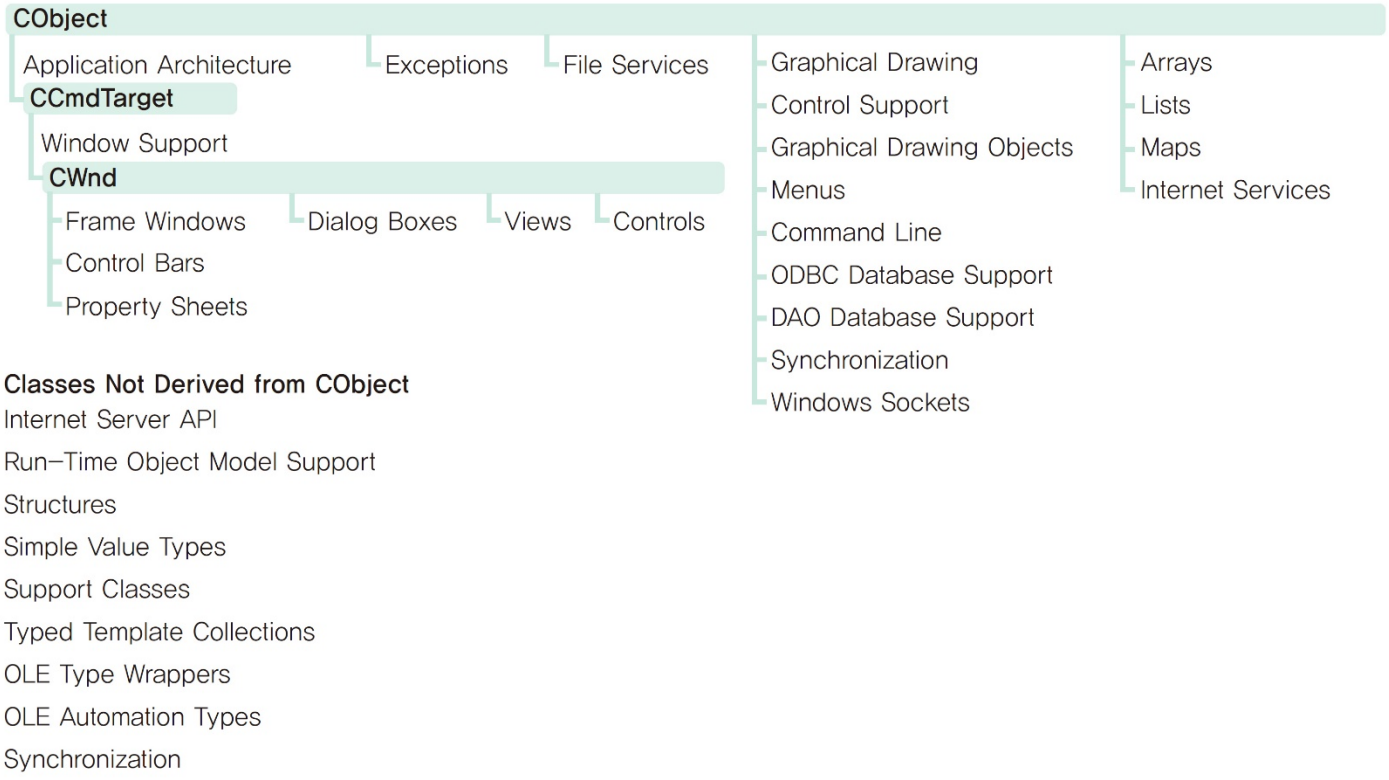



그림 3-2 MFC 클래스 계층도

# CObject 클래스

## CObject 서비스

표 3-2 CObject 서비스

| 서비스 이름 | 기능 |
|---|---|
| 실행 시간 클래스 정보 | 프로그램 실행 중 객체 정보를 알아낸다. |
| 동적 객체 생성 | 객체를 동적으로 생성한다. |
| 직렬화 | 객체를 저장하거나 읽어 들인다. |
| 타당성 점검 | 객체 상태를 점검한다. |
| 집합 클래스와의 호환성 | 서로 다른 클래스 객체를 집합 클래스에 저장할 수 있다. |

# 실행 시간 클래스 정보

- 실행 시간 클래스 정보 기능 추가

```
/* MyClass.h */
class CMyClass : public CObject
{
    DECLARE_DYNAMIC(CMyClass)
    ...
};

/* MyClass.cpp */
#include "MyClass.h"
IMPLEMENT_DYNAMIC(CMyClass, CObject)
...
```

# 실행 시간 클래스 정보

- 실행 시간 클래스 정보 사용 예

```
BOOL IsMyClass(CObject *pObj)
{
    // pObj가 가리키는 객체가 CMyClass 타입인지 확인한다.
    if(pObj->IsKindOf(RUNTIME_CLASS(CMyClass)){
        ...
    }
    else{
        ...
    }
}
```

# 동적 객체 생성

- 동적 객체 생성 기능 추가

```
/* MyClass.h */
class CMyClass : public CObject
{
    DECLARE_DYNCREATE(CMyClass)
public:
    CMyClass( ) ; // 기본 생성자가 반드시 있어야 한다.
    ...
};

/* MyClass.cpp */
#include "MyClass.h“

IMPLEMENT_DYNCREATE(CMyClass, CObject)
...
```

# 동적 객체 생성

- 동적 객체 생성 사용 예

```
// 객체를 동적으로 생성한다.
CRuntimeClass *pRuntimeClass = RUNTIME_CLASS(CMyClass);
CObject pObject = pRuntimeClass->CreateObject( );
// 객체 생성 여부를 확인한다.
ASSERT(pObject->IsKindOf(RUNTIME_CLASS(CMyClass)));
```

# 직렬화

- 직렬화 기능 추가 (1/2)

```
/* MyClass.h */
class CMyClass : public CObject
{
    DECLARE_SERIAL(CMyClass)
public:
    CMyClass( ) ; // 기본 생성자가 반드시 있어야 한다.
    virtual void Serialize(CArchive& ar); // 가상 함수를 재정의한다.
    ...
};
```

# 직렬화

- 직렬화 기능 추가 (2/2)

```
/* MyClass.cpp */
#include "MyClass.h“

IMPLEMENT_SERIAL(CMyClass, CObject, 1)

void CMyClass::Serialize(CArchive& ar)
{
    // CObject 클래스가 제공하는 가상 함수인 Serialize()를 재정의한다.
}
```

# 3단계 매크로

**표 3-3** 3단계 매크로

| 단계 | 사용 목적 | 매크로 이름 | 사용 위치 |
|---|---|---|---|
| 1 | 실행 시간 클래스 정보 | DECLARE_DYNAMIC | 클래스 선언부(*.H) |
| | | IMPLEMENT_DYNAMIC | 클래스 정의부(*.CPP) |
| 2 | 실행 시간 클래스 정보, 동적 객체 생성 | DECLARE_DYNCREATE | 클래스 선언부(*.H) |
| | | IMPLEMENT_DYNCREATE | 클래스 정의부(*.CPP) |
| 3 | 실행 시간 클래스 정보, 동적 객체 생성, 직렬화 | DECLARE_SERIAL | 클래스 선언부(*.H) |
| | | IMPLEMENT_SERIAL | 클래스 정의부(*.CPP) |

# 타당성 점검

- 타당성 점검 기능 추가 (1/2)

```
/* MyClass.h */
class CMyClass : public CObject
{
    // 멤버 변수
    int m_start, m_end;
public:
    virtual void AssertValid() const; // 가상 함수를 재정의한다.
    ...
};
```

# 타당성 점검

- 타당성 점검 기능 추가 (2/2)

```
/* MyClass.cpp */
#include "MyClass.h“


virtual void CMyClass::AssertValid() const
{
    CObject::AssertValid( ) ; // 베이스 클래스의 AssertValid( ) 함수를 먼저 호출한다.
    ASSERT(m_start > 0); // 멤버 변수 m_start 값이 0보다 큰지 검사한다.
    ASSERT(m_end < 100); // 멤버 변수 m_end 값이 100보다 작은지 검사한다.
}
```

