Curs 6

Arhitectura Document/View

Concept

Obiect aplicatie -> mesaje

Fereastra cadru (meniu, toolbar, etc.), nivelul cel mai inalt al aplicatiei - CFrameWnd

Vizualizarea (fereastra descendenta)= zona client CView, CScrollView

Obiectul document = memoreaza datele aplicatiei CDocument

Comunicare bidirectionala intre vizualizare si document

Obiectul aplicatie - CWinApp

```
Functia InitInstance
CSingleDocTemplate* pDocTemplate;
pDocTemplate = new CSingleDocTemplate
( IDR_MAINFRAME,
RUNTIME_CLASS (CAppDoc),
RUNTIME_CLASS (CMainFrame),
RUNTIME_CLASS (CAppView) );
 AddDocTemplate (pDocTemplate);
```

Explicatii

Se creaza un sablon de document SDI din CSingleDocTemplate

Identifica clasa document folosita pentru a gestiona datele aplicatiei.

Sablonul de document memoreaza ID-ul resurselor pe care cadrul de lucru il folosest pentru a incarca meniul, toolbarul, acceleratorii, etc.

RUNTIMECLASS

returneaza un pointer la o structura CRuntimeClass, pentru clasa specificata, iar cadrul de lucru va crea obiecte din aceasta clasa in momentul executiei.

ProcessShellCommand (cmdInfo) proceseaza linia de comanda si apeleaza **CWinApp::OnFileNew()** – pentru a crea un document vid sau **CWinApp::OpenDocumentFile()** pentru a incarca documentul specificat in linia de comanda.

Obiectul document

Datele documentului sunt memorate in variabile membru ale clasei derivate din CDocument.

Pot fi folosite clase ajutatoare din MFC, CBzteArraz, CString, CList, etc.

Operatii - document

GetFirstViewPosition Returneaza o valoare de tip POSITION ce poate fi folosita in *GetNextView* pentru a incepe enumerarea vizualizarilor asociate documentului.

GetNextView Returneaza un pointer la *CView,* urmatoarea vizualizare asociata documentului.

Exemplu – redesenare fiecare vizualizare

```
// POSITION pos = GetFirstViewPosition();
// CView* pFirstView = GetNextView( pos );
void CMyDoc::OnRepaintAllViews()
{   POSITION pos = GetFirstViewPosition();
while (pos != NULL)   {
     CView* pView = GetNextView(pos);
     pView->UpdateWindow();   }
}
```

Acelasi lucru se poate face si apeland functia:
UpdateAllViews(NULL);

Exemplu – redesenare fiecare vizualizare

```
POSITION pos = GetFirstViewPosition();
while (pos != NULL) {
 CView* pView = GetNextView(pos);
 pView->OnUpdate(pSender, lHint, pHint);
 }
```

OnUpdate() este o functie membru din CView.

Operatii - document

GetPathName regaseste numele fisierului si calea completa; daca documentul nu are nume returneaza sirul vid.

GetTitle Regaseste titlul documentului sau sirul vid.

IsModified Semnaleaza faptul ca documentul contine date salvate sau nu.

*SetModifiedFlag*S Seteaza sau sterge flag-ul ce mentine starea documentului (date modificate si nesalvate).

UpdateAllViews Actualizeaza toate vizualizarile asociate documentului, apeland pentru fiecare vizualizare functia OnUpdate.

CDocument – functii virtuale

OnNewDocument Apelata de cadrul de lucru cand se creaza un document nou. Aici putem face initializari specifice documentului, cind acesta este creat.

OnOpenDocument Apelata de cadrul de lucru cand un document este incarcat din disc (Open din meniul File, apeleaza indirect Serialize).

DeleteContents Apelata de cadrul de lucru cand se sterge continutul unui document. Aici putem elibera anumite resurse cand documentul este inchis.

Serialize Apelata de cadrul de lucru pentru a serializa documentul (scriere/citire in/din disc).

Serialize

```
oid CMyDoc::Serialize (CArchive& ar)

(ar.IsStoring ()) {
  // Scrie datele in disc
  ar << m_strNume << m_strPrenume;

se { // Citeste datele din arhiva
 ar >> m_strNume >> m_strPrenume;  }
```

CArchive

Pentru structuri si alte tipuri de date neserializabile putem folosi functiile *Read* si *Write* din CArchive. (vezi si *ReadString* si *WriteString)*

De asemenea putem folosi *CArchive::GetFi* pentru a obtine un pointer la *CFile* pentru a interactiona direct cu fisierul atasat arhivei.

Obiectul vizualizare

Responsabil cu vizualizarea datelor.

GetDocument() – functie ce realizeaza legatura intre date (document) si vizualizare. Vizualizarea identifica documentul cu ajutorul acestei functii.

Cadrul de lucru memoreaza un pointer la obiectu document asociat, in data membru *m_pDocume* din vizualizare si expune acest pointer prin funct GetDocument().

