다양한 뷰 클래스

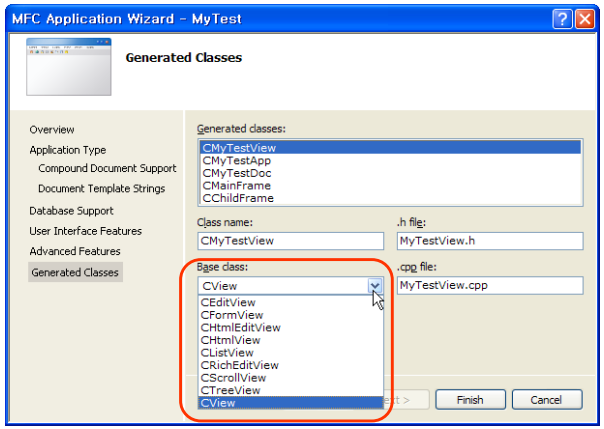
HCI Programming 2 (321190)
2008년 가을학기
12/2/2008
박경신

Overview

- MFC 뷰 클래스의 특징과 용도
- 다양한 뷰 클래스를 다루는 방법

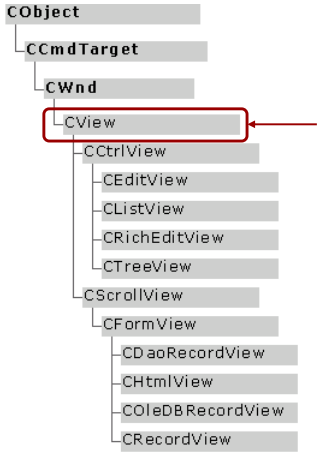
뷰 클래스 종류

- 뷰 클래스 선택



뷰 클래스 종류

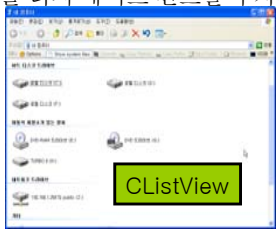
- MFC 클래스 계층도

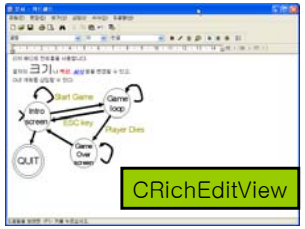


✓ 모든 뷰 클래스의 베이스 클래스
✓ 파생 클래스에 공통의 인터페이스 제공

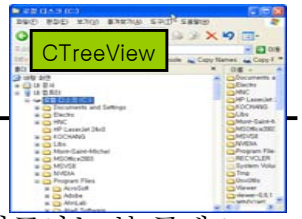
뷰 클래스 종류

- CCtrlView
  - 뷰 객체의 클라이언트 영역이 컨트롤로 구성된 클래스
  - 컨트롤 기반 뷰 클래스를 위한 공통 인터페이스와 기능 제공
  - CEditView: 편집 컨트롤을 기반으로 한 뷰 클래스
    - 뷰 영역 전체를 편집 컨트롤이 차지
  - CListView: 리스트 컨트롤을 기반으로 한 뷰 클래스
    - 뷰 영역 전체를 리스트 컨트롤이 차지
  - CRichEditView: 리치 에디트 컨트롤을 기반으로 한 뷰 클래스
    - 뷰 영역 전체를 리치 에디트 컨트롤이 차지





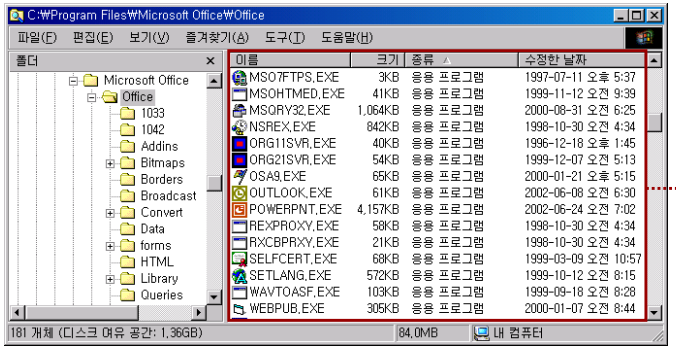
뷰 클래스 종류



  - CTreeView: 트리 컨트롤 기반 뷰 클래스
  - 뷰 영역 전체를 트리 컨트롤이 차지
- CScrollView: 자동화된 스크롤 기능을 제공하는 뷰 클래스
  - CFormView: 대화상자 기반 뷰 클래스
    - 뷰 영역 전체를 대화상자가 차지
    - 컨트롤을 이용하여 시각적으로 화면을 디자인
  - CHtmlView: Internet Explorer가 제공하는 웹 브라우저 기반 뷰 클래스
    - 뷰 영역 전체를 웹 브라우저 컨트롤이 차지
  - CRecordView, CDaoRecordView, COleDBRecordView
    - DB에서 가져온 레코드를 화면에 표시하는 기능을 제공

