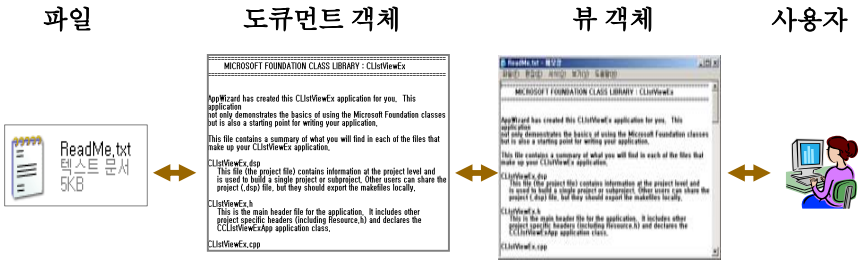
도큐먼트/뷰 구조

HCI Programming 2 (321190)
2008년 가을학기
11/25/2008
박경신

Overview

- 도큐먼트/뷰 구조
- 도큐먼트 템플릿 (Document Template)
- SDI (Single Document Interface) 응용 프로그램의 기본 구조
- MDI (Multiple Document Interface) 응용 프로그램의 기본 구조
- 명령 라우팅의 개념
- 여러 개의 뷰 또는 여러 개의 도큐먼트 타입을 생성 및 활용

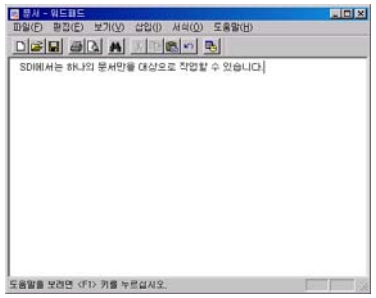
도큐먼트/뷰 구조

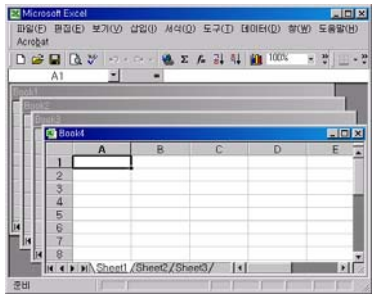


Class	역할
Document	데이터를 저장하거나 읽기 데이터의 변경 사항이 생기면 뷰의 화면을 갱신
View	데이터를 화면에 표시 사용자와의 상호 작용

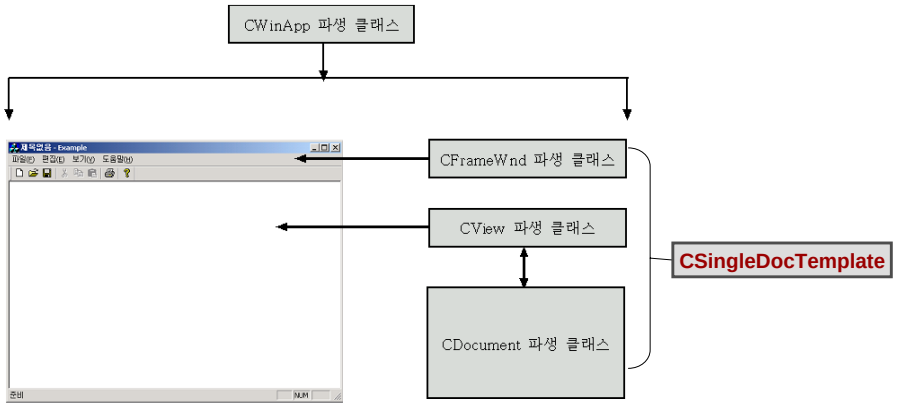
도큐먼트/뷰 구조

- SDI (Single-Document Interface)
  - 하나의 응용프로그램에서 어느 한 순간에 하나의 문서만을 대상으로 작업할 수 있는 사용자 인터페이스
- MDI (Multiple- Document Interface)
  - 하나의 응용프로그램에서 동시에 두 개 이상의 문서를 대상으로 작업할 수 있는 사용자 인터페이스

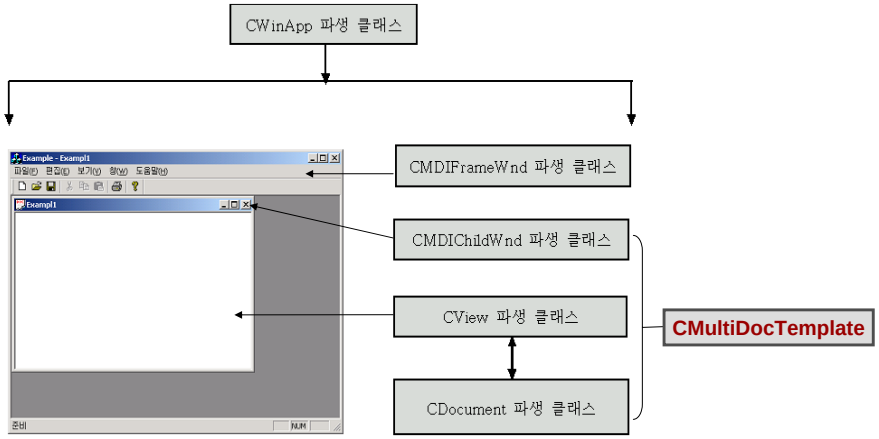




SDI (Single Document Interface)

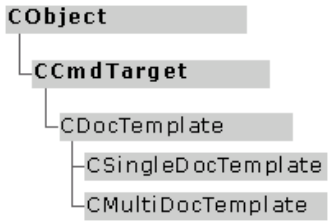


MDI (Multiple Document Interface)



Document Template

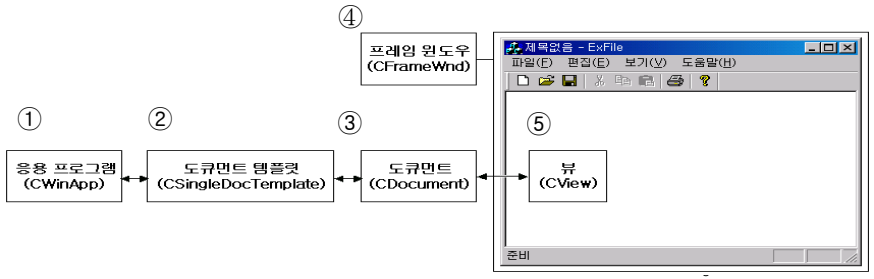
- 도큐먼트, 프레임 윈도우, 뷰 클래스 정보를 유지하는 클래스
- 응용프로그램 클래스의 InitInstance() 에서 생성
- MFC 클래스 계층도




```
BOOL CExSDIApp::InitInstance()
{
   ...
   CSingleDocTemplate* pDocTemplate;
   pDocTemplate = new CSingleDocTemplate(
      IDR_MAINFRAME,
      RUNTIME_CLASS(CExSDIDoc),
      RUNTIME_CLASS(CMainFrame),
      RUNTIME_CLASS(CExSDIView));
   AddDocTemplate(pDocTemplate);
   ...
}
```