# 집합 클래스와의 호환성

## ■ CObject 포인터를 저장할 수 있는 집합 클래스

**표 3-4** CObject 포인터를 저장할 수 있는 집합 클래스

| 종류 | 클래스 이름 |
|---|---|
| 배열 | CObArray, CArray(템플릿 클래스) |
| 리스트 | CObList, CList(템플릿 클래스) |
| 맵 | CMapWordToOb, CMapStringToOb, CMap(템플릿 클래스) |

# MFC 전역 함수

**표 3-5** MFC 전역 함수

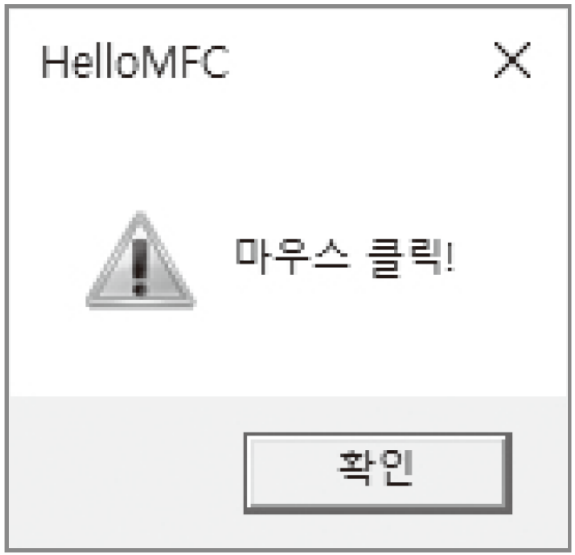
| 함수 이름 | 기능 |
|---|---|
| AfxMessageBox() | 메시지 상자를 표시한다. |
| AfxGetApp() | 응용 프로그램 객체의 주소를 리턴한다. |
| AfxGetMainWnd() | 메인 윈도우 객체의 주소를 리턴한다. |
| AfxGetAppName() | 응용 프로그램의 이름을 리턴한다. |
| AfxGetInstanceHandle() | 인스턴스 핸들을 리턴한다. |
| AfxRegisterWndClass() | 윈도우 클래스를 등록한다. |
| AfxBeginThread() | 스레드를 시작한다. (13장에서 소개) |
| AfxEndThread() | 스레드를 종료한다. (13장에서 소개) |

# AfxMessageBox( ) 함수

- 사용 예

```
void CMainFrame::OnLButtonDown(UINT nFlags, CPoint point)
{
    AfxMessageBox(_T("마우스 클릭!"));
}
```

- 결과



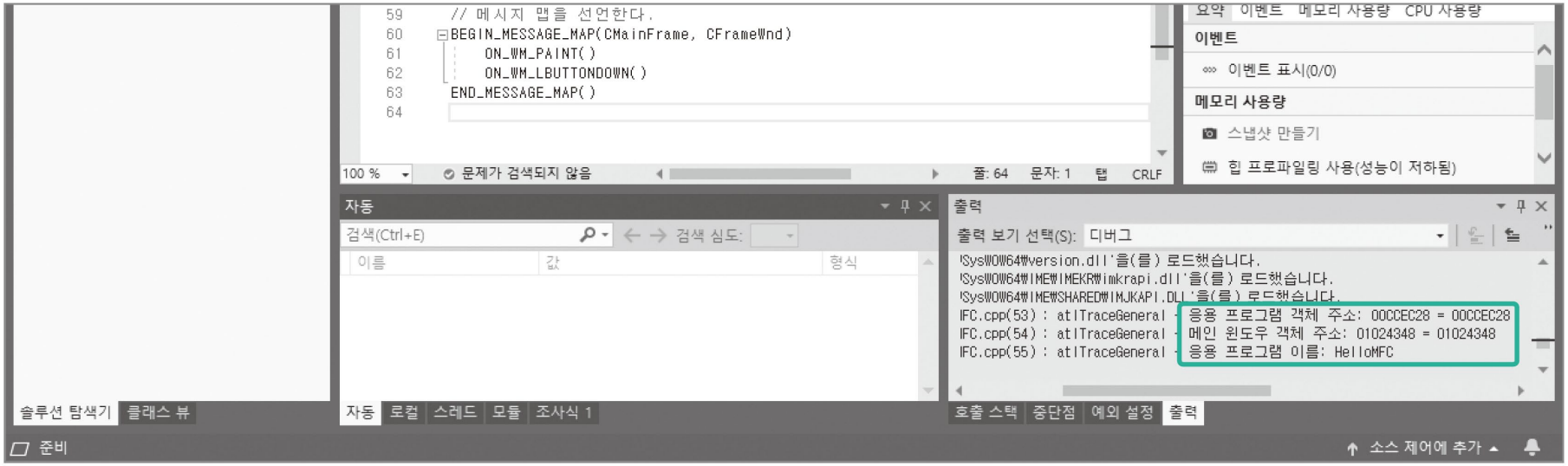
# AfxGetApp( ), AfxGetMainWnd( ), AfxGetAppName( ) 함수

- 사용 예

```
void CMainFrame::OnLButtonDown(UINT nFlags, CPoint point)
{
    TRACE(_T("응용 프로그램 객체 주소: %p = %p\n"), AfxGetApp(), &theApp);
    TRACE(_T("메인 윈도우 객체 주소: %p = %p\n"), AfxGetMainWnd(),
theApp.m_pMainWnd);
    TRACE(_T("응용 프로그램 이름: %s\n"), AfxGetAppName());
}
```

# AfxGetApp( ), AfxGetMainWnd( ), AfxGetAppName( ) 함수

- 결과



# AfxGetInstanceHandle( ) 함수

- 사용 예

```
void CMainFrame::OnLButtonDown(UINT nFlags, CPoint point)
{
    // 인스턴스 핸들값은 실행 파일이 로드된 가상 메모리 주소다.
    TRACE("인스턴스 핸들: %p\n", AfxGetInstanceHandle());
}
```