리스트 뷰

- CListView 클래스
  - 리스트 컨트롤을 이용한 사용자 인터페이스 제공



리스트 컨트롤

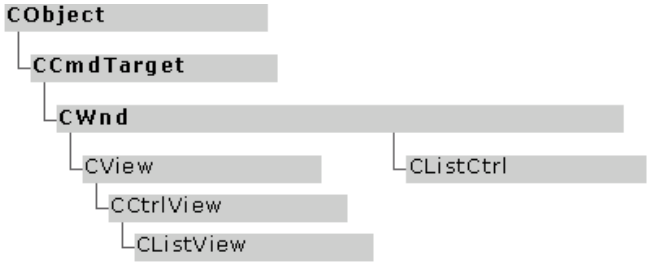
리스트 뷰

- CListView 클래스 사용 예
  - CListView 클래스가 제공하는 대부분의 기능은 리스트 컨트롤을 통해서 사용 – 즉, 리스트 컨트롤 객체를 얻고, 리스트 컨트롤을 다루듯이 코딩해야 함

```
// 리스트 컨트롤 객체에 대한 참조값(CListCtrl 타입)을 얻는다
CListCtrl& list = GetListCtrl();

// 참조 변수를 이용하여 리스트 컨트롤을 다룬다
list.SetImageList(...);
list.InsertColumn(...);
...
```

리스트 뷰

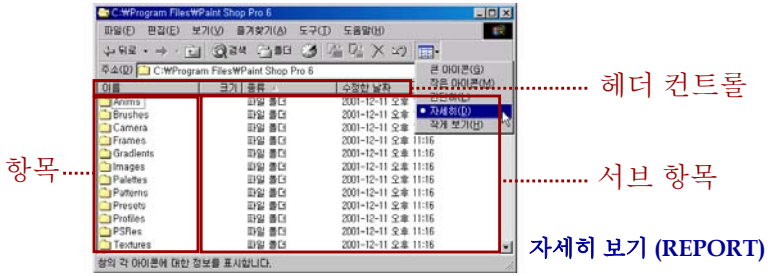
- MFC 클래스 계층도



리스트 컨트롤 (List Control)

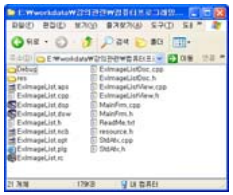
- 리스트 컨트롤 = 리스트 뷰 컨트롤
  - 이미지와 텍스트를 이용하여 다양한 형태로 정보를 표시하는 용도로 사용
- 표시 방법
  - 아이콘 보기 (ICON)
    - 32*32 크기의 아이콘과 텍스트로 항목 표시
  - 작은 아이콘 보기 (SMALLICON)
    - 16*16 크기의 아이콘과 텍스트로 항목 표시
  - 목록 보기 (LIST)
    - 16*16 크기의 아이콘과 텍스트로 항목 표시
  - 자세히 보기 (REPORT)
    - 컬럼 제목이 상단에 있고 아이템들이 컬럼 위치에 설정되어 있는 상태
    - 서브 항목으로 부가적인 정보 표시
    - 헤더 컨트롤로 각 열의 폭 조절과 항목 정렬 가능

리스트 컨트롤 (List Control)



큰 아이콘 보기 (ICON)

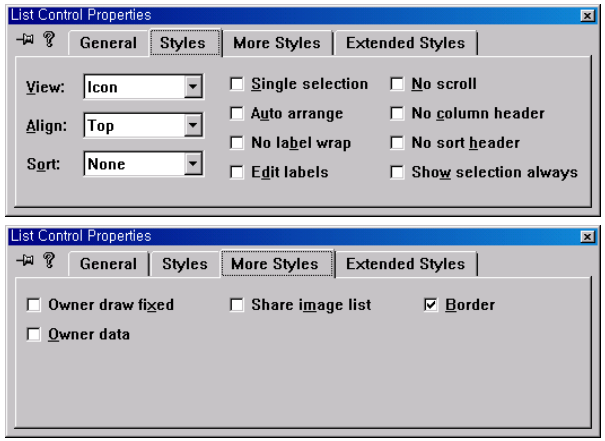
작은 아이콘 보기 (SMALLICON)



간단히 보기 (LIST)

리스트 컨트롤 (List Control)

- 리스트 컨트롤 스타일



리스트 컨트롤 (List Control)

- 리스트 컨트롤 스타일

리스트 컨트롤 스타일	의미
View	리스트 컨트롤 표시 방법: Icon, Small Icon, List, Report
Align	Top: 항목을 표시할 때 위쪽부터 차례로 채움 Left: 항목을 표시할 때 왼쪽 열부터 차례로 채움
Sort	항목 정렬 방식: None, Ascending (오름차순), Descending (내림 차순)
Single Selection	마우스로 하나의 항목만 선택할 수 있게 함
Auto arrange	Icon, Small Icon 보기에서 항목이 자동 정렬
No label wrap	Icon 보기에서 텍스트를 한 줄로 표시
Edit labels	텍스트를 바로 수정가능
No scroll	스크롤바가 나타나지 않게 함
No column header	Report 보기에서 헤더가 표시되지 않음
No sort header	헤더를 마우스로 클릭할 수 없게 함