주요 객체의 생성관계

생성 주체	생성되는 것
①응용 프로그램 객체	②도큐먼트 템플릿 객체
도큐먼트 템플릿 객체	③도큐먼트 객체, ④프레임 윈도우 객체
프레임 윈도우 객체	⑤뷰 객체



```
//(1)응용프로그램 클래스 (CWinApp 파생)
class CExSDIApp : public CWinApp  {…}
BEGIN_MESSAGE_MAP(CExSDIApp, CWinApp)
  // Standard file based document commands
  ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
  ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
END_MESSAGE_MAP()
BOOL CExSDIApp::InitInstance() {…
    CSingleDocTemplate* pDocTemplate;
    pDocTemplate = new CSingleDocTemplate( //도큐먼트 탬플릿 객체 생성
            IDR_MAINFRAME,                  //리소스 ID
            RUNTIME_CLASS(CExSDIDoc),       //Document
            RUNTIME_CLASS(CMainFrame),      //main SDI frame window
            RUNTIME_CLASS(CExSDIView));     //View
    AddDocTemplate(pDocTemplate); //응용프로그램객체에 탬플릿객체 추가

    //… command line을 분석 및 처리
    //생성된 메인 프레임 윈도우를 화면에 보이기
    m_pMainWnd->ShowWindow(SW_SHOW);
    m_pMainWnd->UpdateWindow();  //WM_PAINT 메시지 보내기
    return TRUE;
}
```

```
//(2)프레임 윈도우 클래스 (CFrameWnd 파생)
//동적 객체 생성 기능을 사용하기 위한 매크로
//MainFrm.h
class CMainFrame : public CFrameWnd
{
…
protected:
   DECLARE_DYNCREATE(CMainFrame)//동적 객체 생성을 위한 매크로 선언부
…
}
//MainFrm.cpp
 IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)
          //동적 객체 생성을 위한 매크로 구현부
…
```

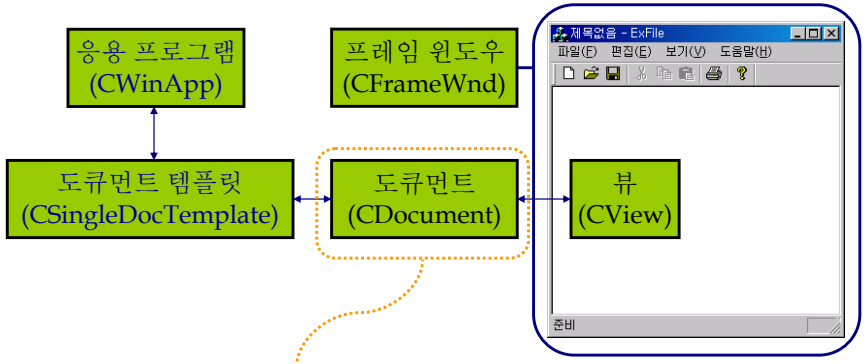


```
//(3)뷰 클래스 (CView 파생)
//ExSDIView.h
class CExSDIView : public CView {
protected:
    DECLARE_DYNCREATE(CExSDIView)
public:
    virtual void OnDraw(CDC* pDC);
…
}
//ExSDIView.cpp
IMPLEMENT_DYNCREATE(CExSDIView, CView)
void CExSDIView::OnDraw(CDC* pDC) {
    CExSDIDoc* pDoc = GetDocument(); //도큐먼트 객체 얻어오기
    CString str = pDoc->m_docStr;       //도큐먼트의 멤버변수 접근
    CFont newFont, *pOldFont=NULL;

    newFont.CreatePointFont(200, "Arial");
    pOldFont =pDC->SelectObject(&newFont);
    pDC->TextOut(100, 100, "SDI Test Program - View"); //view의 문자열 출력
    pDC->TextOut(100, 140, str);                  //document의 문자열 출력
 …
}
```

```
//(3)도큐먼트 클래스 (CDocument 파생)
//ExSDIDoc.h
class CExSDIDoc : public CDocument
{
protected:
    DECLARE_DYNCREATE(CExSDIDoc)
public:
    virtual BOOL OnNewDocument();  //[새파일] 선택 시 호출되는 가상함수
    virtual void Serialize(CArchive& ar); //파일을 읽거나 저장하기 위한 함수
    CString m_docStr;  //멤버변수
…
}

//ExSDIDoc.cpp
IMPLEMENT_DYNCREATE(CExSDIDoc, CDocument)
BOOL CExSDIDoc::OnNewDocument() { //새파일 선택시 초기화 코드 포함
    if (!CDocument::OnNewDocument()) return FALSE;
        return TRUE;
    m_docStr="SDI Test Program - DOC";   //멤버변수 초기화
}
void CExSDIDoc::Serialize(CArchive& ar) {
    if (ar.IsStoring())  {       ar  << m_docStr;// 파일에 저장하기 위한 코드 }
    else            {   ar >> m_docStr; // 파일로부터 읽기 위한 코드 }
}
```

SDI 응용 프로그램 구조

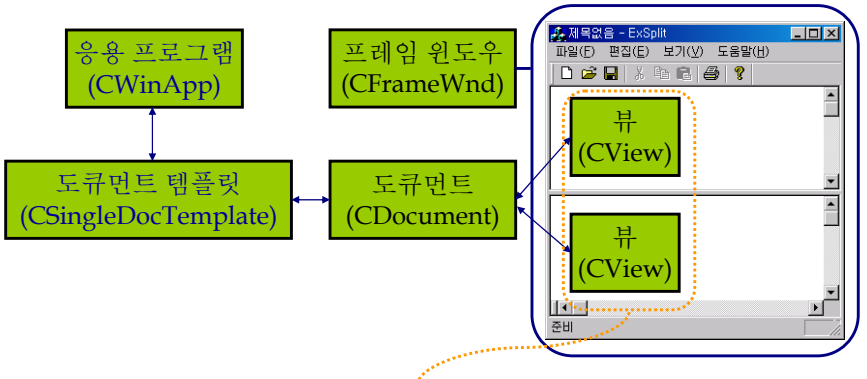
- SDI 응용 프로그램 기본 구조



하나의 도큐먼트만 존재 - 새로운 문서를 생성하거나 저장된 문서를 열 땐 기존의 도큐먼트 객체를 재사용함

SDI 응용 프로그램 구조

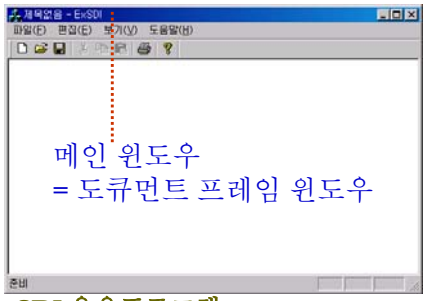
- SDI 응용 프로그램 구조 – 분할 윈도우 사용시



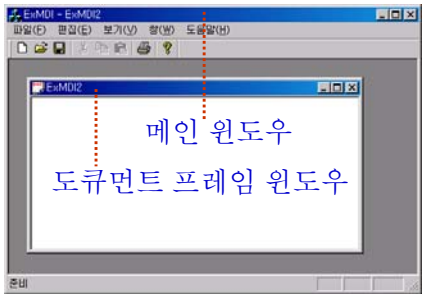
반면, 뷰 객체는 응용프로그램의 필요에 따라 여러 개 생성 가능함

도큐먼트 프레임 윈도우

- 도큐먼트 프레임 윈도우 (Document Frame Window)
  - 도큐먼트의 내용을 화면에 표시하는 역할을 하는 뷰를 자식으로 갖는 윈도우
  - 뷰의 부모 윈도우 또는 뷰를 감싸고 있는 윈도우

