다양한 뷰클래스 I

(6주차)

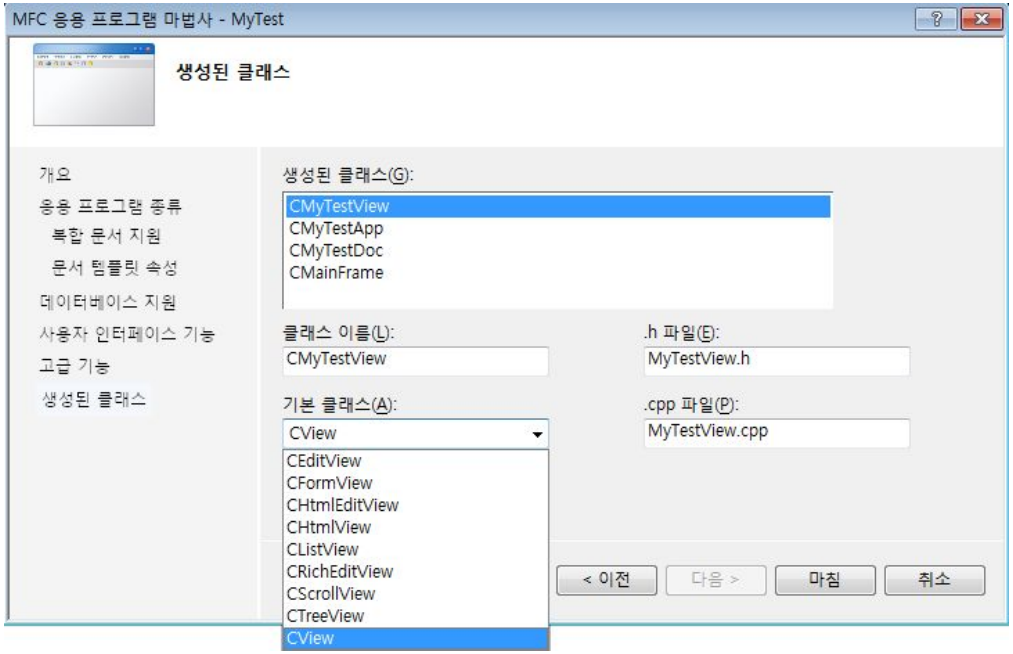
학습개요

- 학습 목표
  - MFC 뷰 클래스의 특징과 용도를 이해한다.
  - 다양한 뷰 클래스를 다루는 방법을 익힌다.