리스트 컨트롤 (List Control)

- 리스트 컨트롤 스타일

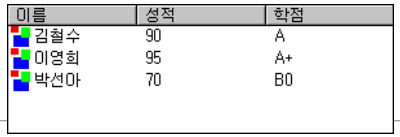
리스트 컨트롤 스타일	의미
Show selection always	리스트 컨트롤이 키보드 포커스를 일더라도 선택된 항목이 계속 반전된 상태로 남아있게 함
Owner draw fixed	Report 보기에서 부모 윈도우가 리스트 컨트롤 항목을 직접 그림 – 부모 윈도우는 WM_DRAWITEM 메시지 처리
Owner data	부모 윈도우가 컨트롤이 필요로 하는 데이터를 공급함 – 대용량 데이터를 리스트 컨트롤에 표시할 때 적합
Share image list	여러 개의 컨트롤이 하나의 이미지 리스트를 공유할 경우 이 스타일을 설정
Border	컨트롤 주위에 경계선을 그림

리스트 컨트롤 클래스

- 리스트 컨트롤 생성과 초기화
  - 리스트 컨트롤 객체 생성
    - CListCtrl 객체 선언 후 CListCtrl::**Create**()로 리스트 컨트롤 생성
  - 이미지 리스트 컨트롤 객체 생성
    - CImageList 객체 선언 후 CImageList::**Create**(), CImageList::**Add**() 등을 이용하여 생성과 초기화
  - 리스트 컨트롤에 이미지 리스트 설정
    - CListCtrl::**SetImageList**()로 리스트 컨트롤에서 사용할 이미지 리스트 설정
  - 리스트 컨트롤에 자세히 (Report) 보기에서 표시할 헤더 초기화
    - CListCtrl::**InsertColumn**()
  - 리스트 컨트롤에 항목 추가
    - CListCtrl::**InsertItem**()으로 항목 추가
  - 리스트 컨트롤의 하위항목 초기화
    - CListCtrl::**SetItemText**()로 하위 항목 초기화

리스트 컨트롤 클래스

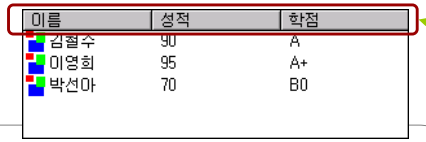
- 예제 코드




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

// ② CImageList 객체 선언과 이미지 리스트 생성, 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));
```

리스트 컨트롤 클래스

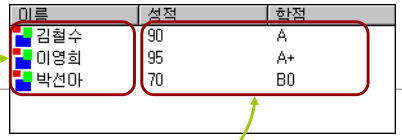
- 예제 코드




```
// 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, "이름", LVCFMT_LEFT, 100);
m_list.InsertColumn(1, "성적", LVCFMT_LEFT, 100);
m_list.InsertColumn(2, "학점", LVCFMT_LEFT, 100);
```


리스트 컨트롤 클래스

- 예제 코드




```
// ⑤ 항목 추가
m_list.InsertItem(0, "김철수", 0);
m_list.InsertItem(1, "이영희", 0);
m_list.InsertItem(2, "박선아", 0);

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


리스트 컨트롤 클래스

- CListCtrl::Create 리스트 컨트롤 생성

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

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

리스트 컨트롤 클래스

- CListCtrl::SetImageList 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

- ClistCtrl::InsertColumn 컬럼 헤더 설정

```
int CListCtrl::InsertColumn (int nCol, const LVCOLUMN* pColumn);
int CListCtrl::InsertColumn (int nCol, LPCTSTR lpszColumnHeading, int
    nFormat = LVCFMT_LEFT, int nWidth = -1, int nSubItem = -1);
```

  - nCol: 열 번호를 나타내며 0부터 시작
  - lpszColumnHeading: 헤더 컨트롤에 표시할 텍스트
  - nFormat: 헤더 컨트롤에 표시할 텍스트의 정렬 방식을 나타내며 LVCFMT_LEFT(왼쪽), LVCFMT_RIGHT(오른쪽), LVCFMT_CENTER(가운데) 중 하나를 선택
  - nWidth: 열의 폭(픽셀 단위)
  - nSubItem: 연관된 하위 항목의 인덱스를 나타내며, 보통 nCol과 같은 값을 사용

리스트 컨트롤 클래스

- LVCOLUMN : 리스트 컨트롤의 컬럼 구조체

```
typedef struct _LVCOLUMN {
  UINT mask;
  int fmt;
  int cx;
  LPTSTR pszText;
  int cchTextMax;
  int iSubItem;
} LVCOLUMN, FAR *LPLVCOLUMN;
```