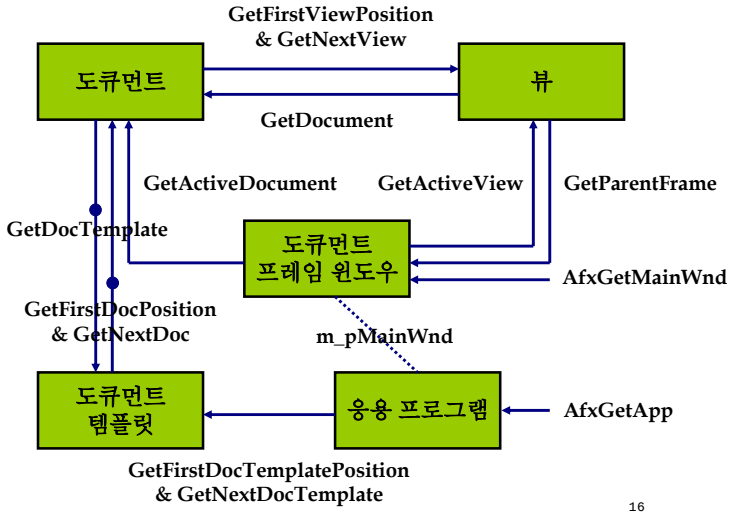


SDI 응용프로그램



MDI 응용프로그램

SDI 응용 프로그램에서 주요 객체 사이의 참조

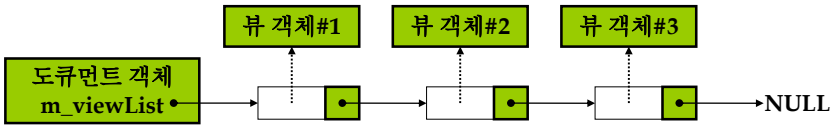


주요 객체 사이의 참조 함수

- CWinApp* **AfxGetApp** ( );
  - 응용 프로그램 객체의 주소를 리턴하는 전역 함수
  - CMyApp *pMyApp = (CMyApp *) AfxGetApp();
- CWnd* **AfxGetMainWnd** ( );
  - 메인 윈도우 객체의 주소를 리턴하는 전역 함수
  - CMainFrame *pMainFrame = (CMainFrame *) AfxGetMainWnd();
- CFrameWnd* CWnd::**GetParentFrame** ( );
  - 부모 윈도우 중 프레임 윈도우 객체의 주소를 리턴
  - 일반적으로 도큐먼트 프레임 윈도우가 이에 해당
- CView* CFrameWnd::**GetActiveView** ( );
  - 활성 뷰 (Active View) 객체의 주소를 리턴

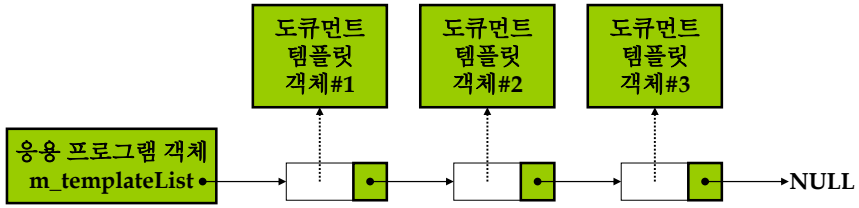
주요 객체 사이의 참조 함수

- CDocument* CFrameWnd::**GetActiveDocument** ( );
  - 활성 도큐먼트 (Active Document) 객체의 주소를 리턴
- CDocument* CView::**GetDocument** ( );
  - 뷰 객체와 연결된 도큐먼트 객체의 주소를 리턴
- POSITION CDocument::**GetFirstViewPosition** ( );
  CView* CDocument::**GetNextView** (POSITION& rPosition);
  - 도큐먼트 객체는 자신과 연결된 뷰를 연결 리스트로 관리
  - 도큐먼트 객체와 연결된 모든 뷰 객체의 주소를 리턴



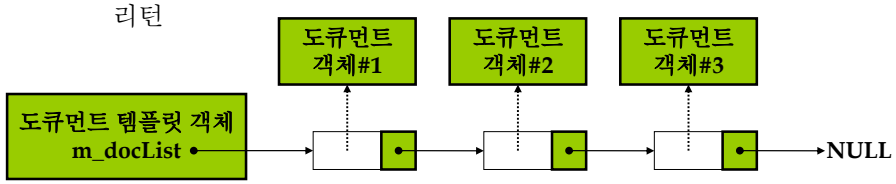
주요 객체 사이의 참조 함수

- POSITION CWinApp::**GetFirstDocTemplatePosition** ( );
  CDocTemplate* CWinApp::**GetNextDocTemplate** (POSITION& pos);
  - 응용 프로그램 객체가 관리하는 모든 도큐먼트 템플릿 객체의 주소를 리턴



주요 객체 사이의 참조 함수

- POSITION CDocTemplate::**GetFirstDocPosition** ( );
  CDocument* CDocTemplate::**GetNextDoc** (POSITION& rPos);
  - MDI 응용 프로그램은 여러 개의 도큐먼트 객체를 생성하며 연결 리스트로 관리
  - 도큐먼트 템플릿 객체가 관리하는 모든 도큐먼트 객체의 주소를 리턴



- CDocTemplate* CDocument::**GetDocTemplate** ( );
  - 도큐먼트 객체와 연결된 도큐먼트 템플릿 객체의 주소를 리턴

SDI 응용 프로그램 예제 작성

- 프로젝트 생성
- 1~6단계 옵션 설정

단계	변경 사항
1	'Single document'를 선택한다.
2	Compound Document Support 변경 사항 없음 Document Template Strings 파일 확장자 sdi를 입력
3	DB 변경 사항 없음
4	UI 변경 사항 없음
5	Advanced Features 'ActiveX Controls' 선택을 해제
6	Generated Classes 변경 사항 없음

SDI 응용 프로그램 예제 작성

- InitInstance() 함수

```
BOOL CExSDIApp::InitInstance()
{
   ...
// 응용 프로그램의 설정 정보가 저장될 레지스트리 위치를 지정
① SetRegistryKey(_T("Local AppWizard-Generated Applications"));
// 가장 최근에 사용한 파일 목록을 레지스트리에서 로드
② LoadStdProfileSettings();

   CSingleDocTemplate* pDocTemplate;
   pDocTemplate = new CSingleDocTemplate(
      IDR_MAINFRAME,
      RUNTIME_CLASS(CExSDIDoc),
      RUNTIME_CLASS(CMainFrame),
      RUNTIME_CLASS(CExSDIView));
   AddDocTemplate(pDocTemplate);
```