- 학습 내용
  - 뷰 클래스 종류
  - 리스트 뷰
  - 실습

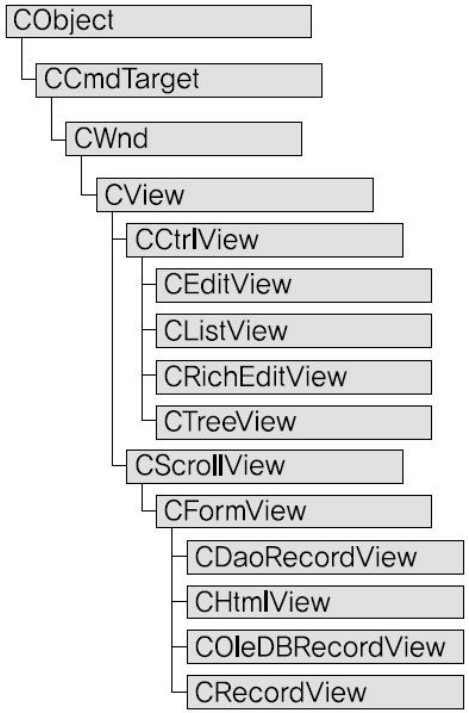
뷰 클래스 종류

- 뷰 클래스 선택



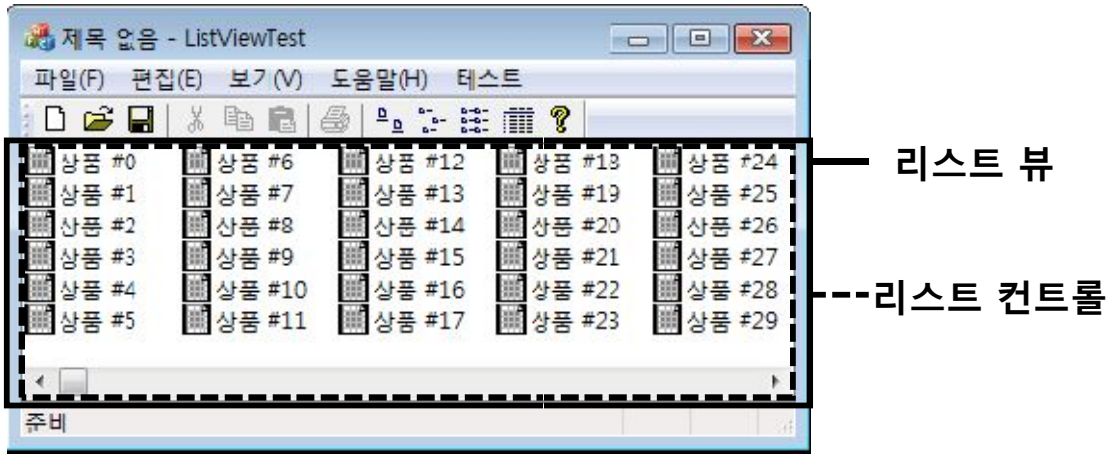
뷰 클래스 종류

- MFC 클래스 계층도



리스트 뷰

- CListView 클래스
  - 도큐먼트/뷰 구조 응용 프로그램에서 리스트 컨트롤을 이용하여 뷰의 기능을 제공
    - 리스트 컨트롤의 기능, 즉 여러 항목을 이미지와 텍스트로 보여주는 기능을 뷰에서 손쉽게 활용 가능



리스트 뷰

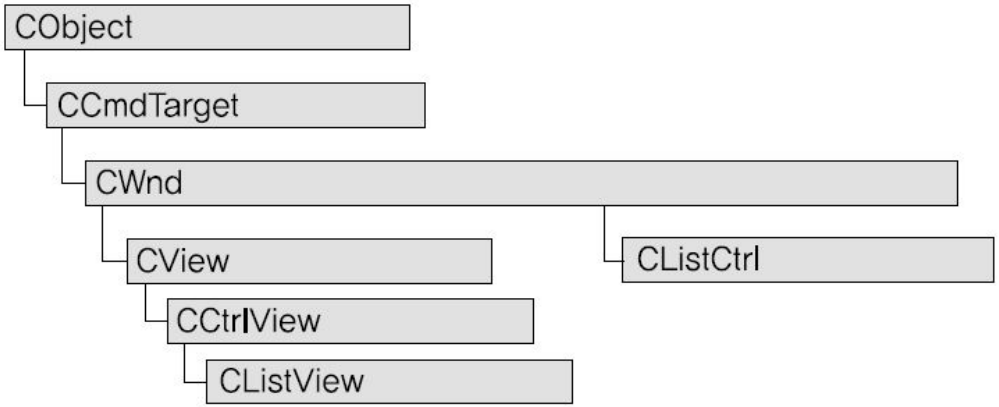
- CListView 클래스 사용 예

```
// 리스트 컨트롤 객체에 대한 레퍼런스를 얻는다.
CListCtrl& list = GetListCtrl();

// 레퍼런스를 통해 리스트 컨트롤을 사용한다.
list.SetImageList(...);
list.InsertColumn(...);
...
```