  - mask : 구조체 내의 각 멤버가 유효한지 확인하는 변수
    LVCF_FMT , LVCF_SUBITEM , LVCF_TEXT ,
    LVCF_WIDTH, LVCF_IMAGE, LVCF_ORDER
  - fmf : 컬럼 헤더와 서브 아이템의 정렬방식

리스트 컨트롤 클래스

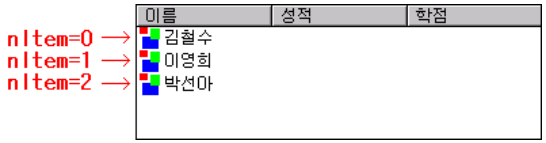
```
LV_COLUMN lvColumn;
char* list[3] = {"이름", "성적", "학점"};
int nWidth[3] ={100, 50, 50};

// 컬럼 구조체를 이용하여 리스트 컨트롤 컬럼 설정
for( int i = 0; i < 3; i++ )
{
        lvColumn.mask = LVCF_FMT | LVCF_SUBITEM |
                        LVCF_TEXT | LVCF_WIDTH;
        lvColumn.fmt = LVCFMT_LEFT;
        lvColumn.pszText = list[i];
        lvColumn.iSubItem = i;
        lvColumn.cx = nWidth[i];
        m_list.InsertColumn(i, &lvColumn);
}
```


리스트 컨트롤 클래스

- CListCtrl::InsertItem 리스트 항목 추가

```
int CListCtrl::InsertItem (LVITEM* pItem);
int CListCtrl::InsertItem (int nItem, LPCTSTR lpszItem, int nImage);
```

  - nItem: 항목 인덱스이며 0부터 시작



  - lpszItem: 항목에 표시할 텍스트
  - nImage: 항목에 표시할 이미지로서 이미지 리스트에서의 인덱스 값을 사용

리스트 컨트롤 클래스

- CListCtrl::SetItem 리스트 항목의 속성 설정

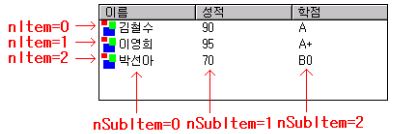
```
int CListCtrl::SetItem (LVITEM* pItem);
int CListCtrl::SetItem (int nItem, int nSubItem, UINT nMask, LPCTSTR
    lpszItem, int nImage, UINT nState, UINT nStateMask, LPARAM
    lParam, int nIndent = -1);
```

  - nItem: 항목 인덱스
  - nSubItem: 하위항목 인덱스
  - nMask: 유효항목
  - lpszItem: 항목에 표시할 텍스트
  - nImage: 항목에 표시할 이미지로서 이미지 리스트에서의 인덱스 값을 사용
  - nState: 항목의 상태
  - nStateMask: 유효한 상태비트에 대한 마스크
  - lParam: 항목관련 변수
  - nIndent: Indentation의 크기 (픽셀 단위)

리스트 컨트롤 클래스

- LVITEM : 리스트 컨트롤의 항목 속성 구조체

```
typedef struct _LVITEM {
  UINT mask;     //유효항목
  int iItem;     //항목인덱스
  int iSubItem;  //하위항목인덱스
  UINT state;    //항목의 상태
  UINT stateMask;  //유효한 상태비트에 대한 마스크
  LPTSTR pszText; //항목의 텍스트
  int cchTextMax;  //텍스트의 최대 길이
  int iImage;      //이미지 인덱스
  LPARAM lParam; //항목관련 변수
} LVITEM, FAR *LPLVITEM;
```

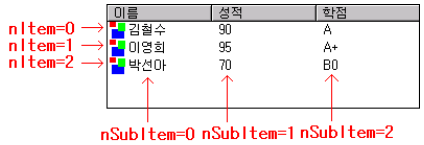


리스트 컨트롤 클래스

- CListCtrl::SetItemText 하위 항목 추가

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

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



리스트 컨트롤 클래스

- 리스트 항목 삭제
  - 하나의 항목을 삭제하려면 CListCtrl::DeleteItem() 사용
  - 모든 항목을 삭제하려면 CListCtrl::DeleteAllItems() 사용

```
// 첫 번째 항목 삭제
m_list.DeleteItem(0);
// 모든 항목 삭제
m_list.DeleteAllItems();
```

리스트 컨트롤 클래스

- 표준 스타일(LVS_*) 변경하기

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

  - dwRemove: 제거할 스타일
  - dwAdd: 추가할 스타일
  - nFlags: 기본값 사용

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

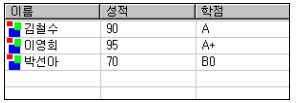

리스트 컨트롤 클래스

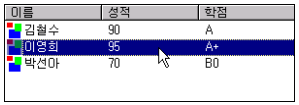
- CListCtrl::SetExtendedStyle 확장 스타일(LVS_EX_*) 변경하기

