



Nghiên cứu Visual C++ trên môi trường Window



Tổng Quan:

Sở khác biệt giữa chương trình được viết trên MS_DOS và chương trình viết trên Window là trên MS_DOS chương trình lấy vào Input thì gọi những hoạt động của hệ thống. Còn trên Window thì chương trình này xử lý thông qua Message Input của hệ thống tổng cộng với Input của user. Hầu hết các Message trong Window đều được định nghĩa trong chương trình khi được tạo. Nhờ WM_CREATE message được gọi thì window được tạo hay Window gọi WM_COMMAND message nên window tổng cộng nạp lại việc chọn lựa menu, dialog button ... Những công việc này đều do Application framework làm. Nó sẽ kết nối các message này với Code của chương trình. Trong quá trình xây dựng tại các module liên kết những việc này được tích hợp trong quá trình xây dựng vào thời điểm runtime thì việc cài đặt được Load và Link. Các ứng dụng có thể chia sẻ DLL này (Data Link Library). Liên kết những việc làm tăng số Module của chương trình bởi vì nó dịch và kiểm tra DDL một cách riêng biệt. Đồng thời Window giới thiệu một lớp GDI (Graphic Device Interface). Chương trình được viết ra tiếp cận driver Video và Printer của hệ thống một cách dễ dàng. Chương trình có thể gọi các hàm chức năng của GDI, tham khảo cấu trúc của dữ liệu bằng gọi Device Context. Window sẽ ánh xạ cấu trúc Device Context thành Physical device và đưa ra Input/ Output tổng cộng.

Microsoft Developer Studio và tiến trình xây dựng :

Visual C++ là một thành phần của Microsoft Developer Studio là IDE (Integrated development environment). IDE có nguồn gốc từ Visual Workbench dựa trên QuickC cho Window. Microsoft Developer Studio cung cấp khả năng:

Help Online làm việc nhờ một Web browser.

App Wizard

Có thể xem Project ở nhiều khía cạnh (editor, workspace)

Class Wizard

Xây dựng giao diện gồm trình nhúng và các khung đối thoại

Compiler , Linking

Gỡ lỗi Debug



Ngành công Visual C++ trên môi trường Window

Thủ việc MFC và lớp VIEW

Khởi tạo ứng dụng

Các bước tạo ứng dụng:

AppWizard sẽ giúp tạo một ứng dụng nhàn hình , ứng dụng này có môi trường làm .EXE nhàn hoặc thì . Nhàn tab môi menu FILE\ NEW thì sẽ xuất hiện một bảng liệt kê, chọn Tab Project với những loại dự án có thể tạo nhàn vào loại MFC AppWizard (.exe) . Nhàn vào tên Project và nhàn nút < OK> tiến hành tạo dự án mới .

Bước 1:

Chọn Multi Document nhàn nút < Next > nhàn làm tiếp hoặc < Back > nếu muốn trôi lại bước và làm hoặc chọn < Finish > nhàn chấp nhàn tất cả những mặc nhàn chông trình nguồn do AppWizard tạo ra.

Bước 2:

Bước này sẽ chọn các dữ liệu hỗ trợ cho ứng dụng .

Bước này nên chọn None nhàn nút < Next > qua bước kế

Bước 3:

Quyết nhàn xem một nhàn hỗ trợ của OLE (ActiveX)

Chọn < None > nhàn nút < Next >

Bước 4:

Nhàn là giai nhàn tạo giao diện cho ứng dụng

- Docking toolbar: Cung cấp một thanh công cụ gồm những nút biểu tượng New, Open, Save, Cut, Paste, About Help
- Initial status bar: cung cấp thanh tình trạng hiển thị những câu nhàn nhàn
- Printing và PrintPreview: Kết sinh nhàn mẫu việc in nhàn
- Context _Sentitive Help: hỗ trợ việc chọn Index và sử dụng Help trên menu Help những nhàn hỏi phải có trình biên dịch Help Compiler.
- 3D control : giao diện của chông trình sẽ có các ô nhàn khiến nhàn theo ba chiều
- MAPI (Message API) : cung cấp khả năng sử dụng message API nhàn gửi fax, e_mail...

Nhấn <Next> để qua bước 5.
Nhấn <Next> để qua bước 5.

