

Schritt für Schritt Anleitung für Datenverwaltung unter MFC

Ziel, Inhalt

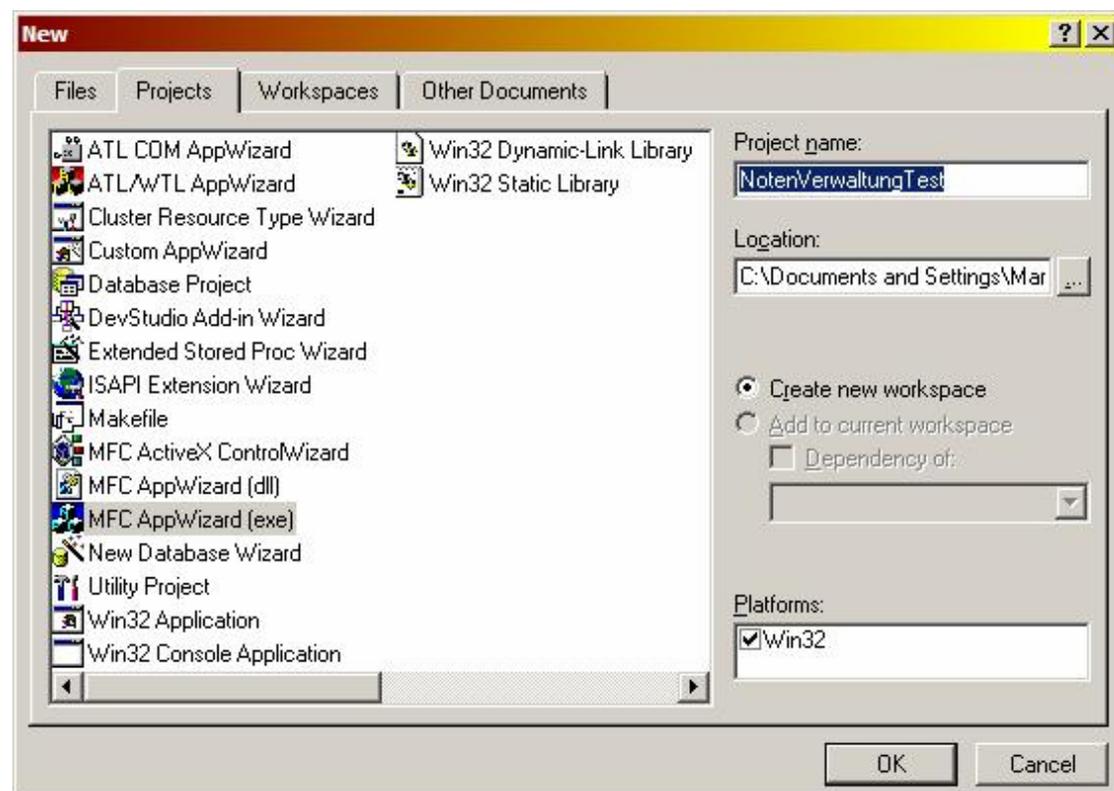
- Wir erhalten hier eine Anleitung um ein einfaches MFC Programm zu erstellen, das als Ausgangspunkt für das Projekt Notenverwaltung dienen kann.

Schritt für Schritt Anleitung für Datenverwaltung unter MFC	1
Ziel, Inhalt	1
Einfache Notenverwaltung, Schritt für Schritt	2
Projekt erzeugen	2
Single - Document Interface	2
Neue Klasse CNote	6
Elemente der Klasse CNote	7
Dialog zum Erzeugen von Noten	9
Zugriff auf den Notenwert im Dialog	13
Menupunkt einfügen	13
Menupunkt behandeln	14
Document Klasse ergänzen	14
CObArray in der Document Klasse	15
Zugriff auf die Noten	16
Anpassen der View Klasse	16