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

  - dwNewStyle : 새로 적용할 확장 스타일

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





리스트 컨트롤 클래스

- 리스트 컨트롤 메시지
  - 선택 항목 알아내기 - 마우스 또는 키보드로 새로운 항목을 선택한 경우

```
// 메시지맵
ON_NOTIFY(LVN_ITEMCHANGED, IDC_LIST1, OnItemchangedList1)

// 통지 메시지 핸들러
void CExListCtrlView::OnItemchangedList1(NMHDR* pNMHDR,
   LRESULT* pResult)
{
   NMLISTVIEW* pNMLV = (NMLISTVIEW*)pNMHDR;
   if(pNMLV->uChanged & LVIF_STATE) {
      if(pNMLV->uNewState & (LVIS_SELECTED|LVIS_FOCUSED)) {
```


리스트 컨트롤 클래스

- 리스트 컨트롤 메시지
  - 선택 항목 알아내기 - 마우스 또는 키보드로 새로운 항목을 선택한 경우

```
         CString str = m_list.GetItemText(pNMLV->iItem, 0);
         AfxGetMainWnd()->SetWindowText(str);
      }
   }

   *pResult = 0;
}
```

리스트 컨트롤 클래스

- NMLISTVIEW 구조체

```
typedef struct tagNMLISTVIEW {
  NMHDR hdr;
  int iItem;   // 항목선택이 바뀌면 새로 선택한 항목의 인덱스를 가짐
  int iSubItem;
  UINT nNewState; // 항목의 새로운 상태
  UINT uOldState;  // 항목의 이전 상태
  UINT uChanged;  // 항목의 속성 중 어느 것이 변경됐는지 나타냄
  POINT ptAction;
  LPARAM lParam;
} NMLISTVIEW, *LPNMLISTVIEW;
```


리스트 컨트롤 클래스

- 리스트 컨트롤 메시지
  - 선택 항목 알아내기 - 마우스로 항목을 더블 클릭한 경우

```
// 메시지맵
ON_NOTIFY(NM_DBLCLK, IDC_LIST1, OnDblclkList1)

// 통지 메시지 핸들러
void CExListCtrlView::OnDblclkList1(NMHDR* pNMHDR, LRESULT*
    pResult) {
   NMITEMACTIVATE* pNMIA = (NMITEMACTIVATE*)pNMHDR;
   if(pNMIA->iItem != -1) {
      CString str = m_list.GetItemText(pNMIA->iItem, 0);
      MessageBox(str);
   }
   *pResult = 0;
}
```


리스트 컨트롤 클래스

- NMITEMACTIVE 구조체

```
typedef struct tagNMITEMACTIVE {
  NMHDR hdr;
  int iItem;   //항목선택이 바뀌면 새로 선택한 항목의 인덱스를 가짐
  int iSubItem;
  UINT nNewState; //항목의 새로운 상태
  UINT uOldState;  //항목의 이전 상태
  UINT uChanged;  //항목의 속성 중 어느 것이 변경됐는지 나타냄
  POINT ptAction;
  LPARAM lParam;
} NMITEMACTIVE, *LPNMITEMACTIVE;
```

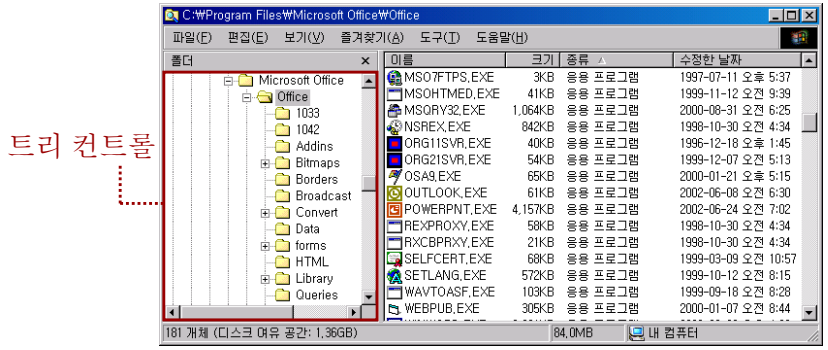

리스트 컨트롤 클래스

- 리스트 컨트롤 메시지
  - 선택된 다수 개의 항목을 조사하는 경우

```
void CExListCtrlView::PrintSelectedItemIndex()
{
   POSITION pos = m_list.GetFirstSelectedItemPosition();
   if(pos != NULL) {
      while(pos) {
         int nItem = m_list.GetNextSelectedItem(pos);
         TRACE("항목 %d번이 선택되었습니다.\n", nItem);
      }
   }
}
```