# MFC 응용 프로그램 구조

- 한 개의 윈도우로 구성된 프로그램

**그림 3-3** 한 개의 윈도우로 구성된 프로그램

# MFC 응용 프로그램 구조

- 두 개의 윈도우로 구성된 프로그램

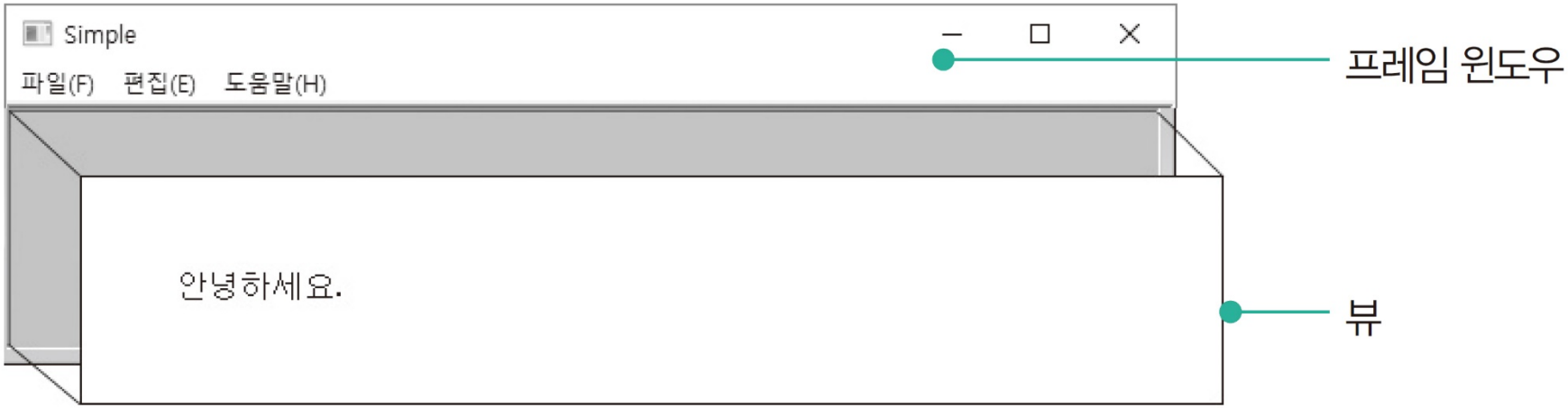



**그림 3-4** 두 개의 윈도우로 구성된 프로그램

# MFC 응용 프로그램 구조

- 프레임 윈도우와 뷰의 관계 : 부모-자식 관계

- 자식 윈도우의 특징
  - 항상 부모 윈도우의 클라이언트 영역 내에 위치
  - 부모 윈도우가 움직이면 같이 이동(상대 위치 고정)
  - 부모 윈도우가 최소화되면 자식 윈도우는 숨겨짐
  - 부모 윈도우가 파괴되면 자식 윈도우도 파괴됨