Bước 5:

Chương trình hỏi có cần hỗ trợ chuỗi thích trong chương trình hay không ?

AppWizard sẽ hỗ trợ chuỗi thích trong chương trình nên hãy nhấn nút **Yes** để tiếp tục.

- Hỏi cần sử dụng thư viện MFC theo cách nào

As a shared DLL : khi thực thi sẽ tải yêu cầu hàm từ thư viện

As a statically linked library: Chương trình sẽ kết nối với thư viện vào lúc thiết kế. Cách này làm tốn bộ nhớ.

Chọn < next > sau khi trả lời những câu hỏi trên. Thông chọn As a shared DLL

Bước 6:

- AppWizard thông báo sẽ kết sinh những lớp mới tổng hợp các lớp liệt kê trong khung liệt kê New Classes. Chọn **Yes** để tiếp tục.

- Class name: Tên lớp mới chọn trong khung liệt kê Chọn CView

- Base Class: Cho biết lớp cơ sở của lớp mới kết sinh nên chọn **Yes** để tiếp tục.

- Header file: Tên tập tin tiêu đề của lớp mới chọn (.h) Implementation file: tên tập tin nguồn (.cpp) nối với lớp mới chọn

Chọn < Finish > để kết thúc .

AppWizard sẽ cho hiện lên khung hỏi thoại New project Information cho biết những thông tin liên quan đến ứng dụng sẽ tạo. Nếu không ý thì chọn < OK >. Vả sẽ tiếp tục tạo ứng dụng với những lựa chọn trên.

Chạy ứng dụng:

Click vào Menu Build \ Build LuuDo.exe

Thực thi : Build\ Execute LuuDo.exe

Một vài yêu cầu của ứng dụng:

Ứng dụng sẽ tạo ra với MDI (Multi Document Interface) gồm một số lớp mới tổng hợp. Những lớp này thuộc các lớp cơ sở Application, Document, View, Frame . Những lớp mới của ứng dụng:

MFC Base Class	Lớp mới dẫn xuất	Tên tập tin
----------------	------------------	-------------

CwinApp	CBaiTap1App	BaiTap1.cpp
Cdocument	CBaiTap1Doc	BaiTap1Doc.cpp
Cview	CBaiTap1View	BaiTap1View.cpp
CframeWnd	CmainFrame	MainFrm.cpp

Những lớp nội nội tổng mà dẫn xuất từ CDocument dùng để cài đặt giao diện của ứng dụng. Chịu trách nhiệm đọc và viết các tập tin.

Lớp CView cho phép user xem và thao tác trên các đối tượng.

Lớp nội nội dẫn xuất từ CMainFrame thiết lập khung cửa sổ chính cho ứng dụng. Lớp này chịu trách nhiệm quản lý trình nhúng, thanh công cụ, thanh tình trạng nội nội cửa sổ của ứng dụng.

CWinApp nội nội dẫn xuất theo sơ đồ



Lớp này lo việc nội ngoại với Windows, một tình huống xảy ra thì thông điệp sẽ nội nội gửi đến cho chương trình nội nội biết cái gì đang xảy ra.

Ví dụ: Một phím nội nội ấn xuống thì thông điệp WM_KEYDOWN sẽ nội nội phát ra và chuyển cho CLuDoApp và chương trình sẽ khởi nội nội hàm giải quyết và nội nội gọi OnKeyDown().

View là một cửa sổ Windows thông thường có thể thay nội nội kích thước, nội nội nội nội. Nội nội nội nội dẫn xuất từ lớp CView trong thư viện MFC. Cũng như các nội nội tổng khác thì nội nội tổng View cũng nội nội xác định bởi hàm thành viên (Member Function) hoặc là biến thành viên (Data Member) của nội nội tổng thời có các hàm nội nội biết, hàm chuẩn nội nội thỏa nội nội tổng nội nội cần biết mà khi chọn trong giai đoạn khởi tạo ứng dụng (trong bước 5). Lớp View nội nội chia làm hai module nguồn: header file (.h) và file bổ sung (.cpp)

Thư viện MFC hỗ trợ 2 loại ứng dụng: SDI (Single Document Interface) và MDI (Multi Document Interface)

Với loại SDI thì ta chỉ có một cửa sổ window. Dữ liệu dạng tập tin (file Document) thì chỉ có 1 tập tin nội nội load lên. MDI thì có nhiều cửa sổ con, mỗi một View này nội nội tải tập tin.