트리 뷰

- CTreeView 클래스
  - 트리 컨트롤을 이용한 사용자 인터페이스 제공



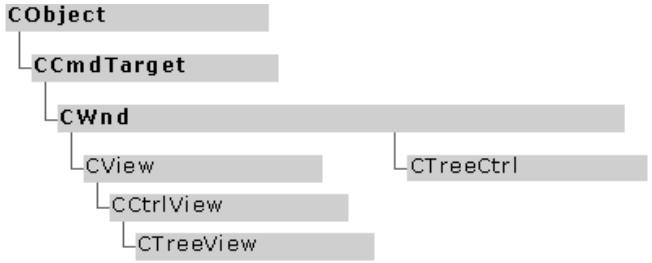
트리 뷰

- CTreeView 클래스 사용 예

```
// 트리 컨트롤 객체에 대한 참조값을 얻는다
CTreeCtrl& tree = GetTreeCtrl();

// 참조 변수를 이용하여 트리 컨트롤을 다룬다
tree.SetImageList(...);
tree.InsertItem(...);
...
```


트리 뷰

- MFC 클래스 계층도

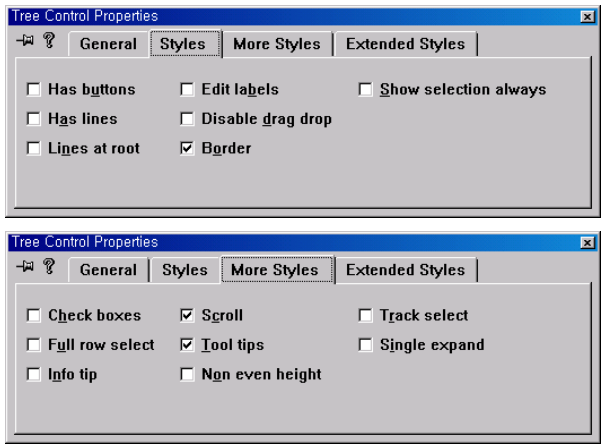


트리 컨트롤 (Tree Control)

- 트리 컨트롤 = 트리 뷰 컨트롤
  - 이미지와 텍스트를 이용하여 계층적인 형태로 정보를 표시하는 용도로 사용
- 용어
  - 항목
    - 트리 컨트롤에 표시되는 하나의 정보
  - 부모 항목
    - 하나 이상의 하위 항목을 가진 항목
  - 자식 항목
    - 부모 항목에 딸린 하위 항목
  - 루트 항목
    - 계층 구조에서 최상위 항목
    - 루트 항목은 부모 항목을 가지지 않음

트리 컨트롤

- 트리 컨트롤 스타일



트리 컨트롤 클래스

- 트리 컨트롤 생성과 초기화
  - CTreeCtrl 객체 선언 후 CTreeCtrl::**Create**()로 트리 컨트롤 생성
  - CImageList 객체 선언 후 CImageList::**Create**(), CImageList::**Add**() 등을 이용하여 생성과 초기화
  - CTreeCtrl::**SetImageList**()로 트리 컨트롤에서 사용할 이미지 리스트 설정
  - CTreeCtrl::**InsertItem**()으로 항목 추가

트리 컨트롤 클래스

- 예제 코드


```
// ① CTreeCtrl 객체 선언과 트리 컨트롤 생성
m_tree.Create(WS_CHILD|WS_VISIBLE|WS_BORDER|
   TVS_HASBUTTONS|TVS_HASLINES|TVS_LINESATROOT,
   CRect(10, 10, 150, 150), this, 101);

// ② CImageList 객체 선언과 이미지 리스트 생성, 초기화
CImageList il;
il.Create(IDB_BITMAP1, 16, 1, RGB(255, 255, 255));
```


트리 컨트롤 클래스

- 예제 코드


```
// ③ 이미지 리스트 설정
m_tree.SetImageList(&il, TVSIL_NORMAL);
il.Detach();

// ④ 항목 추가
// 1레벨 초기화
HTREEITEM hSun = m_tree.InsertItem("태양", 0, 0, TVI_ROOT,
    TVI_LAST);
```

트리 컨트롤 클래스

- 예제 코드


```
// 2레벨 초기화
m_tree.InsertItem("수성", 1, 1, hSun, TVI_LAST);
m_tree.InsertItem("금성", 1, 1, hSun, TVI_LAST);
HTREEITEM hEarth = m_tree.InsertItem("지구", 1, 1, hSun,
    TVI_LAST);
HTREEITEM hMars = m_tree.InsertItem("화성", 1, 1, hSun,
    TVI_LAST);

// 3레벨 초기화
m_tree.InsertItem("달", 2, 2, hEarth, TVI_LAST);
m_tree.InsertItem("포보스", 2, 2, hMars, TVI_LAST);
m_tree.InsertItem("데이모스", 2, 2, hMars, TVI_LAST);
```


트리 컨트롤 클래스

- CTreeCtrl::Create

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

  - dwStyle: 컨트롤 스타일을 지정. 일반 윈도우 스타일(WS_*)과 트리 컨트롤 스타일(TVS_*)의 조합을 사용
  - rect: 컨트롤의 크기와 위치
  - pParentWnd: 부모 윈도우
  - nID: 컨트롤 ID