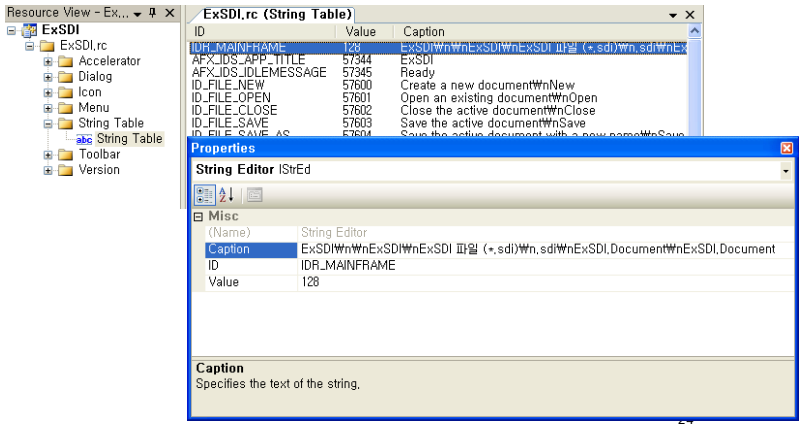

SDI 응용 프로그램 예제 작성

- InitInstance() 함수

```
③ EnableShellOpen(); // 마우스로 더블 클릭해서 파일을 열 수 있게 함
④ RegisterShellFileTypes(TRUE); // 도큐먼트 템플릿을 검사해서 찾아낸
                                // 도큐먼트 타입을 레지스트리에 등록
   CCommandLineInfo cmdInfo;
   ParseCommandLine(cmdInfo);
   if (!ProcessShellCommand(cmdInfo))
      return FALSE;
   m_pMainWnd->ShowWindow(SW_SHOW);
   m_pMainWnd->UpdateWindow();

⑤ m_pMainWnd->DragAcceptFiles(); // 마우스로 파일을 Drag-and-Drop
                                 // 하면 해당 파일을 열도록 함
   return TRUE;
}
```


도큐먼트 문자열 (Document String)

- 도큐먼트 문자열 (Document String)을 위한 리소스
- CDocTemplate::GetDocString()으로 접근



도큐먼트 문자열 (Document String)

ExSDI\n\nExSDI\nExSDI 파일 (*.sdi)\n.sdi\nExSDI.Document
① ② ③ ④ ⑤ ⑥
\nExSDI Document
⑦

번호	의미
1	프레임 윈도우의 타이틀 바에 표시되는 제목
2	새로 생성한 문서의 제목. 생략하면 기본값인 '제목없음'으로 설정
3	두 개 이상의 도큐먼트 타입을 지원하는 MDI 응용 프로그램에서만 사용. 새로운 문서를 생성할 때 도큐먼트 타입을 묻는 대화상자가 뜨는데 이 대화상자에 표시되는 문자열
4	열기 또는 저장하기 대화상자에 표시
5	파일의 기본 확장자로 사용
6	레지스트리에 등록되는 도큐먼트 타입 ID. 공백을 허용하지 않음
7	레지스트리에 등록되는 도큐먼트 타입 문자열. 공백을 허용

도큐먼트 문자열 (Document String)

```
// GetDocString( ) 사용 예
CExSDIApp *pApp = (CExSDIApp *)AfxGetApp();
POSITION pt = pApp->GetFirstDocTemplatePosition();
CDocTemplate* pDocT = pApp->GetNextDocTemplate(pt);

CString title, docname, filenewname,filtername, filterext;
pDocT->GetDocString(title, CDocTemplate::windowTitle);
pDocT->GetDocString(docname, CDocTemplate::docName);
pDocT->GetDocString(filenewname, CDocTemplate::fileNewName);
pDocT->GetDocString(filtername, CDocTemplate::filterName);
pDocT->GetDocString(filterext, CDocTemplate::filterExt);
```


도큐먼트 클래스 주요 함수

- void **SetModifiedFlag** (BOOL bModified = TRUE);
  - 도큐먼트 객체가 유지하는 데이터를 수정한 경우 호출
  - 파일을 저장하지 않고 다른 파일을 열거나 종료 시 확인에 이용
- void **UpdateAllViews** (CView* pSender, LPARAM lHint = 0L, CObject* pHint = NULL);
  - 도큐먼트 객체와 연결된 모든 뷰의 화면을 갱신

CDocument::UpdateAllViews()
⇨ CView::OnUpdate()
⇨ CWnd::Invalidate()
⇨ CWnd::OnPaint()
⇨ CView::OnDraw()

도큐먼트 객체

뷰 객체

  - UpdateAllViews(NULL); // 모든 뷰의 전체 영역 갱신
  - UpdateAllViews(this); // 현재 뷰를 제외한 모든 뷰의 전체영역 갱신
  - UpdateAllViews(NULL, 1, (CObject *) pRect) // 모든 뷰의 사각형 영역만 갱신

도큐먼트 클래스 주요 함수

```
void CDocument::UpdateAllViews(CView* pSender, LPARAM lHint,
CObject* pHint)
{
        // 생략
        POSITION pos = GetFirstViewPosition();
        while (pos != NULL)
        {
           CView* pView = GetNextView(pos);
           if (pView != pSender) // pSender는 제외
                pView->OnUpdate(pSender, lHint, pHint);
           // Invalidate() 호출 WM_PAINT -> OnPaint() -> OnDraw()
        }
}
```

도큐먼트 클래스 주요 가상 함수

- virtual BOOL **OnNewDocument** ( );
  - 새 문서를 생성할 때 자동으로 호출
  - 도큐먼트 클래스의 생성자보다 OnNewDocument 함수에 초기화 코드를 추가하는 것이 바람직함
- virtual BOOL **OnOpenDocument** (LPCTSTR lpszPathName);
  - 파일을 열 때 자동으로 호출, 읽어 들인 데이터를 처리할 때 재정의
- virtual BOOL **OnSaveDocument** (LPCTSTR lpszPathName);
  - 파일을 저장할 때 자동으로 호출, 저장할 데이터를 처리할 때 재정의
- virtual void **DeleteContents** ( );
  - 새로운 문서를 생성하거나 파일을 열 때 자동으로 호출
- virtual void **Serialize** (CArchive& ar);
  - 파일을 열거나 저장할 때 자동으로 호출

도큐먼트 클래스 가상 함수 호출 순서

- 가상 함수 호출 순서
  - [파일]->[새 파일] 메뉴 항목을 선택할 때

OnNewDocument() ⇨ DeleteContents()

  - [파일]->[열기...] 메뉴 항목을 선택할 때

OnOpenDocument() ⇨ DeleteContents() ⇨ Serialize()

  - [파일]->[저장] 또는 [파일]->[다른 이름으로 저장...] 메뉴 항목을 선택할 때

OnSaveDocument() ⇨ Serialize()