Lớp LuDoView có nội nội cần biết là CWnd tạo khung hình chôn nội ngoại ra con là nội nội ngôi ngôi của framework trong ứng dụng.

Một View có thể nội nội gắn với một document mà chịu trách nhiệm giải quyết nội nội kiểu nhập liệu khác nhau nhờ tổ hợp phím, bấm tại con chuột, lối thả con chuột, cũng nhờ hiển thị đối tượng của Document. Nội nội nội nội nhờ là một trung gian giữa Document và nội nội sử dụng, phản ánh hình ảnh của Document lên màn hình hay lên máy in nhờ hàm thành viên OnDraw(). Hàm thành viên này tìm thấy trong BaiTap1View.cpp. Hàm nội nội gọi thông

xuống bởi Framework của ứng dụng để làm công việc vẽ trên màn hình. Gọi OnDraw xong những Windows không cho tiếp cận trực tiếp hardware mà phải gọi một "Device Context" để nói với Window bằng dùng con trỏ HDC chứ tôi nói nhớ một tham số của hàm này. Lúc này coi thể bất kỳ hàm thành viên nào của lớp CDC này hoạt động cho việc vẽ.

Gọi bằng menu Build\ Configuration

Có hai chế độ dịch: Debug Target và Release Target.

Bảng tóm tắt:

	Release Build	Debug Build
Debug mã nguồn	Không thể	Cần cho compiler và linker
MFC nở ra Macro	Không thể	Có thể
Kết nối thư viện	MFC phong thích thư viện	MFC debug thư viện
Tốc độ dịch	Nhanh	Không nhanh

Mapping Mode:

Window cung cấp một số chế độ mapping mode hay còn nói gọi là tọa độ hệ thống, nó gắn liền với thiết bị ngoại vi (Device Context).

MM_TEXT:

Tọa độ đơn vị tính là Pixel, giá trị y tăng dần từ trên xuống và tọa độ x từ trái qua phải. Cho phép thay đổi vị trí origin bằng cách gọi hàm chức năng của lớp CDC nhờ SetViewportOrg và SetWindowOrg. Một số mapping mode:

những chế độ này nếu coi giá trị x tăng hướng sang phải và y giảm hướng xuống

MM_LOENGLISH 0.01 inch MM_HIMETRIC 0.01 mm

MM_HIENGLISH 0.001 inch MM_LOMETRIC 0.1 mm

NỘI TỌA NỘI:

Một khi đã xác lập chế độ ảnh xạ (mapping mode) cho device context rồi và coi thể dùng tham số tọa độ logic cho hầu hết tất cả hàm thành viên của lớp CDC. Nếu nhận được tọa độ tính bằng chuột từ Message Window Mouse (tham số Point trong OnLButtonDown) thì lúc này nâng xối lý với tọa độ thiết bị. Do đó cần phải có tọa độ từ Physic sang Logic hay ngược lại để thích hợp hơn trong quá trình viết chương trình. GDI quản lý việc chuyển đổi này. Nếu có giữa hai hệ thống thì CDC cung cấp hàm LPtoDP và DPtoLP.

Những hàm thành viên của CDC nếu nhận và trả về thông số có tọa độ Logic.

Những hàm thành viên của CDC nếu giao thông số có tọa độ Device.

Các hit-test nếu làm việc trên tọa độ Device nhờ CRect::PtInRect làm việc với tọa độ Device.

CScroll View:

Lớp CScrollView: là con của lớp CView là một lớp View chịu trách nhiệm cuộn trang. Lớp CScrollView cung cấp khả năng cuộn trở thành cuộn không phải từ bàn phím

Chức năng : Quản lý kích thước của cửa sổ và viewport, chế độ mapping. Tôi không cuộn trang nạp lại thông điệp của thanh điều hành, thông điệp bàn phím và con chuột.

Những muốn chúng làm việc thì phải dẫn xuất lớp View của ứng dụng (trong quá trình AppWizard xây dựng ứng dụng (bước 6)) từ lớp CScrollView thay cho CView (mặc định).