트리 컨트롤 클래스

- CTreeCtrl::SetImageList

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

  - pImageList: 트리 컨트롤에서 사용할 이미지 리스트
  - nImageListType: 이미지 리스트에 포함된 이미지의 용도. 항목을 나타내는 일반 이미지일 경우에는 TVSIL_NORMAL, 사용자 정의 상태 이미지일 경우에는 TVSIL_STATE를 사용

트리 컨트롤 클래스

- CTreeCtrl::InsertItem

```
HTREEITEM CTreeCtrl::InsertItem (LPCTSTR lpszItem, int nImage, int
    nSelectedImage, HTREEITEM hParent = TVI_ROOT, HTREEITEM
    hInsertAfter = TVI_LAST);
```

  - lpszItem: 항목에 표시할 텍스트
  - nImage: 항목에 표시할 이미지를 나타내며 이미지 리스트에서의 인덱스 값을 사용
  - nSelectedImage: 항목이 선택되면 표시할 이미지를 나타내며 이미지 리스트에서의 인덱스 값을 사용
  - hParent: 부모 항목을 나타내는 HTREEITEM 값이다. 루트 항목을 추가할 경우에는 TVI_ROOT를 사용
  - hInsertAfter: 항목을 추가할 위치를 나타내며 보통 다음 표의 값 중 하나를 사용. 항목 다음 위치에 추가하고 싶을 경우 그 항목을 나타내는 HTREEITEM 값을 사용

트리 컨트롤 클래스

- CTreeCtrl::InsertItem

```
HTREEITEM CTreeCtrl::InsertItem (LPCTSTR lpszItem, int nImage, int
    nSelectedImage, HTREEITEM hParent = TVI_ROOT, HTREEITEM
    hInsertAfter = TVI_LAST);
```

값	의미
TVI_FIRST	제일 앞쪽에 추가한다
TVI_LAST	제일 뒤쪽에 추가한다
TVI_ROOT	루트 항목으로 추가한다
TVI_SORT	철자 순으로 정렬이 되도록 추가한다

트리 컨트롤 클래스

- CTreeCtrl::InsertItem

```
HTREEITEM CTreeCtrl::InsertItem (LPTVINSERTSTRUCT
    lpInsertStruct);
HTREEITEM CTreeCtrl::InsertItem (UINT nMask, LPCTSTR lpszItem,
    int nImage, int nSelectedImage, UINT nState, UINT nStateMask,
    LPARAM lParam, HTREEITEM hParent, HTREEITEM
    hInsertAfter);
HTREEITEM CTreeCtrl::InsertItem (LPCTSTR lpszItem, HTREEITEM
    hParent = TVI_ROOT, HTREEITEM hInsertAfter = TVI_LAST);
HTREEITEM CTreeCtrl::InsertItem (LPCTSTR lpszItem, int nImage,
    int nSelectedImage, HTREEITEM hParent = TVI_ROOT,
    HTREEITEM hInsertAfter = TVI_LAST);
```


트리 컨트롤 클래스

- 스타일 변경하기

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

  - dwRemove: 제거할 스타일
  - dwAdd: 추가할 스타일
  - nFlags: 기본값 사용

트리 컨트롤 클래스

- CTreeCtrl::GetSelectedItem 선택 항목 알아내기

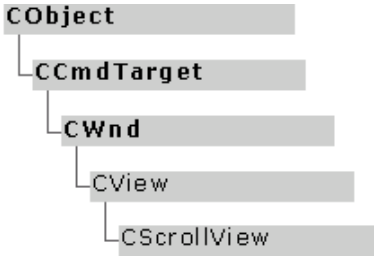
```
HTREEITEM hItem = m_tree.GetSelectedItem();
if(hItem != NULL) {
   CString str = m_tree.GetItemText(hItem);
   MessageBox(str);
}
```

- CTreeCtrl::DeleteItem 선택 항목 삭제

```
HTREEITEM hItem = m_tree.GetSelectedItem();
if(hItem != NULL) {
   m_tree.DeleteItem(hItem);
}
```

스크롤 뷰

- CScrollView 클래스
  - 자동화된 스크롤 기능 제공
- MFC 클래스 계층도



스크롤 뷰

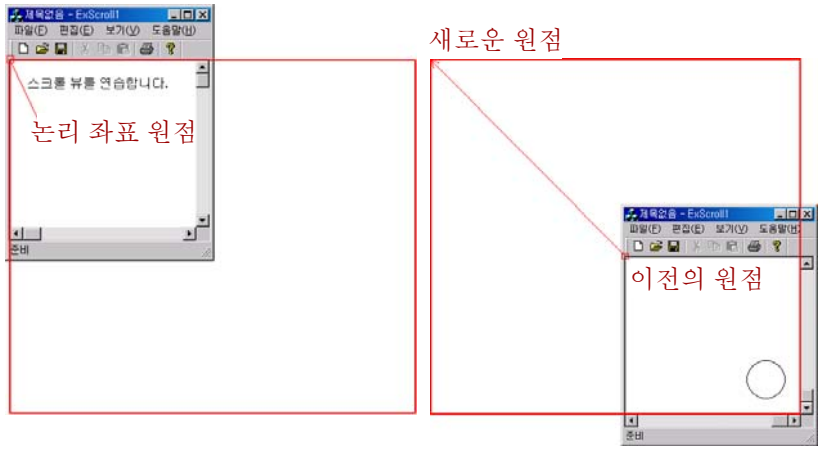
- CScrollView::SetScrollSizes

```
void CScrollView::SetScrollSizes (int nMapMode, SIZE sizeTotal, const
    SIZE& sizePage = sizeDefault, const SIZE& sizeLine = sizeDefault);
```