뷰 클래스 주요 가상 함수

- virtual void **OnDraw** (CDC* pDC);
  - 화면 출력, 인쇄, 인쇄 미리 보기를 할 때 자동으로 호출
- virtual void **OnInitialUpdate**();
  - 뷰 객체가 도큐먼트 객체와 연결된 후 화면에 보이기 전에 자동으로 호출
  - CView::OnInitialUpdate() 함수는 CView::OnUpdate() 함수를 호출하여 뷰의 화면 전체를 무효화
- virtual void **OnUpdate** (CView* pSender, LPARAM lHint, CObject* pHint);
  - CDocument::UpdateAllViews() 함수와 CView::OnInitialUpdate() 함수에서 호출
  - CWnd::Invalidate()를 이용하여 뷰 화면 전체를 무효화
  - 효과적인 화면 갱신이 필요할 때 재정의
  - 두 번째와 세 번째 인자를 참조하여 뷰의 화면 일부만 무효화

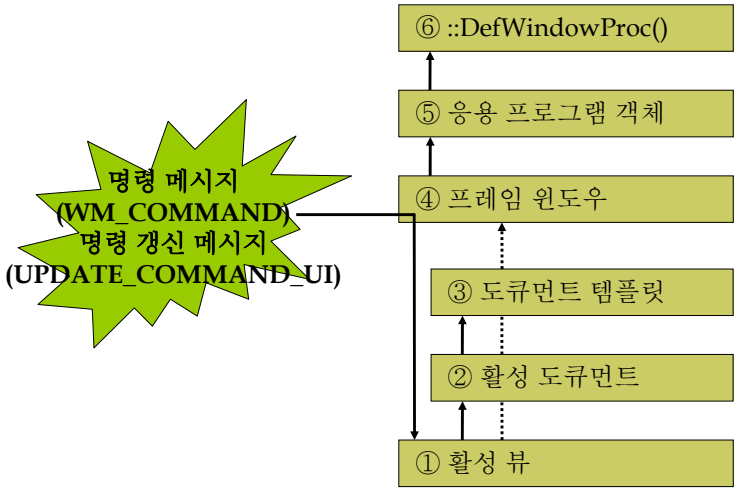
명령 라우팅 (Command Routing)

- MFC 응용 프로그램에서 사용하는 메시지 종류

종류	설명	메시지맵 매크로
윈도우 메시지	윈도우 생성, 종료, 마우스, 키보드 등 다양한 원인에 의해 발생	ON_WM_XXX
명령 메시지	메뉴, 툴바, 가속기 등에 의해 발생	ON_COMMAND(ID, 함수)
명령 갱신 메시지	메뉴, 툴바, 상태바 등의 상태를 갱신할 필요가 있을 때 발생. MFC에서만 사용하는 고유의 메시지	ON_UPDATE_COMMAND_UI(ID, 함수)
통지 메시지	컨트롤 (자식 윈도우)이 부모 윈도우에게 보내는 메시지	ON_XXX (ID, 함수)

명령 라우팅 (Command Routing)

- 명령 라우팅 순서



분할 윈도우 (Splitter Window)

- 동적 분할 윈도우 (Dynamic Splitter Window)
  - 같은 뷰 클래스를 기반으로 여러 개의 뷰를 생성
  - 총 네 개의 구획(Pane) 생성 가능

```
CBOOL CMainFrame::OnCreateClient(LPCREATESTRUCT /*lpcs*/,
    CCreateContext* pContext)
{
    return m_wndSplitter.Create(this,
        2, 2,               // TODO: adjust the number of rows, co
        CSize(10, 10),      // TODO: adjust the minimum pane size
        pContext);
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    // TODO: Modify the Window class or styles here by modifying
    //  the CREATESTRUCT cs

    return TRUE;
}
```

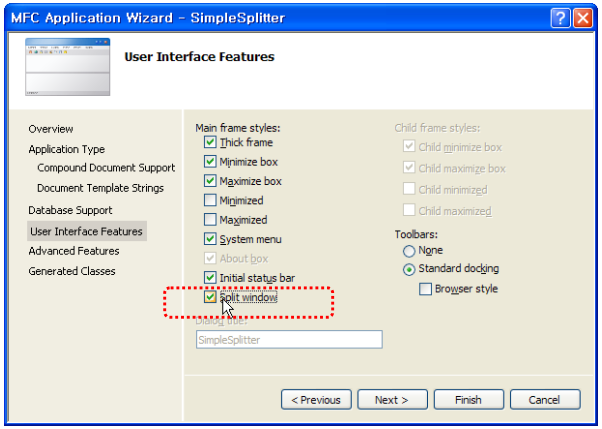

동적 분할 윈도우 구현 예제 작성

- 프로젝트 생성
- 1~6단계 옵션 설정

단계	변경 사항
1	'Single document'를 선택한다.
2	Compound Document Support 변경 사항 없음 Document Template Strings 파일 확장자를 입력
3	DB 변경 사항 없음
4	UI에서 Split Window를 체크
5	Advanced Features 'ActiveX Controls' 선택을 해제
6	Generated Classes 변경 사항 없음

동적 분할 윈도우 구현 예제 작성

- 동적 분할 윈도우 구현



```
//동적 분할 윈도우 지원을 위해 생성된 코드
class CMainFrame : public CFrameWnd {
…
// Attributes
protected:
  CSplitterWnd m_wndSplitter;  //동적 분할 윈도우 지원을 위한 클래스 추가
}
//동적 분할 윈도우 지원을 위해 생성된 코드
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT /*lpcs*/,
        CCreateContext* pContext) {
  return m_wndSplitter.Create(
        this,           //부모 윈도우
        2, 2,           //가로, 세로 분할 개수
        CSize(10, 10), //구획(pane)의 기본 크기
                        //이 크기보다 작으면 분할을 없앰
        pContext);      //MFC 윈도우생성에 필요한 구조체
}
```



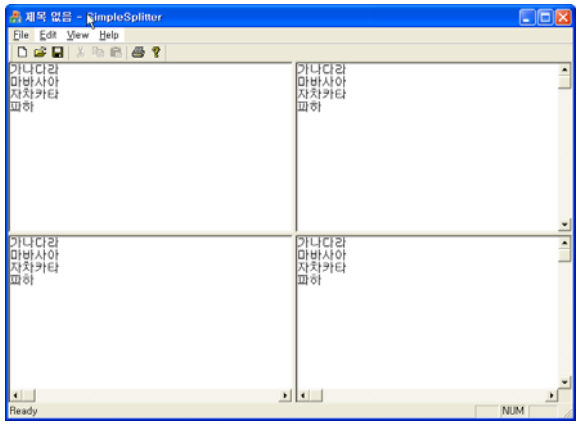
```
// Document 파일에 m_str을 분할된 윈도우에 출력하는 프로그램
// (WM_CHAR 메시지 처리에 의하여 입력된 문자를 m_str에 추가)
//(1)Document class
class CSplitterWinDoc : public CDocument
{
…
public:
        CString m_str; //변수 선언
}

BOOL CSplitterWinDoc::OnNewDocument()
{
        if (!CDocument::OnNewDocument()) return FALSE;
        m_str = "";  //변수 초기화
        return TRUE;
}
```



```
//(2)View Class
void CSplitterWinView::OnDraw(CDC* pDC)
{
        CSplitterWinDoc* pDoc = GetDocument();
        CRect rect;
        GetClientRect(&rect);
        // Document 객체의 m_str 출력
        pDC->DrawText(pDoc->m_str, &rect, DT_LEFT);
}

void CSplitterWinView::OnChar(UINT nChar, UINT nRepCnt, UINT
nFlags)
{
        CSplitterWinDoc* pDoc = GetDocument();
        pDoc->m_str += (char) nChar; //입력된 문자를 m_str에 추가
        // Document와 연결된 모든 View에 대한 화면 갱신
        pDoc->UpdateAllViews(NULL);
        CView::OnChar(nChar, nRepCnt, nFlags);
}
```