Thông thường View port không có thể ở ngoài biên Left Top của Window do đó CScrollView cho phép dịch chuyển viewport nằm trong Window bằng việc sử dụng hàm ScrollWindow và SetWindowOrg của lớp Cwnd. Microsoft Window đang trình bày scroll Bar ở bên của Window. Nhiệm vụ của CScrollView xử lý thông điệp WM_HSCROLL và WM_VSCROLL do Scroll bar tạo ra cho View.

Muốn tôi xác định kích thước và chế độ Mode của View thì dùng hàm SetScrollSizes () hoặc gọi trong CView::OnInitialUpdate () hay CView::OnUpdate ().

- OnInitialUpdate Function là một hàm thành viên quan trọng bởi vì nó là hàm đầu tiên được gọi bởi Framework sau khi đã khởi tạo View. Framework gọi nó trước khi gọi OnDraw. Đây là nơi thiết lập Logical size và Mapping mode cho Scrolling view

Một số hàm thành viên của lớp này

- GetDeviceScrollPosition () const _Trả về vị trí hiện hành của Scroll boxes theo chiều ngang và chiều dọc tính theo đơn vị thiết bị.
- GetScrollPosition () nhớ hàm trên nhưng tính theo đơn vị Logic

Window Message

Khung làm việc của ứng dụng không sử dụng hàm chức năng Virtual cho Windows Message. Thay vào đó sử dụng macros ánh xạ message hàm chức năng của lớp được dẫn xuất. Nếu MFC dùng hàm virtual cho message thì ứng dụng cần nên bằng 440 byte cho việc xử lý những message Virtual. Do đó xử lý message MFC nói tới có prototype, thay hàm chương trình và phải xem nhập trong việc ánh xạ message.

Mọi hoạt động trên Framework đều thông qua thông điệp tổng ứng với mã nhân diện ID ta sẽ các lệnh các thông điệp window. Ví dụ như người sử dụng di chuyển con chuột thì cửa sổ này sẽ nhận được thông điệp WM_MouseMove. Chương trình cần phải nạp ứng bằng một hàm chức năng tôi tạo. Do đó lớp View sẽ có một hàm chức năng tổng ứng

```
void CBaiTap1View :: OnMouseMove( UINT nFags, CPoint point)
```

```
{ //Năng thời trong file .h phải có prototype tổng ứng
```

```
Afx_msg void OnMouseMove ( UINT nFags, CPoint point)
```

Và một macro thông điệp ánh xạ trong file .cpp để tiếp cận với khung làm việc của ứng dụng

```
BEGIN_MESSAGE_MAP (CView, CView)
ON_WM_MOUSEMOVE ()
END_MESSAGE_MAP()
```

Macro `afx_msg` là khai báo hàm thành viên nạp ống thông điệp. Nếu ta bỏ hàm thông điệp này ta sẽ dùng ClassWizard chọn lớp cần biến đổi hàm nào, sau đó chọn Window Message tổng ống sau khi nào có sự kiện (click < OK >) thì nó sẽ hiển thị ảnh xài vào trong chương trình ta có thể vào nó để phát triển hàm xử lý tổng ống với thông điệp nào

- Các thông điệp lệnh mang những hàm giải quyết mặc nhiên trong MFC nhờ với Menu File : New, Open, Close, Save ..

Edit: Clear, Clear All, Cut, File ..

View: Toolbar, Statusbar..

- Thông điệp Window (bắt đầu bằng WM_) giải quyết các hướng thông điệp. Ngoài các thông điệp mặc nhiên, ta có thể định nghĩa chúng bằng cách vào WM_USER.

GDI (Graphic Device Interface) , CDC (lớp Device Context)

Lớp Device Context: (lớp thiết bị đồ họa)

CDC là lớp cần biến cung cấp các hàm thành viên (và một số hàm Virtual) cần thiết cho việc vẽ. Nếu muốn xây dựng một nơi tổng hợp dẫn xuất từ CDC thì có thể chuyển con trỏ CDC đến hàm nhử trong OnDraw.

DC có độ dài 32 bit, là một cấu trúc dữ liệu lưu trữ thông tin cần thiết mà ứng dụng Window sẽ cần đến khi phải hiển thị kết xuất trên thiết bị.