리스트 뷰

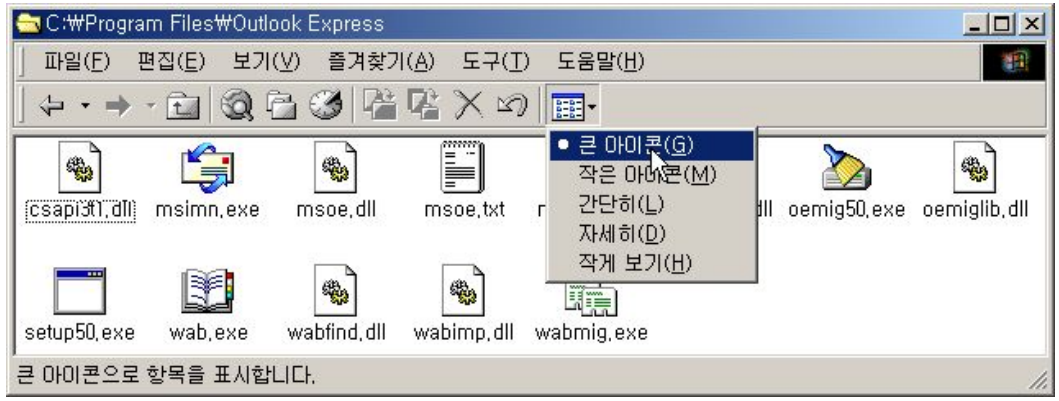
- MFC 클래스 계층도



리스트 컨트롤

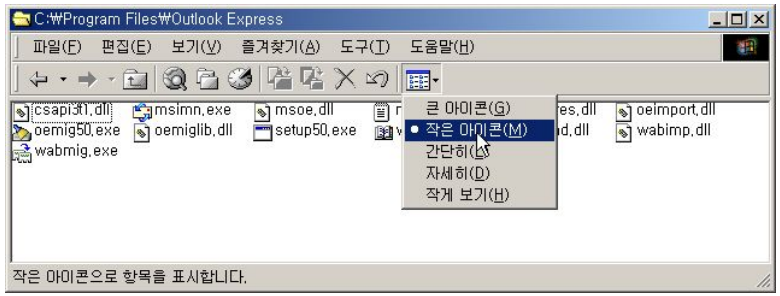
- 리스트 컨트롤(=리스트 뷰 컨트롤)
  - 이미지와 텍스트를 이용하여 다양한 형태로 정보 표시