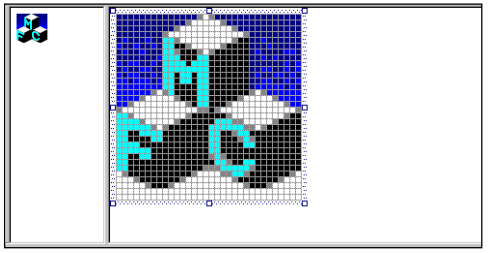

동적 분할 윈도우 구현 예제 작성

- 실행결과



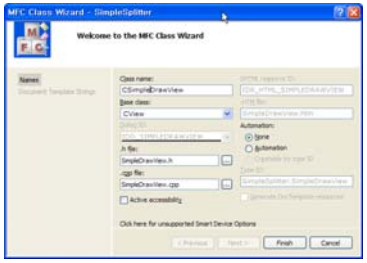
분할 윈도우

- 정적 분할 윈도우 (Static Splitter Window)
  - 서로 다른 뷰 클래스를 기반으로 여러 개의 뷰를 생성
  - 총 256개의 구획 생성 가능



분할 윈도우

- 정적 분할 윈도우 구현
  - CView 를 기반으로 하는 새로운 뷰 클래스를 생성



  - 두 개의 뷰 클래스에 대하여 서로 다른 처리 내용 구현

```
// 두 번째 분할 영역을 처리하는 뷰 클래스에 WM_LBUTTONDOWN 메시지 처리
// 추가 (사각형 출력)
void CSimpleDrawView::OnLButtonDown(UINT nFlags, CPoint point) {
  CClientDC dc(this);
  dc.SelectStockObject(LTGRAY_BRUSH);
  dc.Rectangle(point.x-20, point.y-20, point.x+20, point.y+20);
  CView::OnLButtonDown(nFlags, point);
}
```


분할 윈도우

- 정적 분할 윈도우 구현
  - 프레임 윈도우 클래스의 OnCreateClient () 함수 수정

```
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT /*lpcs*/,
    CCreateContext* pContext) {
  m_wndSplitter.CreateStatic(this, 2, 1); // 가로=2, 세로=1 분할
  m_wndSplitter.CreateView(0, 0,          // row=0, col=0 구획
                           RUNTIME_CLASS(CSimpleSplitterView),
                           CSize(300, 200), pContext);
  m_wndSplitter.CreateView(1, 0,          // row=1, col=0 구획
                           RUNTIME_CLASS(CSimpleDrawView),
                           CSize(300, 200), pContext);
  // row=0, col=0에 위치하는 뷰 객체의 주소를 받아서 활성뷰로 설정
  SetActiveView((CView *)m_wndSplitter.GetPane(0, 0));
  return TRUE;
}
```

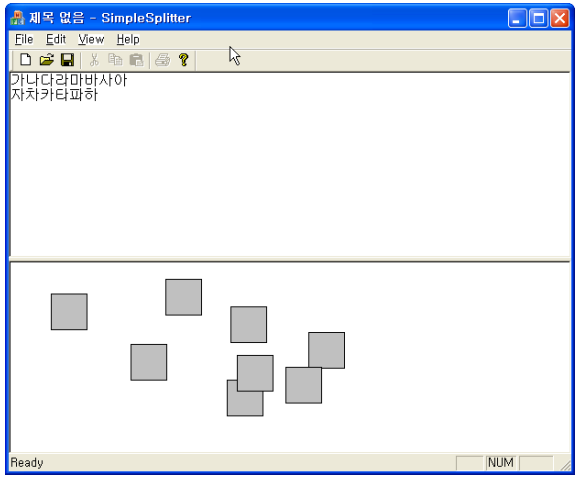

분할 윈도우

- 정적 분할 윈도우 구현
  - 프레임 윈도우 클래스 (MainFrm.cpp)에 헤더 파일 추가
    - #include "SimpleDrawView.h"
    - #include "SimpleSplitterView.h"
    - #include "SimpleSplitterDoc.h"
  - 뷰 클래스의 OnChar에 Invalidate를 호출

```
BOOL CSimpleSplitterView::OnChar(UINT nChar, UINT nRepCnt, UINT
    nFlags) {
  CSimpleSplitterDoc* pDoc = GetDocument();
  pDoc->m_str += (char) nChar; // 입력된 문자를 doc의 m_str에 추가

  Invalidate(); // 이 뷰만 다시 그림
  return TRUE;
}
```