Những thông tin này liên quan đến việc vẽ. Trước khi sử dụng bất cứ hàm vẽ GDI nào cần phải tạo một DC cho thiết bị.

Các loại DC : DisplayDC, PrinterDC, MemoryDC, MetafileDC

Thư viện MFC cung cấp 5 lớp giúp gói gọn các DC

Lớp CDC, CPaintDC, CMetaFileDC, CClientDC, CWindowDC

CClientDC, CWindowDC: là hai lớp lo việc trình bày lên màn hình. Với CClientDC thì sẽ có một Device Context chèn ảnh xài vào vùng Client. CWindowDC thì có hiệu lực trong cái frame Window

Lớp CDC lo việc in ấn hoặc nhồi

Lớp CPaintDC này là lớp cần thiết cho hàm OnPaint ()

CGDIObject:

Là lớp trừu tượng cơ bản của những lớp nội tổng GDI. GDI cho phép lên nhiều loại thiết bị khác nhau nhờ màn hình, máy in, máy vẽ. Hỗ trợ vẽ vòng cung (curves) đường kẻ (line) các hình đa giác (polygon) các Bitmap và văn bản (text).

Các nội tổng về logic nội GDI cung cấp Stock object:

Pen (vẽ những đường cong)

Brush (có vẽ, dùng để tô những vùng
font (phông chữ dùng hiện trên màn hình)

Logical color (màu sắc)

Bitmaps (vẽ hình ảnh)

Các lớp dẫn xuất GDI:

CBitmap: là một dãy các bit dùng để biểu diễn hình ảnh có thể dùng để vẽ

CBrush : có vẽ nên hình ảnh bởi mảng pixel dùng để tô màu .

CFont : là một tập những tính chất bên ngoài và kích thước.

CRgn : là một vùng mà hình ảnh của nó là hình đa giác (polygon), Ellipse hay số kết nối của hình đa giác và ellipse. Có thể tô màu , kiểm tra có tồn tại trong hình .

Để chọn một đối tượng GDI vào một trường ngôn ngữ ta dùng

CDC:: SelectObject ()

Hàm này trả về con trỏ của đối tượng được chọn trước. Không cho phép chọn lại đối tượng nếu như không chọn một đối tượng mới. Cách giải quyết là lưu lại đối tượng GDI ban đầu bằng một biến tạm giữ lại con trỏ trước sau đó muốn chọn lại đối tượng thì ta gọi lại SelectObject (tạm).

Không nên khởi tạo đối tượng dẫn xuất từ lớp CGdiObject mà chỉ nên khởi tạo những lớp dẫn xuất từ nó. Đối tượng GDI không bị hủy trừ khi hủy Window.

Hộp thoại kiểu Modal và Modeless

Dialog nhờ một Window cũng bao gồm những thức trên trình diện của nó. Có thể gain trên nó gồm Edit, button, list boxes, combo boxes, static text, tree view, progress indicator.. Có thể sắp xếp hay tab tab ưu tiên cho từng mục chọn (Tab order).

Dialog có thể nhiều khi cần bởi con trỏ CWnd hoặc bởi file nguồn . Người sử dụng có thể click vào button hay nhấn vào màn hình , thì nó sẽ gọi thông điệp nên dialog của nó sẽ trả lời. Một hộp thoại bằng việc chọn Tab Resource ở Workspace nếu dialog chọn Dialog thì sẽ thấy xuất hiện ID_ABOUTBOX, ID_DIALOG1. Nếu tạo ứng dụng MDI thì sẽ có thêm . Ta có thể chọn loại Dialog khác bằng cách vào menu Tool\ Customize chọn loại Dialog.

- Modal Dialog: Loại này thông báo khi mở dialog loại này nếu muốn qua window khác thì phải đợi dialog lại. Khi xây dựng Dialog có thể đặt tên biến cho từng bộ phận trên nó (nên lấy từ bảng Control), gain các giá trị min, max. Ứng với dialog thì phải tạo lớp mới nên dẫn xuất từ CDialog . Gain tên biến cho các bộ phận trên nó bằng cách vào ClassWizard chọn tên lớp dialog làm lớp dẫn xuất, nhấn nút Add Variable (công biến) thì sẽ xuất hiện bảng liệt kê các bộ phận

trên dialog đôi khi dùng mã ID nữa để tránh nhầm lẫn muốn công biến sau khi khai tên biến. Nếu thêm hay thay đổi những khai báo mã như của biến cần vào properties bằng menu View \ properties hoặc doubleclick vào.