- 표시 방법
  - 아이콘 보기(Icon View)
    - 큰 아이콘과 아래쪽의 텍스트로 *항목* 표시



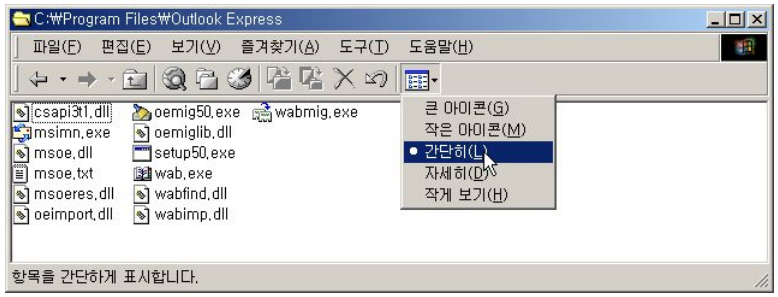
리스트 컨트롤

- 표시 방법
  - 작은 아이콘 보기(Small Icon View)
    - 작은 아이콘과 오른쪽의 텍스트로 *항목* 표시



마우스로 드래그하여 항목 위치 변경 가능

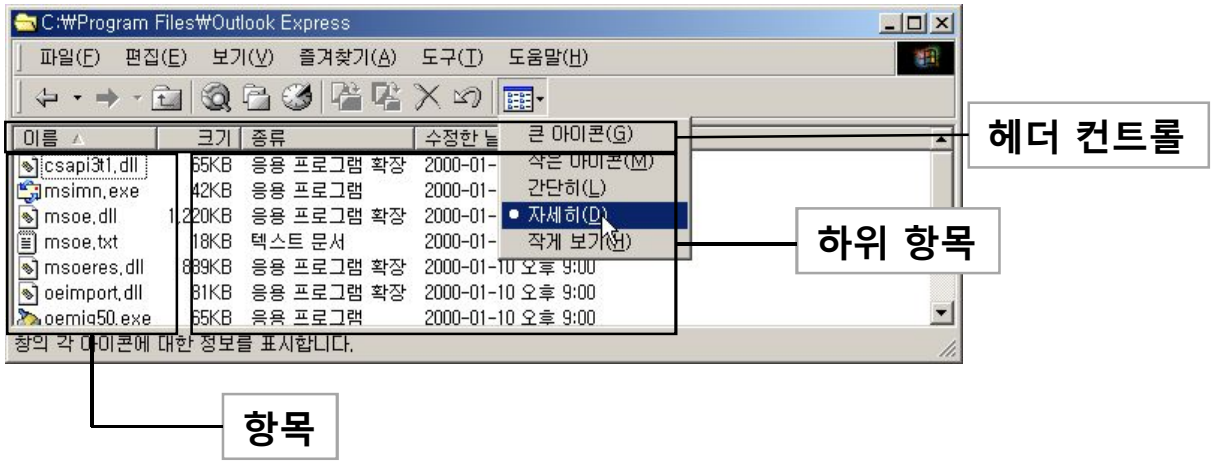
  - 목록 보기(List View)
    - 작은 아이콘과 오른쪽의 텍스트로 *항목* 표시



항목이 항상 자동 정렬됨

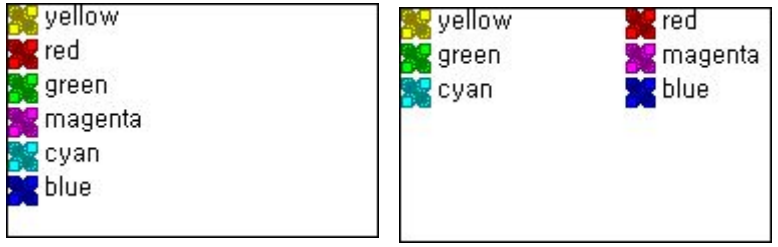
리스트 컨트롤

- 표시 방법
  - 보고서 보기(Report View)
    - 맨 왼쪽 열은 작은 아이콘과 오른쪽의 텍스트로 *항목* 표시
    - 나머지 열(*하위 항목*(Subitem)이라 부름)은 응용 프로그램이 자유롭게 필요한 정보를 표시
    - 맨 위쪽에 *헤더*(Header) *컨트롤*을 옵션으로 붙여 각 열의 폭 조절과 항목 정렬 가능



리스트 컨트롤

- 리스트 컨트롤 스타일



리스트 컨트롤 클래스

- 리스트 컨트롤 생성과 초기화
  ① CListCtrl 객체 선언 후 CListCtrl::Create() 함수로 리스트 컨트롤 생성
  ② CImageList 객체 선언 후 CImageList::Create(), CImageList::Add() 등의 함수로 이미지 리스트 초기화
  ③ CListCtrl::SetImageList() 함수로 리스트 컨트롤에서 사용할 이미지 리스트 설정
  ④ CListCtrl::InsertColumn() 함수로 보고서 보기에서 표시할 헤더 초기화
  ⑤ CListCtrl::InsertItem() 함수로 항목 추가
  ⑥ CListCtrl::SetItemText() 함수로 하위 항목 채움

리스트 컨트롤 클래스

- 예제 코드

```
#define IDC_LIST1 100

CListCtrl m_list;

// ① 리스트 컨트롤 생성
m_list.Create(WS_CHILD|WS_VISIBLE|WS_BORDER|LVS_REPORT,
    CRect(0, 0, 300, 100), this, IDC_LIST1);

// ② 이미지 리스트 생성과 초기화
CImageList ilLarge, ilSmall;
ilLarge.Create(32, 32, ILC_COLOR4, 1, 1);
ilSmall.Create(16, 16, ILC_COLOR4, 1, 1);
ilLarge.Add(AfxGetApp()->LoadIcon(IDI_ICON1));
ilSmall.Add(AfxGetApp()->LoadIcon(IDI_ICON1));
```