  - nMapMode: 매핑 모드. MM_ISOTROPIC, MM_ANISOTROPIC을 제외하고 모두 사용 가능
  - sizeTotal: 작업 영역의 전체 크기(논리 단위)
  - sizePage: 페이지 크기(논리 단위)
  - sizeLine: 한 줄의 크기(논리 단위)

스크롤 뷰

- 스크롤 바 위치에 따른 원점 이동



스크롤 뷰

- 스크롤 뷰 출력

```
void CExScrollView::OnDraw(CDC* pDC)
{
   pDC->TextOut(20, 20, CString("스크롤 뷰를 연습합니다."));
   pDC->Ellipse(1930, 1930, 1980, 1980);
}

void CExScrollView::MyDraw()
{
   CClientDC dc(this);
   OnPrepareDC(&dc);    // dc를 이용하여 출력한다
}
```

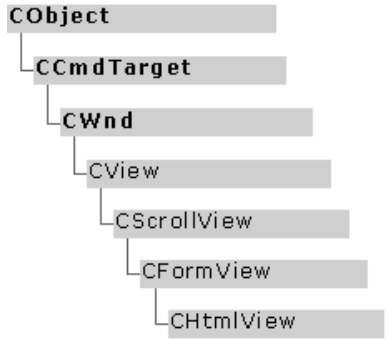
스크롤 뷰

- 스크롤 최적화

```
void CExScrollView::OnDraw(CDC* pDC)
{
    CExScrollDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    CRect rect;
    pDC->GetClipBox(&rect); // rect 영역만 다시 그린다
}
```


HTML 뷰

- CHtmlView 클래스
  - 웹 브라우저 컨트롤을 기반으로 한 뷰 클래스
- MFC 클래스 계층도



HTML 뷰

- 주요 함수

함수명	기능
Navigate, Navigate2	URL로 지정한 웹 문서 또는 로컬 파일을 열어서 표시
GoHome	인터넷 익스플로러에서 설정한 시작 페이지로 이동
GoBack	히스토리 목록에서 이전 항목으로 이동
GoForward	히스토리 목록에서 다음 항목으로 이동
Refresh, Refresh2	현재 표시하고 있는 내용을 다시 로드
Stop	현재 작업을 중지

HTML 뷰

- CHtmlView::Navigate2() 함수 사용 예

```
// 폴더와 파일을 표시한다
Navigate2(_T("C:\\"), NULL, NULL);
// 워드 문서를 열어서 편집한다
Navigate2(_T("C:\\test.doc"), NULL, NULL);
// 엑셀 문서를 열어서 편집한다
Navigate2(_T("C:\\test.xls"), NULL, NULL);
// 텍스트 문서를 표시한다
Navigate2(_T("C:\\test.txt"), NULL, NULL);
// GIF 포맷으로 저장된 그림을 표시한다
Navigate2(_T("C:\\test.gif"), NULL, NULL);
```

Practice

- SimpleMp3 프로그램을 이용하여 볼륨조절과 리스트에서 곡을 선택하여 재생하는 콘트롤을 구현하라.
  - 대화상자에 슬라이더를 추가하고, 슬라이더를 움직이면 볼륨조절
    - Add Member Variable을 control variable 로 해서 m_ctrlSlider로 추가
    - 대화상자의 OnInitDialog 함수에서 SetRange(0,255,FALSE) 등 슬라이더 콘트롤 초기화 (즉: 0=최소볼륨 & 255=최대볼륨)
    - OnVScroll(UINT nSBCode, UINT nPos, CScrollBar *pScrollBar)를 사용하여 현재 슬라이더 위치에 따라 Volume 함수를 사용하여 제어 (WM_VSCROLL 추가)
  - 대화상자에 리스트를 추가하고, 리스트에서 항목을 선택하면 mp3 파일 재생
    - Add Member Variable을 control variable 로 해서 m_ctrlList로 추가
    - 대화상자의 OnInitDialog 함수에서 InsertColumn 등 리스트 콘트롤 초기화
    - OnLvnItemchangedList(NMHDR *pNMHDR, LRESULT *pResult)를 사용하여 현재 리스트에서 항목이 선택되면 바로 mp3 파일을 재생 (Add Event Handler 사용)