Còn sắp xếp các biến đặt trên bảng việc chọn các nội tổng muốn sắp xếp. Khi xây dựng xong cần phải có một method để gọi nội Tổng hàm thành viên của CDialog::DoModal () trong thời phải khai báo #include tên lớp dialog vào nơi gọi nội

- Modeless Dialog: Dialog kiểu này cho phép không đóng khi không làm

việc trên nữa.



Kiến trúc Document _View.

Mà của lớp Document tổng tài với menu File Menu và File Save. Lớp Document nhớ và ghi dữ liệu của nội tổng Document (Application Framework chịu trách nhiệm trình bày hợp thoại File Open, File Save ; môi trường, nhớ ghi File). Còn lớp View tổng tài với lớp Document gắn liền nội trình bày ứng dụng , Printer I/O.

Số tổng tài giữa Document và View :

Khi nội tổng Doc gửi dữ liệu mà nội tổng View trình bày dữ liệu và cho phép sửa chữa. Một số hàm chức năng quan trọng như

Cview::GetDocument

Nội tổng View chỉ gắn liền duy nhất một nội tổng Doc. Hàm này cho phép cho phép ứng dụng nhìn thông qua View đến Doc của nội Tổng View nhận message mà user nhập dữ liệu môi vào Edit. Th view phải nối với nội tổng Doc cập nhập lại dữ liệu. Hàm này cung cấp con trỏ Doc để sử dụng tiếp cần hàm thành viên của Doc hay hay biến thành viên.

CDocument::UpdateAllViews

Nếu dữ liệu Doc thay đổi vì lý do nào nội tại cái View phải cập nhập lại dữ liệu hiển hành.

Nếu UpdateAllViews được gọi từ hàm chức năng từ lớp Doc Th tham số pSender là (NULL), nếu được gọi từ hàm dẫn xuất từ lớp View Th pSender là (this).

Cview::OnUpdate

Này hàm Virtual được gọi bởi application Framework trả lời hàm UpdateAllView. Còn để gọi trực tiếp dẫn xuất từ lớp View. Hàm này lấy dữ liệu của Doc sau nội cập nhập lại dữ liệu thành viên của View nếu khi phân xử thay đổi.

CView::OnInitialUpdate

Này là hàm Virtual Cview được gọi khi ứng dụng khởi tạo, khi user chọn New từ File menu và khi user chọn Open từ File Menu. Hàm này sẽ gọi OnUpdate. Hàm còn lại được gọi nhiều lần.

CDocument::OnNewDocument
Framework gọi hàm Virtual này sau khi nội tổng Doc được khởi tạo lần đầu và khi user chọn New từ File menu. Đây là nơi có thể xác lập giá trị đầu của biến dữ liệu

Khung làm việc chính (Mainframe window) và lớp Document

Trong SDI thì View window nằm trong một window khác và là khung làm việc chính của ứng dụng. Main Frame Window có thanh tiêu đề (title bar) thanh trình đơn (menu bar).

Các cửa sổ window con khác con có thể thêm toolbar view, view window, Status bar view. Khung làm việc chính của ứng dụng chứa kiến sở tổng tại giữa frame và View bởi việc gói thông điệp từ Frame nên View. Các nội tổng Document được biểu diễn bởi lớp dẫn xuất từ lớp cơ sở CDocument. Nội tổng View và Frame được gắn chặt với nhau bằng Document Template.

Lớp Document sẽ cần giới thiệu kiến mà ứng dụng cần nên. Hoạt trình bày những thay đổi tiến hành trên dữ liệu cho View. Trên dữ liệu trên các tập tin, trình bày dữ liệu thành trang in. Việc này cho phép tận dụng khả năng serialization của MFC nhờ một số chức năng được trình sẵn ví dụ menu file: New, Open, Save, Save as.