이름	성적	학점
임현선	90	A0
김기태	95	A+
서유경	70	B0

리스트 컨트롤 클래스

- 예제 코드


```
// ③ 이미지 리스트 설정
m_list.SetImageList(&ilLarge, LVSIL_NORMAL);
m_list.SetImageList(&ilSmall, LVSIL_SMALL);
ilLarge.Detach();
ilSmall.Detach();

// ④ 헤더 초기화
m_list.InsertColumn(0, _T("이름 "), LVCFMT_LEFT, 100, 0);
m_list.InsertColumn(1, _T("성적 "), LVCFMT_LEFT, 100, 1);
m_list.InsertColumn(2, _T("학점 "), LVCFMT_LEFT, 100, 2);

// ⑤ 항목 추가
m_list.InsertItem(0, _T("임현선 "), 0);
m_list.InsertItem(1, _T("김기태 "), 0);
m_list.InsertItem(2, _T("서유경 "), 0);
```

리스트 컨트롤 클래스

- 예제 코드

```
// ⑥ 하위 항목 추가
m_list.SetItemText(0, 1, _T("90 “));
m_list.SetItemText(0, 2, _T("A0 “));
m_list.SetItemText(1, 1, _T("95 “));
m_list.SetItemText(1, 2, _T("A+ “));
m_list.SetItemText(2, 1, _T("70 “));
m_list.SetItemText(2, 2, _T("B0 “));
```

리스트 컨트롤 클래스

- 단계별 핵심 함수 - ① 리스트 컨트롤 생성

```
BOOL CListCtrl::Create(DWORD dwStyle, const RECT& rect, CWnd* pParentWnd, UINT nID);
```

  - dwStyle: 일반 윈도우 스타일(WS_*)과 리스트 컨트롤 스타일(LVS_*)을 조합하여 컨트롤 스타일을 지정
  - rect: 컨트롤의 크기와 위치
  - pParentWnd: 부모 윈도우
  - nID: 컨트롤 ID

- 단계별 핵심 함수- ② 이미지 리스트 생성과 초기화

리스트 컨트롤 클래스

- 단계별 핵심 함수 - ③ 이미지 리스트 설정

```
CImageList* CListCtrl::SetImageList(CImageList* pImageList, int nImageListType);
```

  - pImageList: 리스트 컨트롤에서 사용할 이미지 리스트
  - nImageListType: 이미지 리스트에 포함된 이미지의 용도를 나타내는 상수값

nImageListType	용도
LVSIL_NORMAL	아이콘 보기
LVSIL_SMALL	작은 아이콘 보기, 목록 보기, 보고서 보기
LVSIL_STATE	상태 이미지

리스트 컨트롤 클래스

- 단계별 핵심 함수 - ④ 헤더 초기화

```
int CListCtrl::InsertColumn(int nCol, LPCTSTR lpszColumnHeading,
                                  ①                  ②
        int nFormat=LVCFMT_LEFT, int nWidth=-1, int nSubItem=-1);
            ③                        ④              ⑤
```

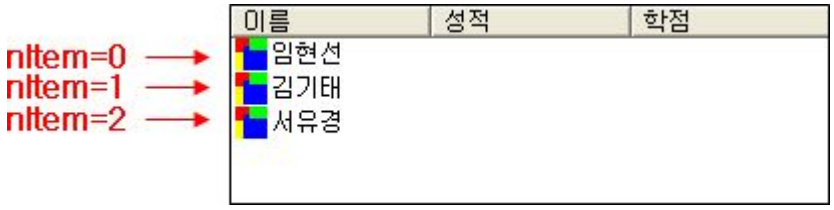
  - nCol: 초기화할 열 번호(0부터 시작)
  - lpszColumnHeading: 헤더에 표시할 텍스트
  - nFormat: 헤더에 표시할 텍스트의 정렬 방식
    - LVCFMT_LEFT(왼쪽), LVCFMT_RIGHT(오른쪽), LVCFMT_CENTER(가운데)
  - nWidth: 열의 폭(픽셀 단위)
  - nSubItem: 연관된 하위 항목의 인덱스
    - 보통 nCol과 같은 값을 사용