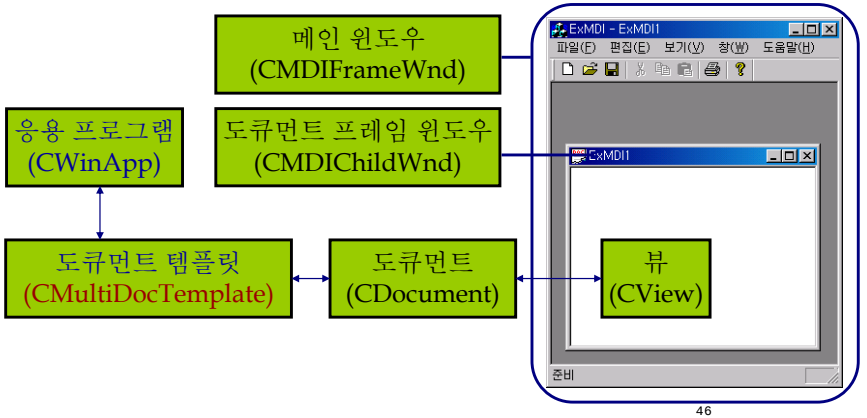
정적 분할 윈도우 구현 예제 작성

- 실행결과



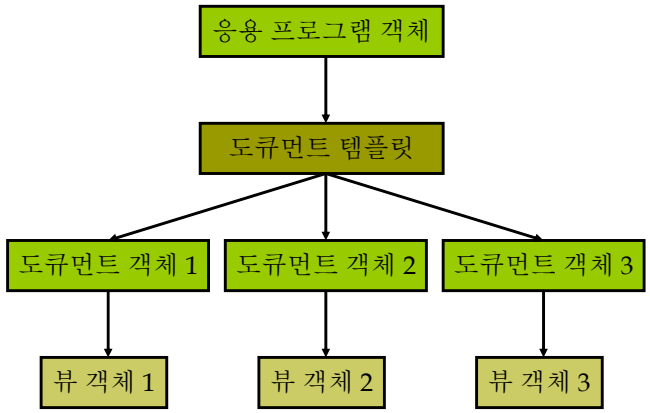
MDI 응용 프로그램 구조

- MDI 응용 프로그램 기본 구조



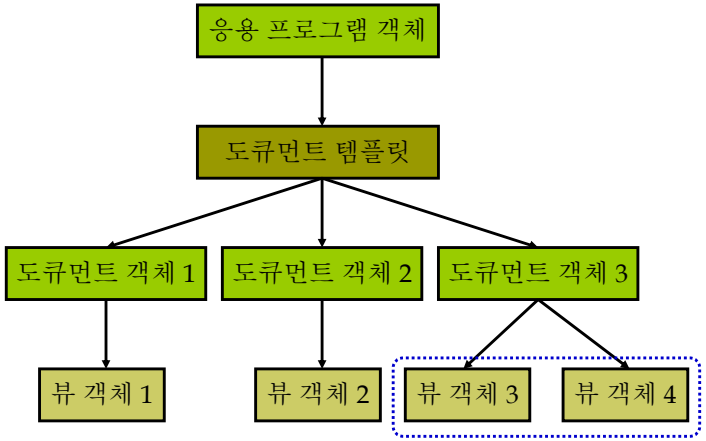
MDI 응용 프로그램 구조

- MDI 응용 프로그램 일반 구조



MDI 응용 프로그램 구조

- MDI 응용 프로그램 일반 구조



SDI와 MDI 응용 프로그램 비교

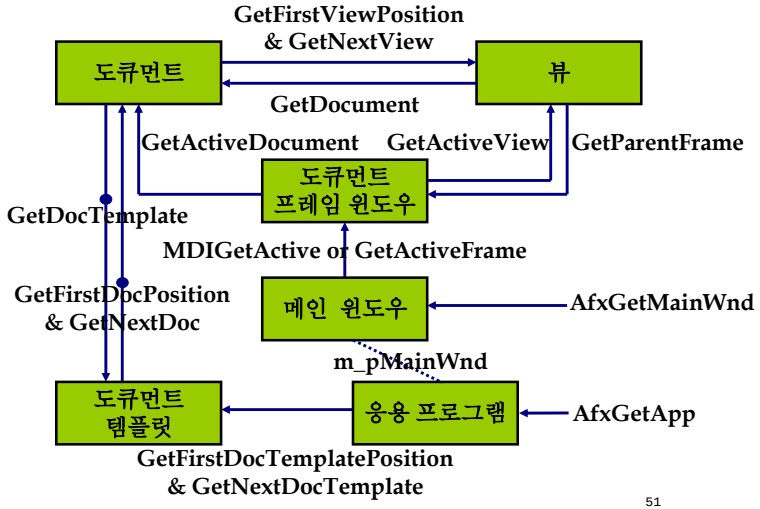
- 도큐먼트 템플릿으로 CSingleDocTemplate 클래스 대신 **CMultiDocTemplate** 클래스 사용
- SDI 응용 프로그램과 달리 MDI 응용 프로그램은 도큐먼트 객체를 재사용하지 않고 매번 **새로 생성**
- MDI 응용 프로그램에서는 메인 윈도우와 도큐먼트 프레임 윈도우가 별개이며 각각 **CMDIFrameWnd, CMDIChildWnd** 클래스 사용

MDI 응용 프로그램의 InitInstance()

- InitInstance() 함수

```
BOOL CSimpleMDIApp::InitInstance()
{
   ...
   CMultiDocTemplate* pDocTemplate;
   pDocTemplate = new CMultiDocTemplate(
      IDR_SimpleMDITYPE,
      RUNTIME_CLASS(CSimpleMDIDoc),
      RUNTIME_CLASS(CChildFrame), // 도큐먼트 프레임 윈도우
      RUNTIME_CLASS(CSimpleMDIView));
   AddDocTemplate(pDocTemplate);

   CMainFrame* pMainFrame = new CMainFrame; // 메인 윈도우
   if (!pMainFrame->LoadFrame(IDR_MAINFRAME))
      return FALSE;
   m_pMainWnd = pMainFrame;
   ...
}
```


MDI 응용 프로그램에서 주요 객체 사이의 참조



주요 객체 사이의 참조 함수

- CMDIChildWnd* CMDIFrameWnd::**MDIGetActive** (BOOL* pbMaximized = NULL);
  CFrameWnd* CFrameWnd::**GetActiveFrame** ( );
  - 활성 도큐먼트 프레임 윈도우 객체의 주소를 리턴

MDI 응용 프로그램 특징

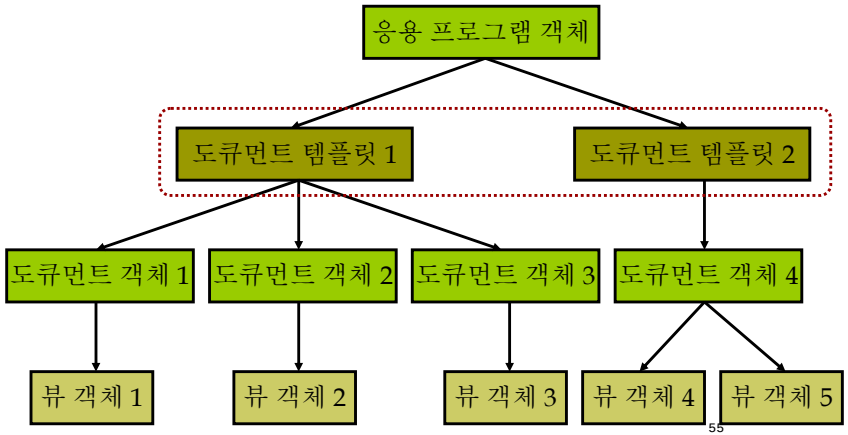
- [파일]->[새 파일] 메뉴 항목을 선택하면 새로운 도큐먼트, 도큐먼트 프레임 윈도우, 뷰 객체가 생성
- [창]->[다음 창] 메뉴 항목을 선택하면 활성 도큐먼트에 대한 뷰를 추가로 생성 가능
- 최소 두 개의 메뉴 사용

다양한 도큐먼트 타입 지원

- 새로운 도큐먼트 타입 추가
  - ① 새로운 도큐먼트와 뷰 클래스를 생성하고 구현 코드 작성
  - ② 아이콘, 메뉴, 도큐먼트 문자열 등 리소스를 추가
  - ③ InitInstance() 함수 수정