# MFC 응용 프로그램 구조

- 좀더 일반적인 MFC 프로그램의 구조

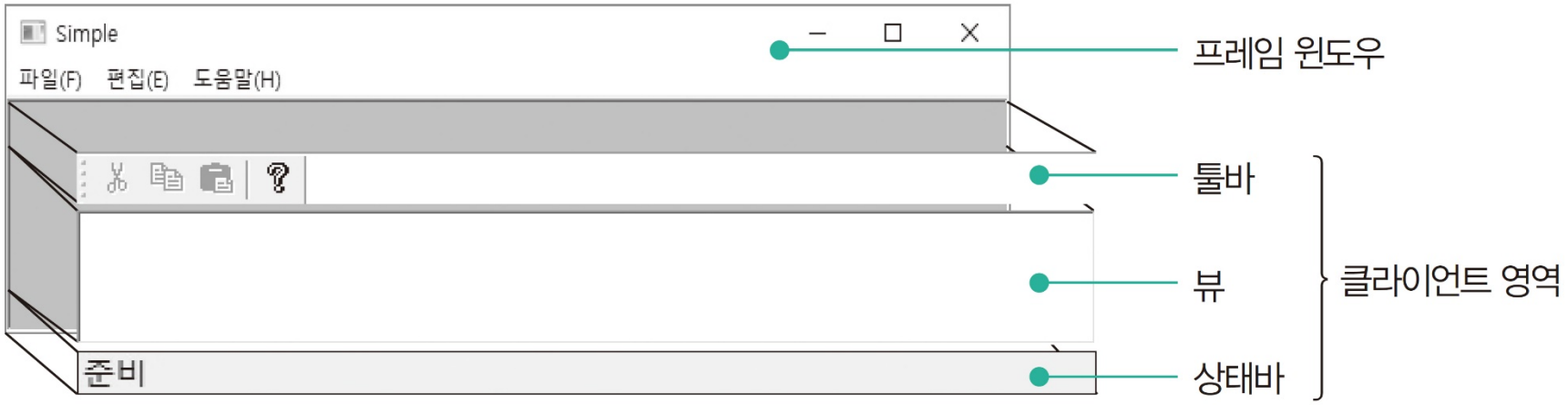



그림 3-5 일반적인 MFC 프로그램의 구조

# MFC 응용 프로그램 생성하기

- 프로젝트 종류 선택

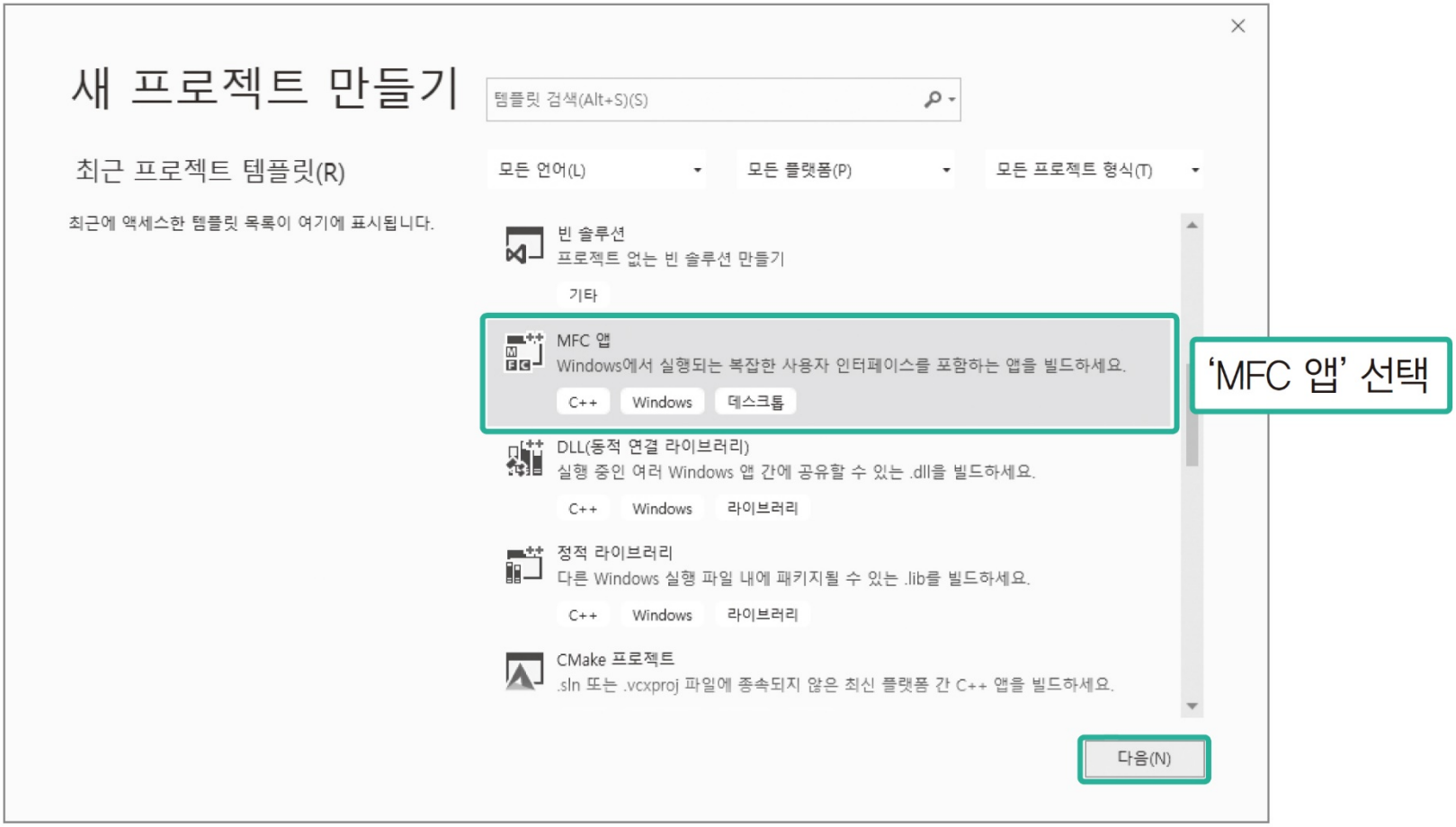



**그림 3-7** 프로젝트 종류 선택

# MFC 응용 프로그램 생성하기

- 프로젝트 이름과 위치 지정

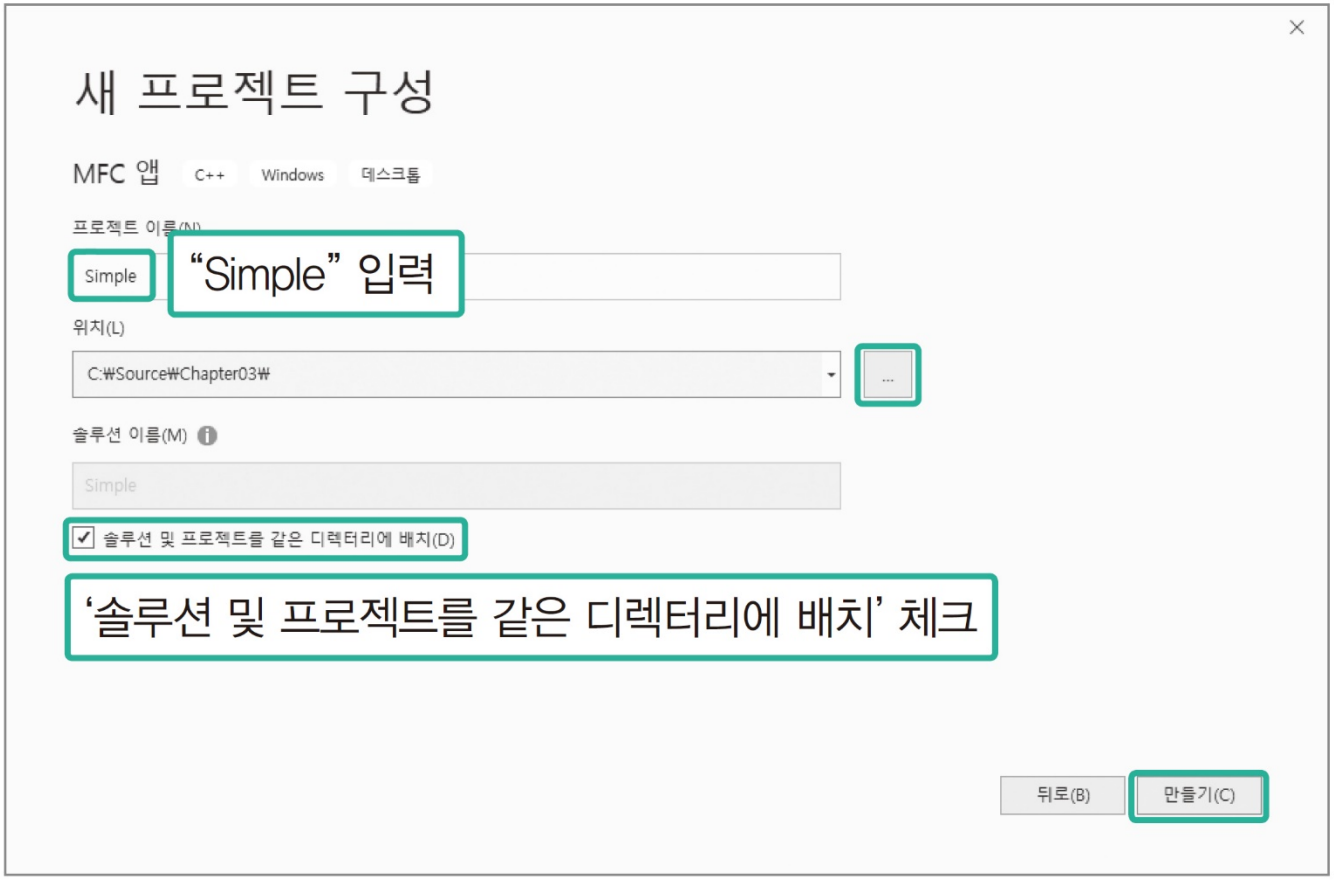



**그림 3-8** 프로젝트 이름과 위치 지정

# MFC 응용 프로그램 생성하기

- MFC 응용 프로그램 마법사 - 애플리케이션 종류

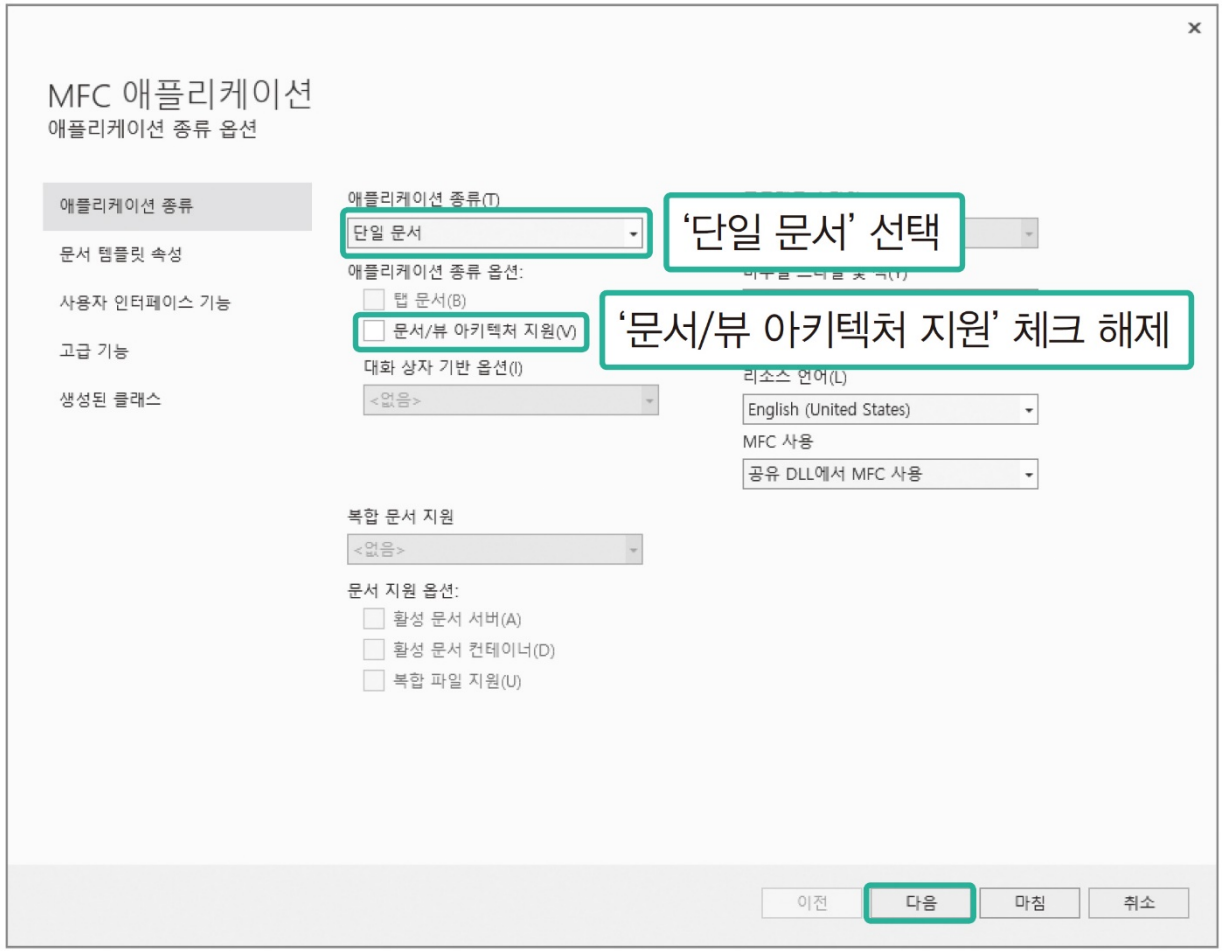



**그림 3-9** MFC 응용 프로그램 마법사 – 애플리케이션 종류

# MFC 응용 프로그램 생성하기

- MFC 응용 프로그램 마법사 - 문서 템플릿 속성