리스트 컨트롤 클래스

- 단계별 핵심 함수 - ⑤ 항목 추가

```
int CListCtrl::InsertItem(int nItem, LPCTSTR lpszItem, int nImage);
                          ①            ②             ③
```

  - nItem: 항목 인덱스(0부터 시작)



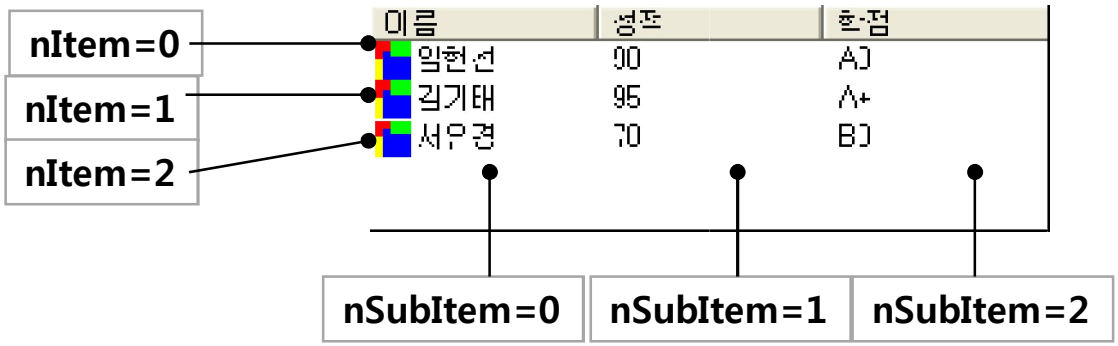
  - lpszItem: 항목에 표시할 텍스트
  - nImage: 항목에 표시할 이미지를 이미지 리스트에 포함된 이미지의 인덱스 값으로 지정

리스트 컨트롤 클래스

- 단계별 핵심 함수 - ⑥ 하위 항목 추가

```
BOOL CListCtrl::SetItemText(int nItem, int nSubItem, LPCTSTR lpszText);
                            ①          ②           ③
```

  - nItem: 항목 인덱스(0부터 시작)
  - nSubItem: 하위 항목 인덱스(1부터 시작). 0을 사용하면 항목 텍스트 변경 가능
  - lpszText: 하위 항목(nSubItem>0) 또는 항목(nSubItem=0)에 표시할 텍스트



리스트 컨트롤 클래스

- 표준 스타일(LVS_*) 변경하기

```
BOOL CWnd::ModifyStyle(DWORD dwRemove, DWORD dwAdd,UINT nFlags = 0);
```

  - dwRemove: 제거할 스타일
  - dwAdd: 추가할 스타일
  - nFlags: 컨트롤의 크기나 위치를 제어(대개 기본값 사용)

```
// 아이콘 보기로 변경한다.
m_list.ModifyStyle(LVS_TYPEMASK, LVS_ICON);
// 레이블 편집을 가능하게 한다.
m_list.ModifyStyle(0, LVS_EDITLABELS);
```

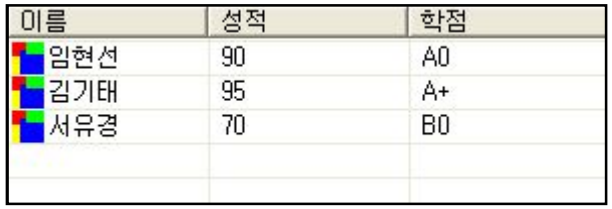
리스트 컨트롤 클래스

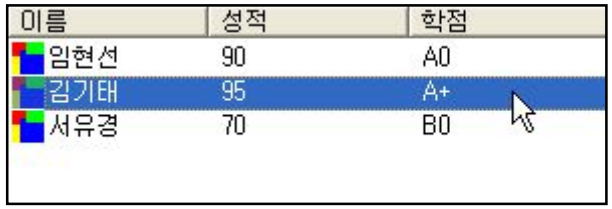
- 확장 스타일(LVS_EX_*) 변경하기

```
DWORD CListCtrl::SetExtendedStyle(DWORD dwNewStyle);
```