다양한 도큐먼트 타입 지원

- MDI 응용 프로그램 일반 구조
  - 두 개 이상의 도큐먼트 타입 지원



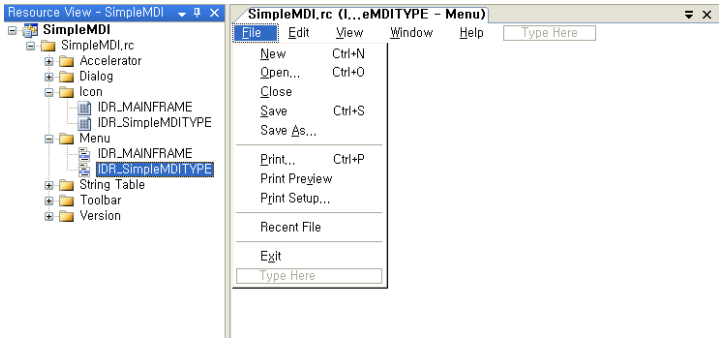
MDI 응용 프로그램 예제 작성

- 프로젝트 생성
- 1~6단계 옵션 설정

단계	변경 사항
1	'Multiple document'를 선택한다.
2	Compound Document Support 변경 사항 없음
3	DB 변경 사항 없음
4	UI 변경 사항 없음
5	Advanced Features 'ActiveX Controls' 선택을 해제함
6	뷰 클래스의 Base Class를 CEditView로 변경

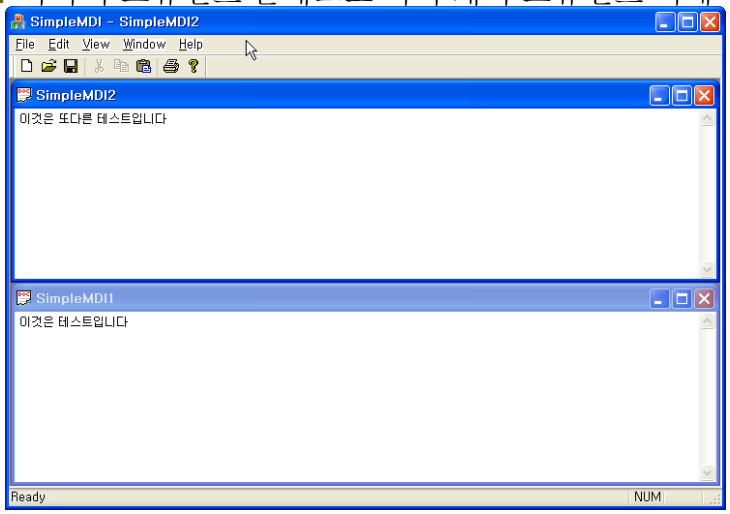
MDI 응용 프로그램 예제 작성

- 메뉴 리소스



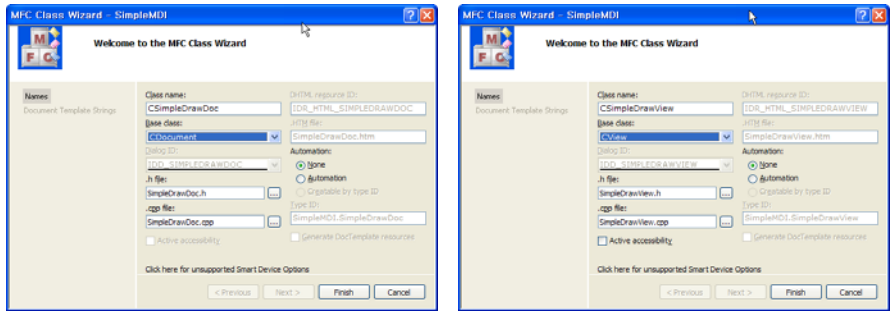
MDI 응용 프로그램 예제 작성

- 하나의 도큐먼트 클래스로 여러 개의 도큐먼트 객체 생성 예



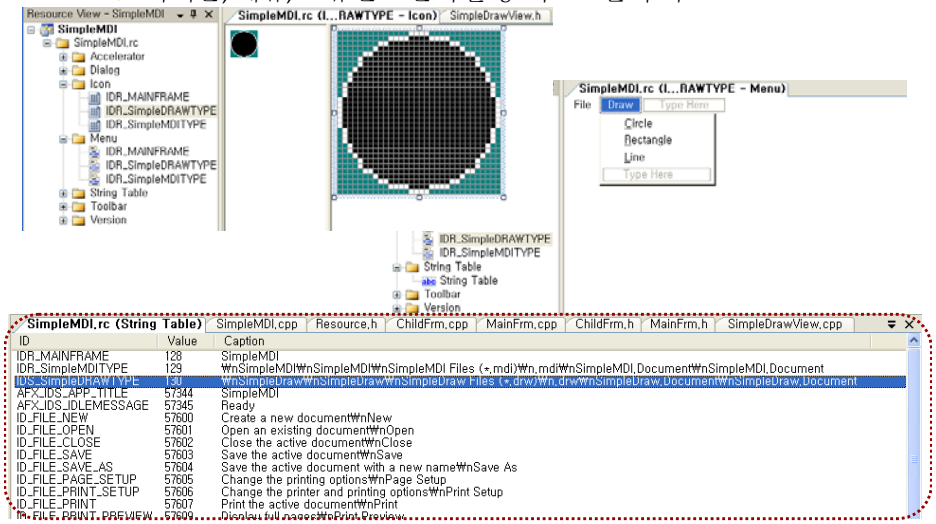
MDI 응용 프로그램 예제 작성

- 여러 개의 도큐먼트 클래스를 이용한 예를 추가
  1. 새로운 도큐먼트와 뷰 클래스를 생성하고 구현 코드 작성



MDI 응용 프로그램 예제 작성

2. 아이콘, 메뉴, 도큐먼트 문자열 등 리소스를 추가



MDI 응용 프로그램 예제 작성

3. 응용 프로그램 클래스의 InitInstance() 함수 수정

```
BOOL CSimpleMDIApp::InitInstance() { // 생략…
  CMultiDocTemplate* pDocTemplate;
  pDocTemplate = new CMultiDocTemplate(
        IDR_SimpleMDITYPE,
        RUNTIME_CLASS(CSimpleMDIDoc),
        RUNTIME_CLASS(CChildFrame), // custom MDI child frame
        RUNTIME_CLASS(CSimpleMDIView));
  AddDocTemplate(pDocTemplate);
  //추가된 토큐먼트, 뷰를 위한 탬플릿 생성 및 응용프로그램 객체에 추가
  pDocTemplate = new CMultiDocTemplate(
        IDR_SimpleDRAWTYPE,
        RUNTIME_CLASS(CSimpleDrawDoc),
        RUNTIME_CLASS(CChildFrame), // custom MDI child frame
        RUNTIME_CLASS(CSimpleDrawView));
  AddDocTemplate(pDocTemplate);
// 생략…
}
```


MDI 응용 프로그램 예제 작성

4. 응용 프로그램 클래스에 헤더 파일 추가

```
// SimpleMDI.cpp

#include "stdafx.h"
#include "SimpleMDI.h"
#include "MainFrm.h"

#include "ChildFrm.h"
#include “SimpleMDIDoc.h"
#include “SimpleMDIView.h"

#include “SimpleDrawDoc.h"
#include “SimpleDrawView.h"
```


MDI 응용 프로그램 예제 작성

5. 응용 프로그램 실행 시 존재하는 도큐먼트 타입을 선택하여 새로 만들기