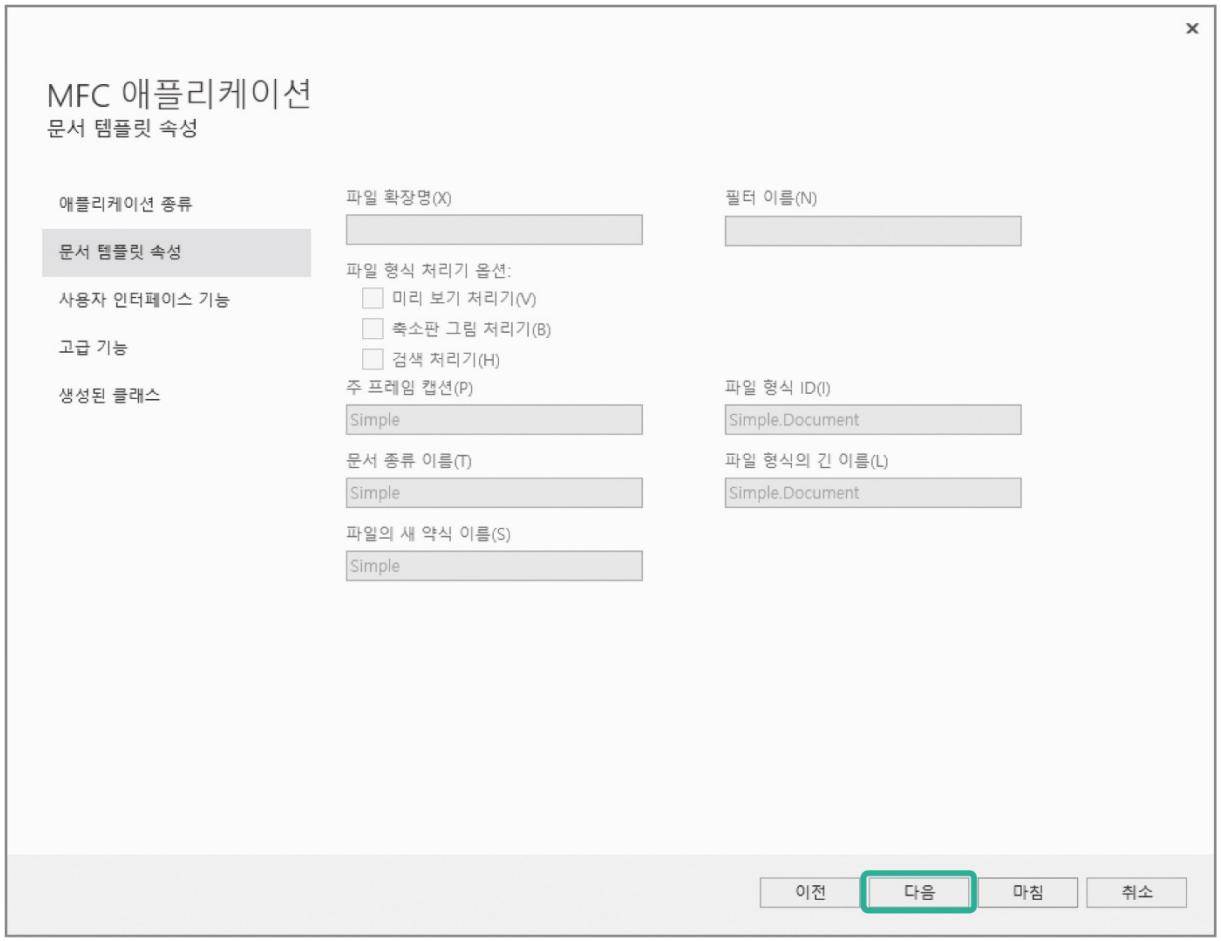



그림 3-10 MFC 응용 프로그램 마법사 – 문서 템플릿 속성

# MFC 응용 프로그램 생성하기

- MFC 응용 프로그램 마법사 - 사용자 인터페이스 기능

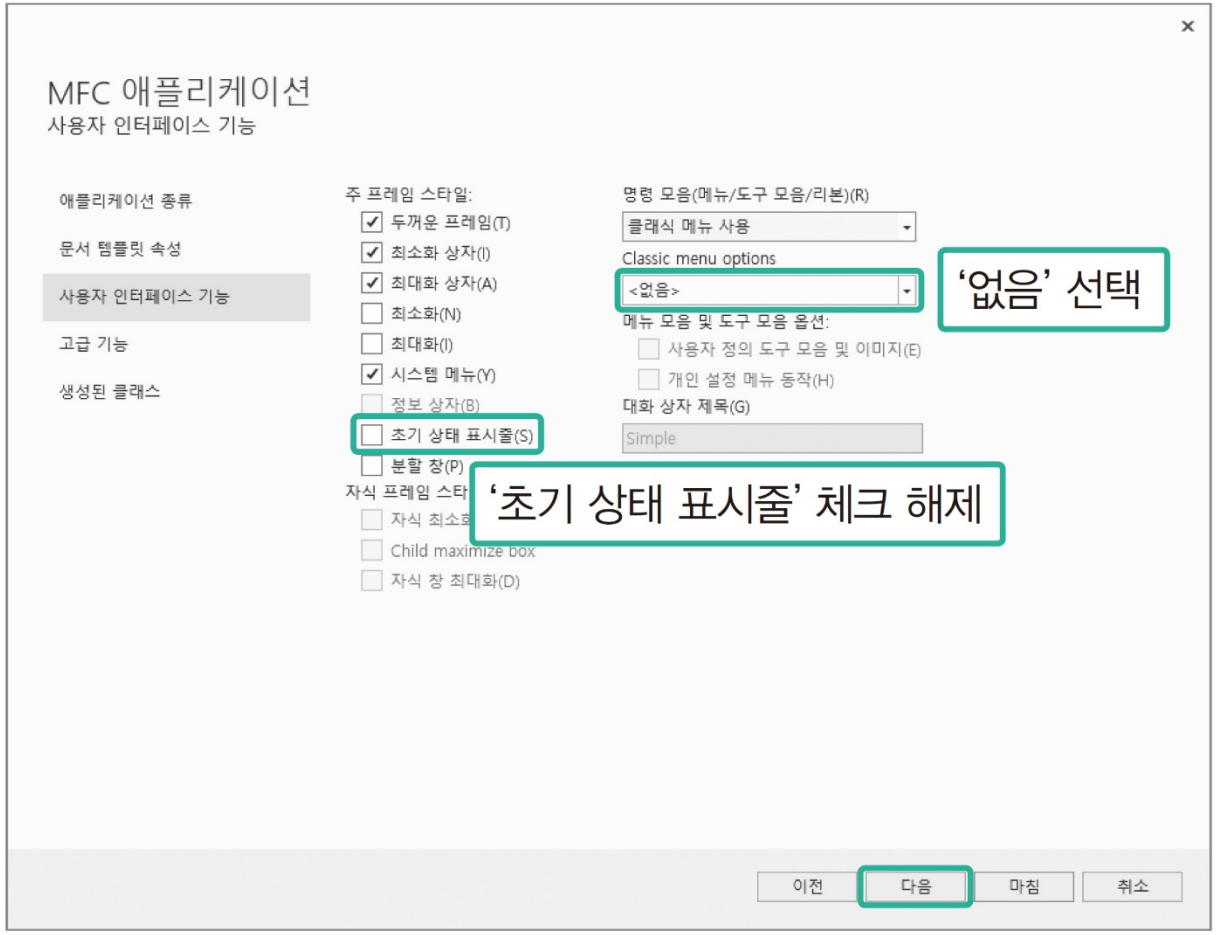



**그림 3-11** MFC 응용 프로그램 마법사 – 사용자 인터페이스 기능

# MFC 응용 프로그램 생성하기

- MFC 응용 프로그램 마법사 - 고급 기능

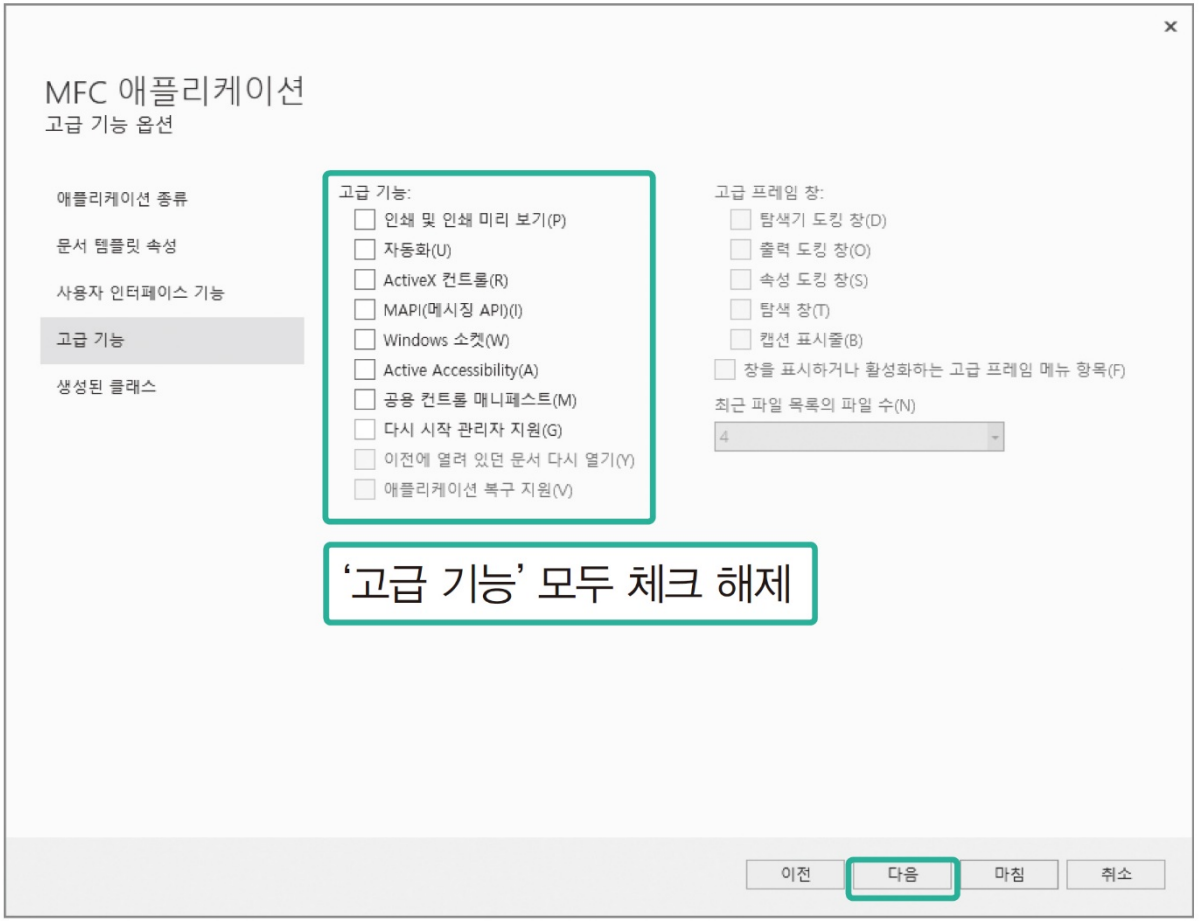



그림 3-12 MFC 응용 프로그램 마법사 – 고급 기능

# MFC 응용 프로그램 생성하기

- MFC 응용 프로그램 마법사 - 생성된 클래스

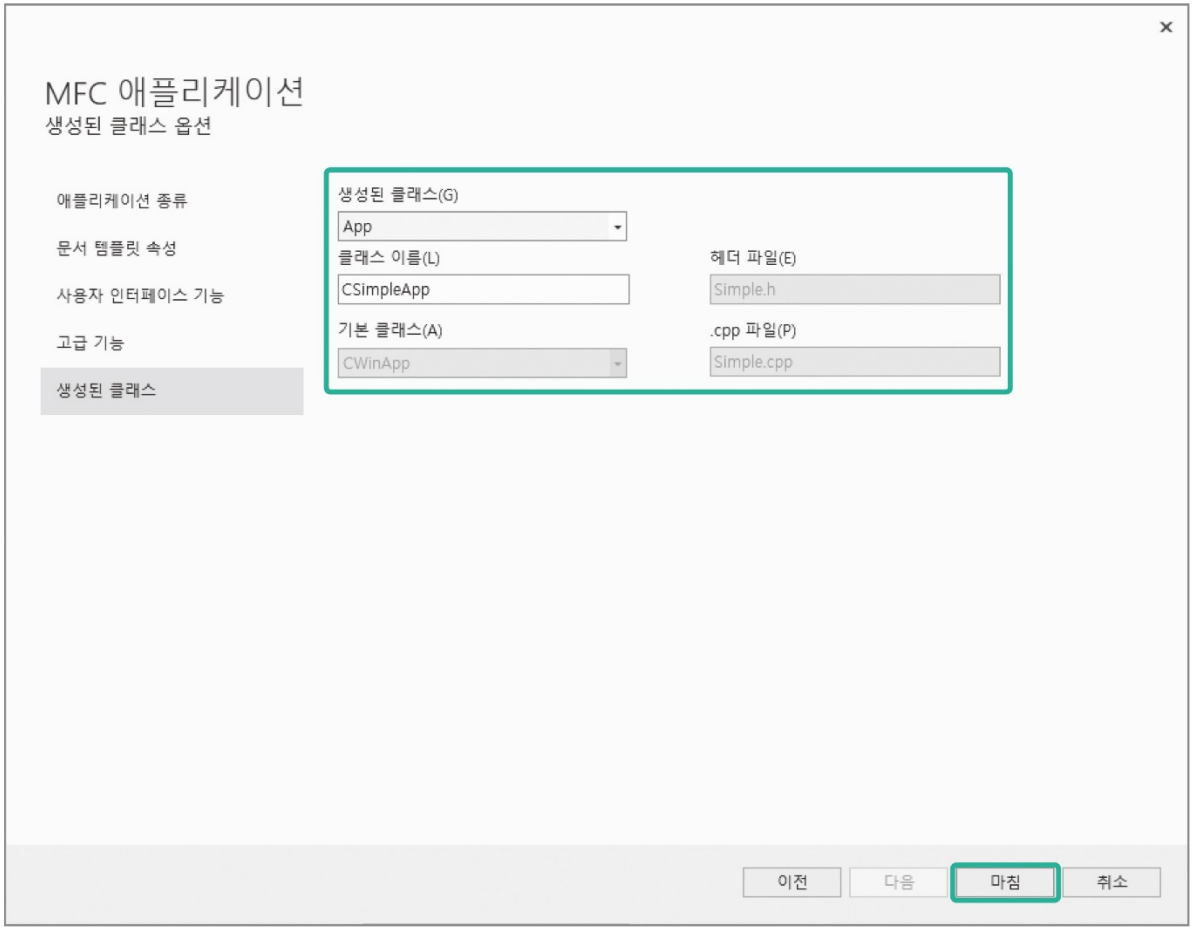



그림 3-13 MFC 응용 프로그램 마법사 – 생성된 클래스

# MFC 응용 프로그램 생성하기

## 코드 추가

```