  - dwNewStyle : 새로 적용할 확장 스타일

```
// 보고서 보기에서 격자 무늬를 표시한다.
m_list.SetExtendedStyle(m_list.GetExtendedStyle()|LVS_EX_GRIDLINES);
// 보고서 보기에서 항목을 선택하면 줄 전체가 반전된다.
m_list.SetExtendedStyle(m_list.GetExtendedStyle()|LVS_EX_FULLROWSELECT);
```





리스트 컨트롤 클래스

- 선택 항목 알아내기 ①
  - 마우스나 키보드로 새로운 항목을 선택한 경우

```
// 메시지맵
ON_NOTIFY(LVN_ITEMCHANGED, IDC_LIST1, &CListCtrlTest2View::OnLvnItemchangedL
    ist1)

// 메시지 핸들러
void CListCtrlTest2View::OnLvnItemchangedList1(NMHDR *pNMHDR, LRESULT *pResu
    lt)
{
    LPNMLISTVIEW pNMLV = reinterpret_cast<LPNMLISTVIEW>(pNMHDR);


    if(pNMLV->uChanged & LVIF_STATE){ // 상태가 변경되었다면...
      if(pNMLV->uNewState & (LVIS_SELECTED|LVIS_FOCUSED)){
        CString str = m_list.GetItemText(pNMLV->iItem, 0);
        AfxGetMainWnd()->SetWindowText(str);
      }
    }
    *pResult = 0;
}
```

리스트 컨트롤 클래스

- 선택 항목 알아내기 ②
  - 마우스로 항목을 더블 클릭한 경우

```
// 메시지 맵
ON_NOTIFY(NM_DBLCLK, IDC_LIST1, &CListCtrlTest2View::OnNMDblclkList1)

// 메시지 핸들러
void CListCtrlTest2View::OnNMDblclkList1(NMHDR *pNMHDR, LRESULT *pResult)
{
    LPNMITEMACTIVATE pNMItemActivate = reinterpret_cast<LPNMITEMACTIVATE>(pN
    MHDR);

    if(pNMItemActivate->iItem != -1){
     CString str = m_list.GetItemText(pNMItemActivate->iItem, 0);
     MessageBox(str);
    }

    *pResult = 0;
}
```

리스트 컨트롤 클래스

- 선택 항목 알아내기 ③
  - 선택된 여러 개의 항목을 조사하는 경우

```
void CListCtrlTest2View::OnPrintIndex()
{
    POSITION pos = m_list.GetFirstSelectedItemPosition();
    if(pos != NULL){
      while(pos){
        int nItem = m_list.GetNextSelectedItem(pos);
        TRACE(_T("Item %d selected.Wn"), nItem);
      }
    }
}
```

리스트 컨트롤 클래스

- 항목 추가와 삭제
  - 항목 추가
    - CListCtrl::InsertItem() 함수
  - 항목 삭제
    - 항목 하나 삭제: CListCtrl::DeleteItem()
    - 항목 전체 삭제: CListCtrl::DeleteAllItems()

실습

학습정리

- CView는 모든 뷰 클래스의 베이스 클래스이다.
- CView를 상속한 CCtrlView는 모든 컨트롤 기반 뷰 클래스의 베이스 클래스이며, CScrollView는 자동화된 스크롤 기능을 제공한다.
- CScrollView를 상속한 CFormView는 대화상자를 기반으로 하는 뷰 클래스로, 컨트롤을 이용해 시각적으로 화면을 디자인할 수 있다.
- CCtrlView를 상속한 CListView 클래스는 도큐먼트/뷰 구조 응용 프로그램에서 리스트 컨트롤을 이용해 뷰의 기능을 제공한다.
- CListView 클래스가 제공하는 대부분의 기능은 리스트 컨트롤을 통해서만 사용할 수 있으므로 CListView::GetListCtrl() 함수가 제공된다.
- 리스트 컨트롤 초기화는 리스트 컨트롤 생성 -> 이미지 리스트 생성 -> 헤더 초기화 -> 항목 추가 -> 하위 항목 추가 순으로 진행된다.