Nội với dữ kiện phức tạp có thể dùng thêm biến kiểu COBlist. COBlist là một lớp kiểu cấu trúc dữ liệu mà nội tổng là những liệt kê (list) hoạt động theo kiểu kết nối. Việc truy xuất, cập nhập, sửa, xóa cũng khá thuận lợi.

Cách để truy xuất nội tổng document từ trong long ứng dụng. Tại các nội tổng view nếu được gắn liền với một nội tổng document khi chúng được tạo. Có thể xem nhập vào document được gắn liền với view bằng việc gọi hàm

Cview:: GetDocument ()

Hay

CFrameWnd :: GetActiveDocument()

Window menu và Keyboard Accelerators (phím tắt)

- Có thể tìm thấy ở Tab Resource chọn menu. Mặc định thì menu này có sẵn một số và cho phép thêm vào hay hiệu chỉnh chúng. Đặt mã ID cho chúng bởi vì Window kết nối với menu bằng mã tắt tổa cho chúng hay ghi những câu nhắc nhở hiện ra khi dùng chuột chập tới trong lúc thức thì. Nếu thay đổi những thuộc tính nên vào properties của chúng. Tạo hàm thành viên trình bày những hoạt động trong hàm. Dùng class Wizard để sinh ra vào lớp view (Chọn ID của menu mà muốn đặt tên).

Còn hai mức chọn lệnh :
COMMAND_UI, ON_UPDATE_COMMAND_UI (nếu chọn lệnh này thì menu sẽ
xác định menu nào có thể hay không thể thực thi bằng việc làm mờ đi)

- Nếu nhấn phím tắt thì sau khi nhấn vào menu rồi với phím tắt nào đó ghi trong
bảng properties thì chọn lại Accelerator trong tab Resource. Mỗi bảng liệt kê các
ID của menu cùng với phím tắt của nó. Nó sẽ xuống cuối bảng và nhấn
Enter thì sẽ làm xuất hiện bảng Accel properties. Trong bảng này nhấn vào mã ID
hoặc click vào mũi tên bên cạnh và chọn ID, nhấn vào phím tắt tổng thì nó sẽ
nhấn vào menu. Sau khi hoàn thành lúc thực thi nó sẽ sử dụng phím tắt.

Toolbar và Status bar

- Toolbar là một đối tượng của CToolBar, còn Status bar là đối tượng của lớp
CStatusBar. Còn hai lớp nếu nó dẫn xuất từ lớp CControlBar mà lớp này lại nó
dẫn xuất từ lớp CWnd.

ControlBar nhìn vào trí thanh như khi trong Frame window, có thể thay đổi kích
thước nhìn vào trí khi cần sửa đổi hoặc thay đổi. Khung hoạt động
của ứng dụng sẽ có công việc khởi tạo, hủy bỏ của đối tượng này. AppWizard
sinh mã thanh như khi trong file MainFrm.cpp và MainFrm.h. Toolbar bao gồm
một nhóm các button có hình ảnh đặt trong nó

Những hình ảnh của button được chứa trong một single Bitmap mà file resource
của ứng dụng có thể tiếp cận được. Khi button được click thì nó sẽ gọi thông điệp
lệnh như vai trò của menu và keyboard accelerators.

Một thông điệp UPDATE_COMMAND_UI để cập nhật trạng thái của button.

Hình ảnh của Toolbar có kích thước 15 pixel chiều cao và chiều rộng 16 pixel.

Khung hoạt động của ứng dụng sẽ cung cấp những viên, màu của button.

Toolbar được gắn với bitmap resource và trong file RC

Nhắc và viết Document (SAVE)

Serialization:

Này là quá trình Saving và Restoring đối tượng. Hàm này load dữ liệu hoặc
trở dữ liệu vào đối tượng Archive. Application Framework sẽ gọi hàm chức năng
Serialize của lớp Doc trong suốt quá trình xử lý File Open hay File Save. Thợ viết
MFC cho phép khi Application framework gọi Serialize cho một đối tượng riêng
biệt thì sẽ lưu trên đĩa hoặc ngược lại đưa ra. Không nên không phải thay thế cho hệ
thống cơ sở dữ liệu. Tất cả mọi đối tượng gắn với Doc sẽ được lưu hay ghi vào đĩa
một cách tuần tự. Serialize ngược và ghi dữ liệu nhớ theo đó.