dc.TextOut(50, 50, CString(_T("안녕하세요.")));
```

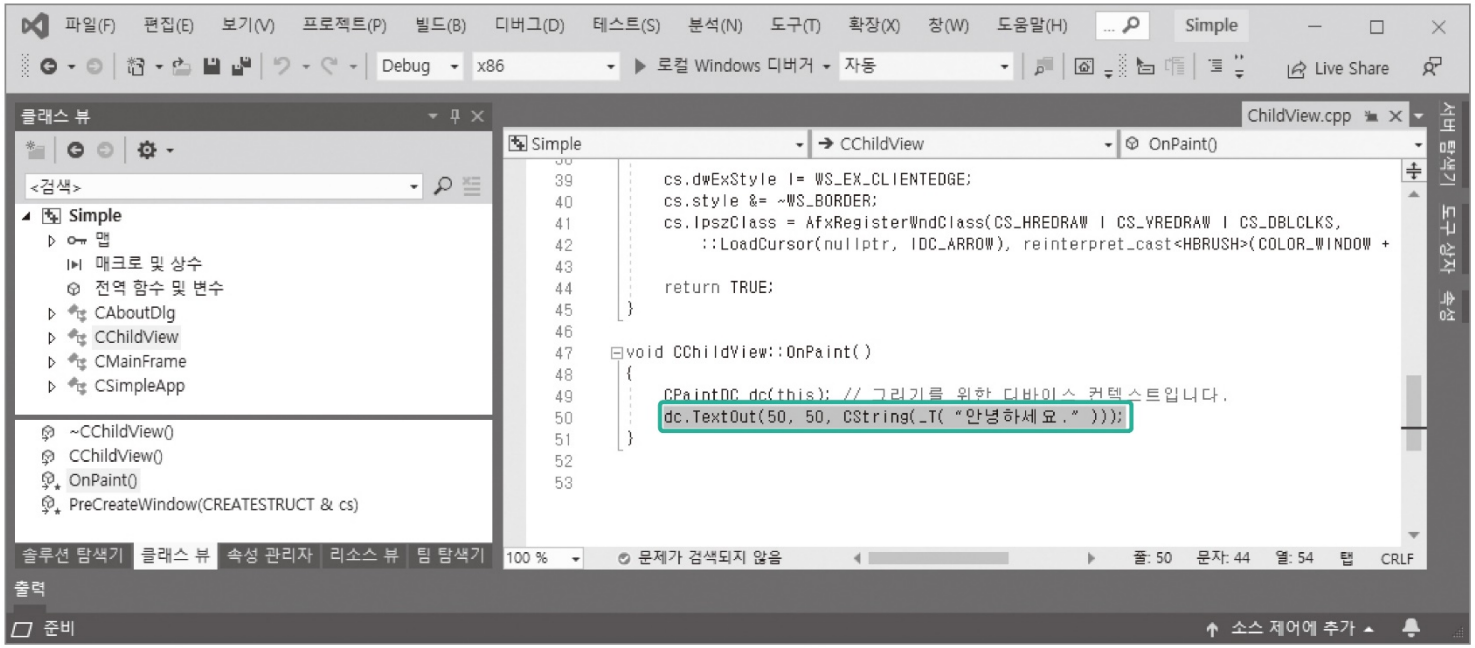



그림 3-14 코드 추가

# MFC 응용 프로그램 생성하기

- 결과

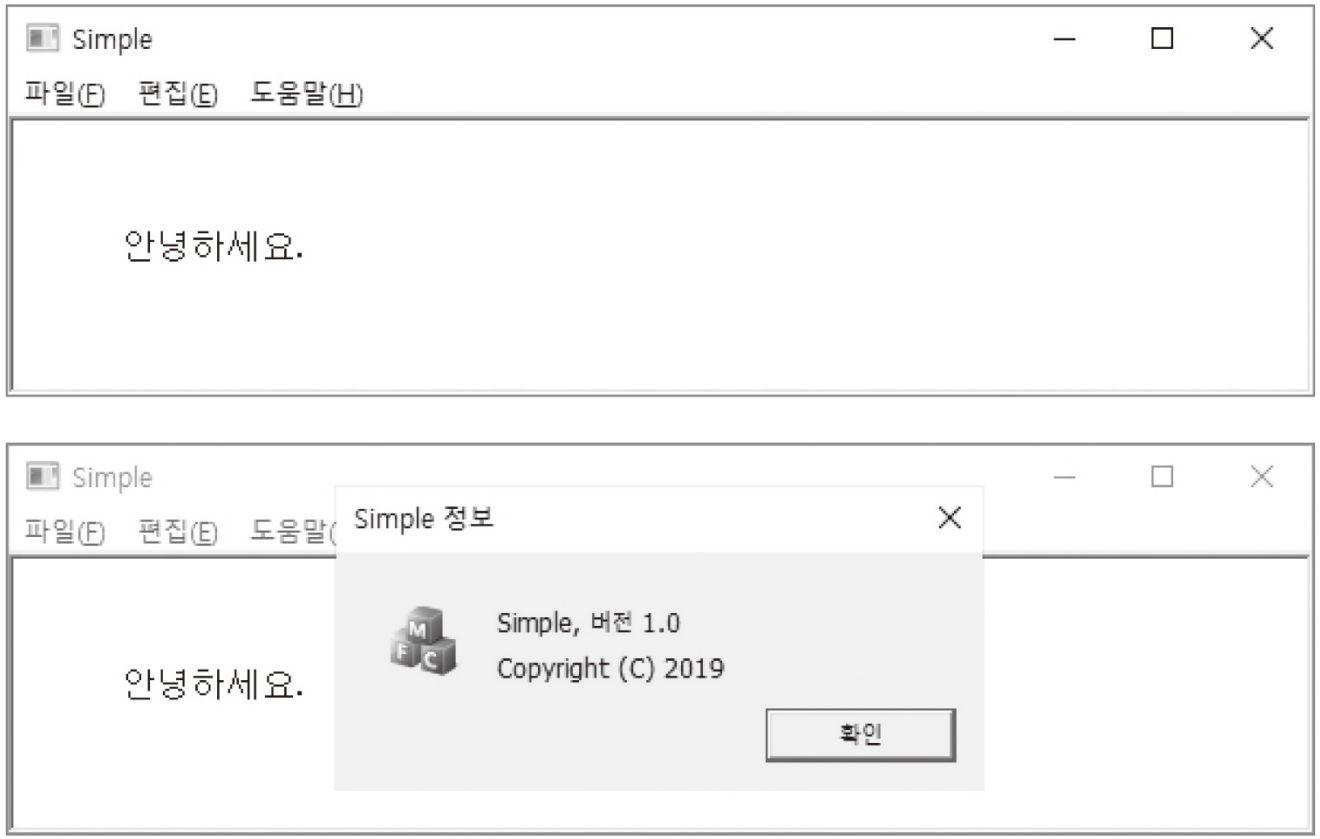



**그림 3-6** 실행 결과

# 응용 프로그램 마법사가 생성한 MFC 응용 프로그램 코드 분석

- 응용 프로그램 클래스

Simple.h

```
01  #include "resource.h"
02
03  class CSimpleApp : public CWinApp
04  {
05  public:
06      CSimpleApp() noexcept;
07  public:
08  virtual BOOL InitInstance();
09  virtual int ExitInstance();
10  public:
11  afx_msg void OnAppAbout();
12  DECLARE_MESSAGE_MAP()
13  };
14
15  extern CSimpleApp theApp;
```

# 응용 프로그램 마법사가 생성한 MFC 응용 프로그램 코드 분석

## 응용 프로그램 클래스(1/3)

Simple.cpp

```
01  #include "pch.h"
02  #include "framework.h"
03  #include "afxwinappex.h"
04  #include "afxdialogex.h"
05  #include "Simple.h"
06  #include "MainFrm.h"
07
08  BEGIN_MESSAGE_MAP(CSimpleApp, CWinApp)
09     ON_COMMAND(ID_APP_ABOUT, &CSimpleApp::OnAppAbout)
10  END_MESSAGE_MAP()
11
12  CSimpleApp::CSimpleApp() noexcept
13  {
14  SetAppID(_T("Simple.AppID.NoVersion"));
15  }
16
17  CSimpleApp theApp;
18
```