CView – functii virtuale

OnDraw Apelata de cadrul de lucru pentru a afis
datele documentului.

OnInitialUpdate Apelata cand o vizualizare este
atasata prima data la un document. Aici putem
initializa obiectul cand documentul este creat.
(Faceti analogie cu mesajul WM_INITDIALOG d
la casete de dialog).

OnUpdate Apelata cand datele documentului s-
au modificat si vizulizarea trebuie actualizata. Se
pot desena numai parti ale vizualizarii.

OnDraw()

```
void CMyView::OnDraw (CDC* pDC)
{ CMyDoc* pDoc = GetDocument ();
CString string = pDoc->GetString ();
CRect rect; GetClientRect (&rect);
pDC->DrawText (string, rect,
DT_SINGLELINE ¦ DT_CENTER ¦
DT_VCENTER); }
```

OnDraw()

Acelasi cod este folosit pentru afisarea intr-o fereastra, preview, listarea la imprimanta.

Mesajul WM_PAINT este tratat in cadrul acestei functii.

Cadrul de lucru furnizeaza contextul de dispozitiv necesar.

OnInitialUpdate()

Se pot face initializari inainte ca documentu sa fie vizualizat.

Apeleaza intern **OnUpdate()**.

OnUpdate() apeleaza intern **Invalidate()**, echivalentul trimiterii mesajului WM_PAINT la fereastra. Aceasta functie este apelata si din **UpdateAllViews()**. In general aceasta functie nu se suprascrie.

Vizualizari

Active si **inactive**. La un moment dat exista o singura vizualizare activa.

O vizualizare poate determina cand este activata si dezactivata prin suprascrierea functiei ***CView::OnActivateView***.

Fereastra cadru foloseste functiile **CFrameWnd::GetActiveView()** si **CFrameWnd::SetActiveView()**.

Obiectul fereastra cadru

Defineste spatiul fizic al aplicatiei pe ecran s
serveste ca un container pentru o
vizualizare. O aplicatie SDI foloseste o
singura fereastra cadru – *CFrameWnd,*
fereastra de nivel inalt a aplicatiei.

CFrameWnd

Redimensioneaza vizualizarea cand fereastra cadru este redimensionata;

Gestioneaza toolbar-ul, bara de stare, etc.

Salveaza documentul cand aplicatia se termina.

Gestioneaza documentul;

Gestioneaza vizualizarea.

Crearea dinamica a obiectelor

Obiectele ce vor fi create in mod dinamic trebuie sa aiba declarate macrourile DECLARE_DYNCREATE (nume_clasa) (in .h) si IMPLEMENT_DYNCREATE (nume_clasa, clasa_de_baza) (in .cpp) si sa fie derivate din CObject.

Exemplu:

UNTIME_CLASS (CMyClass)->CreateObject ();

Creare dinamica obiecte

RUNTIMECLASS foloseste o macrosubstitutie, ca in exemplul de mai jos (acest cod nu este corect in C++):

```
CString strClassName = _T ("CMyClass");
CMyClass* ptr = new strClassName;
```

Creare dinamica obiecte

DECLARE_DYNCREATE (CMyClass) adauga urmatoarele la
clasa:

blic: static const AFX_DATA CRuntimeClass classCMyClass;
tual CRuntimeClass* GetRuntimeClass() const;
atic CObject* PASCAL CreateObject();

IMPLEMENT_DYNCREATE initializeaza structura *CRuntimeClass* cu informatii despre numele clasei si marimea instantei clasei.

IMPLEMENT_DYNCREATE (CMyClass, CBaseClass) Se expandeaza astfel:

Object* PASCAL CMyClass::CreateObject()
eturn new CMyClass; }

Rutarea comenzilor

Mesaje de comanda in MFC este analog cu WM_COMMAND (comenzi din meniuri, tast acceleratoare, butoane din toolbar).

Mecanismul de rutare al comenzilor ne permite sa punem fct. ce trateaza mesaje in clasa cea mai potrivita scopului ales de noi (vizualizare, aplicatie si chiar document).

Rutarea comenzilor

Cand fereastra cadru primeste un mesaj WM_COMMAND, aceasta apeleaza functia virtuala *OnCmdMsg* pentru a incepe procesul de rutare. Implemntarea acestei functii in *CFrameWnd* arata astfel:

```
OL CFrameWnd::OnCmdMsg(...) {
Pump through current view FIRST.
iew* pView = GetActiveView();
pView != NULL && pView->OnCmdMsg(...))
                        // se trimite la CDocument si la CDocTemplate
 return TRUE;
hen pump through frame.
CWnd::OnCmdMsg(...))
 return TRUE;
ast but not least, pump through application.
WinApp* pApp = AfxGetApp();
pApp != NULL && pApp->OnCmdMsg(...))
 return TRUE;
urn FALSE;  // se trimite la ::DefWindowProc()}
```