Application Framework không Doc lại (nếu tổng của lớp dẫn xuất từ CDocument). Lúc chọn Save hay Open từ menu File thì Application framework tạo một tổng CArchive, sau đó sẽ gọi hàm serialize chuyển tham khảo vào một tổng CArchive không thời serialize mỗi biến thành viên của nó

Ứng dụng không thể tiếp cận trực tiếp nữa I/O mà thay vào đó tiến trình xử lý serialization. Gọi hàm Serialize vào một tổng CFile là một tổng Archive (của lớp CArchive). Một tổng CArchive nắm giữ dữ liệu cho một tổng CFile, nó duy trì một Flag để chỉ ra Archive đang trữ hay load về từ đĩa. Application Framework quản lý việc khởi tạo một tổng CFile và CArchive, mô phỏng File cho CFile và gắn Archive với file. Lớp có thể serialize phải được dẫn xuất trực tiếp hay gián tiếp từ CObject và phải khai báo macro DECLARE_SERIAL, IMPLEMENT_SERIAL.

Tham số tham khảo ar của CArchive giúp phân biệt một tổng Archive của ứng dụng. hàm chức năng CArchive::IsStoring báo archive có đang storing hay loading. Toàn bộ

<< : nghĩa đang restoring;

>> : đang loading.

```
void CLuu_doDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        // TODO: add storing code here
    }
    else
    {
        // TODO: add loading code here
    }
}
```

Printing và Print Preview

Print Preview là một đặc điểm của thư viện MFC cho không phải của Window. Không thể ảnh hưởng những ảnh hưởng của Print Preview một với

chương trình. Việc xác định vị trí thì dựa trên device context của Printer. Sau khi chọn Font tổng cộng, chương trình trình bày những nội dung trong window Print Preview. Application framework làm hầu hết những công việc cho Printing và Print Preview

Printer Device Context và hàm chức năng CView::OnDraw

Khi chương trình in trên máy in thì nội dung một nội dung device context của lớp CDC. Nếu trình bày hàm OnPaint gọi OnDraw và device context lúc này sẽ là display context. Nếu in thì OnDraw được gọi bởi OnPrint của CView với device context của printer nhờ một tham số. Trong chế độ Print Preview tham số trong OnDraw thực sự là con trỏ của chế độ nội dung CPreviewDC.

Hàm CView::OnPrint

Hàm này sẽ gọi OnDraw (OnDraw có thể dùng cái Device Context nếu trình bày và cái in ấn. Việc in sẽ phải được xác lập trước khi OnPrint được gọi. Có thể thêm vào những phần trình bày nhờ tọa của trang, header, Footer. Tham số trong OnPrint là

1. Một con trỏ của Device Context
2. Một con trỏ của chế độ thông tin in (CPrintInfo) bao gồm kích thước trang giấy, số trang hiện thời và số trang lớn nhất.

Khi gọi hàm có thể không cần gọi OnDraw vì có thể có những phần riêng biệt so với trình bày. Application framework gọi hàm OnPrint cho mỗi trang được in với số trang hiện thời trong cấu trúc CPrintInfo.

Bắt đầu và Kết thúc việc In Ấn

Application Framework sẽ gọi hàm chức năng của Cview là OnPreparePrinting, OnBeginPrinting khi bắt đầu việc In (AppWizard sinh ra OnPrepare Printing, OnBeginPrinting, OnEndPrinting nếu chọn Printing and PrintPreview).

OnPreparePrinting được gọi trước khi trình bày hộp thoại Print. Nếu biết số trang gọi CPrintInfo::SetMinPage, CPrintInfo::SetMaxPage. Số trang được xuất hiện trong dialog Print nếu user chọn.

OnBeginPrinting được gọi sau khi hộp thoại Print thoát. Ở hàm này nên tạo nội dung GDI nhờ là Font cần cho chương trình chạy nhanh hơn do không phải tạo lại Font cho mỗi trang.

OnEndPrinting hoặc gọi khi In xong trang cuối cùng. Ở đây nên xóa nội
tổng GDI của OnBeginPrinting.



[Home](#)

[Mục Lục](#)