# 응용 프로그램 마법사가 생성한 MFC 응용 프로그램 코드 분석

## 응용 프로그램 클래스(2/3)

Simple.cpp

```
19  BOOL CSimpleApp::InitInstance()
20  {
21     CWinApp::InitInstance();
22     EnableTaskbarInteraction(FALSE);
23     SetRegistryKey(_T("로컬 애플리케이션 마법사에서 생성된 애플리케이션"));
24
25     CFrameWnd* pFrame = new CMainFrame;
26     if (!pFrame)
27        return FALSE;
28     m_pMainWnd = pFrame;
29
30     pFrame->LoadFrame(IDR_MAINFRAME,
31        WS_OVERLAPPEDWINDOW | FWS_ADDTOTITLE, nullptr,
32  nullptr);
33  pFrame->ShowWindow(SW_SHOW);
34  pFrame->UpdateWindow();
35
```

# 응용 프로그램 마법사가 생성한 MFC 응용 프로그램 코드 분석

- 응용 프로그램 클래스(3/3)

Simple.cpp

```
36  return TRUE;
37  }
38
39  int CSimpleApp::ExitInstance()
40  {
41  return CWinApp::ExitInstance();
42  }
43
44  // 대화 상자 관련 클래스 선언 및 정의 부분(생략)
45  ...
46
47  void CSimpleApp::OnAppAbout()
48  {
49  CAboutDlg aboutDlg;
50  aboutDlg.DoModal();
51  }
```

# 응용 프로그램 마법사가 생성한 MFC 응용 프로그램 코드 분석

## 프레임 윈도우 클래스(1/2)

MainFrm.h

```
01  #include "ChildView.h"
02
03  class CMainFrame : public CFrameWnd
04  {
05  public:
06     CMainFrame() noexcept;
07  protected:
08     DECLARE_DYNAMIC(CMainFrame)
09  public:
10     virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
11     virtual BOOL OnCmdMsg(UINT nID, int nCode, void* pExtra,
    AFX_CMDHANDLERINFO* pHandlerInfo);
12  public:
13     virtual ~CMainFrame();
14  #ifdef _DEBUG
15     virtual void AssertValid() const;
16     virtual void Dump(CDumpContext& dc) const;
```

# 응용 프로그램 마법사가 생성한 MFC 응용 프로그램 코드 분석

- 프레임 윈도우 클래스(2/2)

MainFrm.h

```
17  #endif
18      CChildView m_wndView;
19  protected:
20      afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
21      afx_msg void OnSetFocus(CWnd *pOldWnd);
22      DECLARE_MESSAGE_MAP()
23  };
```

# 응용 프로그램 마법사가 생성한 MFC 응용 프로그램 코드 분석

## 프레임 윈도우 클래스(1/4)

MainFrm.cpp

```
01  #include "pch.h"
02  #include "framework.h"
03  #include "Simple.h"
04
05  #include "MainFrm.h"
06
07  IMPLEMENT_DYNAMIC(CMainFrame, CFrameWnd)
08
09  BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
10      ON_WM_CREATE()
11      ON_WM_SETFOCUS()
12  END_MESSAGE_MAP()
13
14  CMainFrame::CMainFrame() noexcept
15  {
16  }
17
```

# 응용 프로그램 마법사가 생성한 MFC 응용 프로그램 코드 분석

- 프레임 윈도우 클래스(2/4)

MainFrm.cpp

```
18  CMainFrame::~CMainFrame()
19  {
20  }
21
22  int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
23  {
24     if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
25        return -1;
26
27     if (!m_wndView.Create(nullptr, nullptr, AFX_WS_DEFAULT_VIEW,
28        CRect(0, 0, 0, 0), this, AFX_IDW_PANE_FIRST, nullptr))
29     {
30        TRACE0("뷰 창을 만들지 못했습니다.\n");
31        return -1;
32     }
33     return 0;
34  }
```

# 응용 프로그램 마법사가 생성한 MFC 응용 프로그램 코드 분석

- 프레임 윈도우 클래스(3/4)

MainFrm.cpp

```
35
36  BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
37  {
38     if (!CFrameWnd::PreCreateWindow(cs))
39        return FALSE;
40
41     cs.dwExStyle &= ~WS_EX_CLIENTEDGE;
42     cs.lpszClass = AfxRegisterWndClass(0);
43     return TRUE;
44  }
45
46  #ifdef _DEBUG
47  void CMainFrame::AssertValid() const
48  {
49     CFrameWnd::AssertValid();
50  }
51
```

# 응용 프로그램 마법사가 생성한 MFC 응용 프로그램 코드 분석

- 프레임 윈도우 클래스(4/4)

MainFrm.cpp

```
52  void CMainFrame::Dump(CDumpContext& dc) const
53  {
54     CFrameWnd::Dump(dc);
55  }
56  #endif //_DEBUG
57
58  void CMainFrame::OnSetFocus(CWnd* /*pOldWnd*/)
59  {
60     m_wndView.SetFocus();
61  }
62
63  BOOL CMainFrame::OnCmdMsg(UINT nID, int nCode, void* pExtra,
    AFX_CMDHANDLERINFO* pHandlerInfo)
64  {
65     if (m_wndView.OnCmdMsg(nID, nCode, pExtra, pHandlerInfo))
66        return TRUE;
67
68     return CFrameWnd::OnCmdMsg(nID, nCode, pExtra, pHandlerInfo);
69  }
```

# 응용 프로그램 마법사가 생성한 MFC 응용 프로그램 코드 분석

ChildView.h

- 뷰 클래스

```
01  class CChildView : public CWnd
02  {
03  public:
04      CChildView();
05  protected:
06      virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
07  public:
08      virtual ~CChildView();
09  protected:
10      afx_msg void OnPaint();
11      DECLARE_MESSAGE_MAP()
12  };
```

# 응용 프로그램 마법사가 생성한 MFC 응용 프로그램 코드 분석

- 뷰 클래스

ChildView.cpp

```
01  #include "pch.h"
02  #include "framework.h"
03  #include "Simple.h"
04  #include "ChildView.h"
05
06  CChildView::CChildView()
07  {
08  }
09
10  CChildView::~CChildView()
11  {
12  }
13
14  BEGIN_MESSAGE_MAP(CChildView, CWnd)
15  ON_WM_PAINT()
16  END_MESSAGE_MAP()
17
```

# 응용 프로그램 마법사가 생성한 MFC 응용 프로그램 코드 분석

- 뷰 클래스

ChildView.cpp

```
18  BOOL CChildView::PreCreateWindow(CREATESTRUCT& cs)
19  {
20  if (!CWnd::PreCreateWindow(cs))
21  return FALSE;
22
23  cs.dwExStyle |= WS_EX_CLIENTEDGE;
24  cs.style &= ~WS_BORDER;
25  cs.lpszClass = AfxRegisterWndClass(CS_HREDRAW | CS_VREDRAW | CS_DBLCLKS,
26  ::LoadCursor(nullptr, IDC_ARROW), reinterpret_cast<HBRUSH>
    (COLOR_WINDOW + 1), nullptr);
27
28  return TRUE;
29  }
30
31  void CChildView::OnPaint()
32  {
33  CPaintDC dc(this);
34  dc.TextOut(50, 50, CString(_T("안녕하세요.")));
35  }
```

# 응용 프로그램 마법사가 생성한 MFC 응용 프로그램 코드 분석

## ■ 클래스별 핵심 내용

**표 3-6** 클래스별 핵심 내용

| 클래스 종류 | 베이스 클래스 이름 | 핵심 함수 – 주 역할 |
|---|---|---|
| 응용 프로그램 클래스 | CWinApp | InitInstance() – 프레임 윈도우 생성<br>Run() – 메시지 루프 제공 |
| 프레임 윈도우 클래스 | CFrameWnd | OnCreate() – 뷰 생성 |
| 뷰 클래스 | CWnd | OnPaint() – 화면 출력, 사용자 입력 |