Einfache Notenverwaltung, Schritt für Schritt

Projekt erzeugen

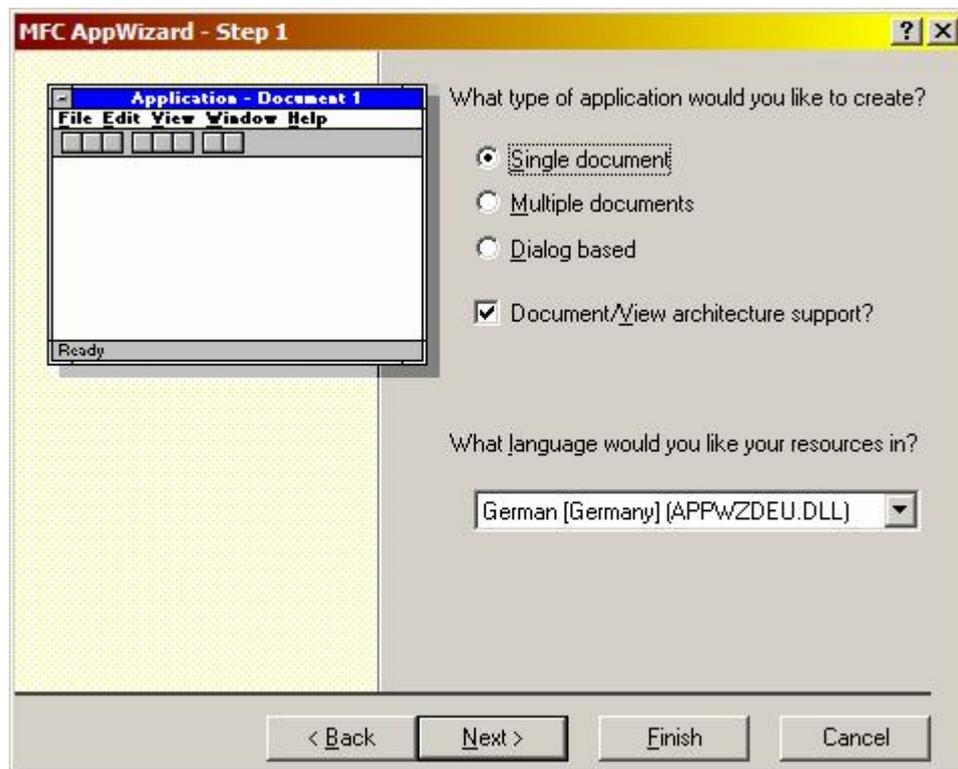
Wir erzeugen ein Projekt mit dem MFC - Applikationsassistenten. Du hast die Möglichkeit dieses Projekt zu verwenden um daraus später die endgültige Notenverwaltung zu bauen, oder du kannst es als paralleles Projekt dazu betreiben. Ich verwende den zweiten Ansatz, wo ich ein zweites Projekt erzeuge und die gewonnenen Erkenntnisse danach im endgültigen Projekt verwende.



Applikationsassistent

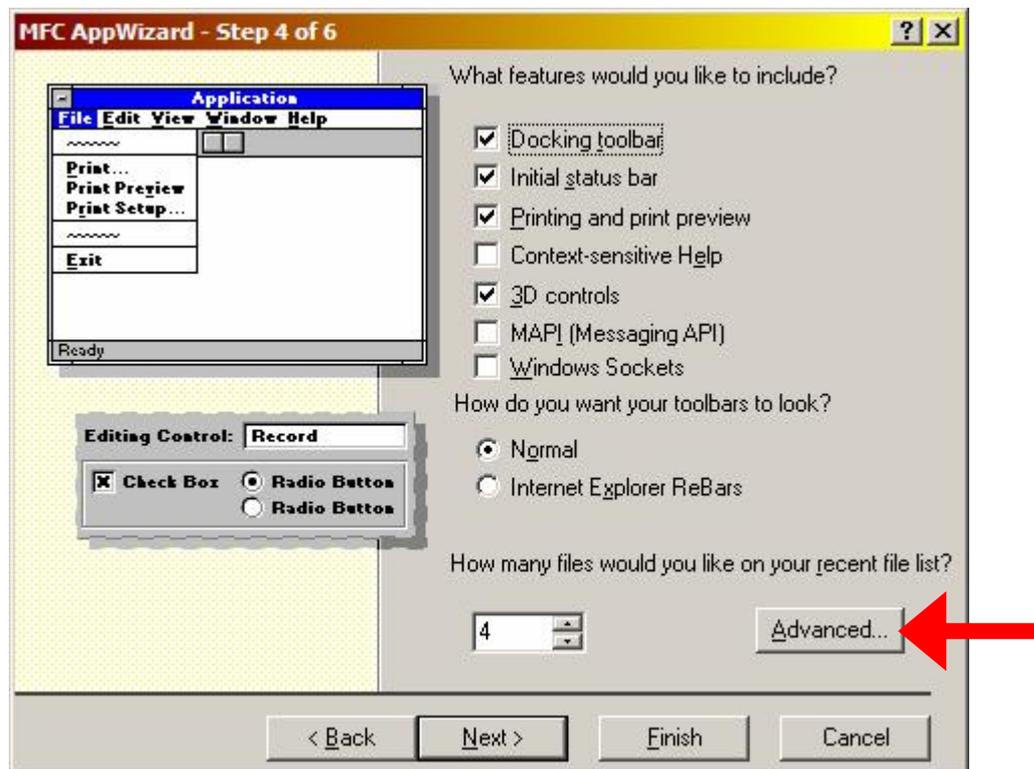
Single - Document Interface

Unser Programm soll nur ein Dokument geöffnet halten können. Sollen mehrere Dokumente bearbeitet werden, muss der Benutzer das Programm noch einmal starten. Diese Interface - Art ist moderner und unterstützt das die Idee an einem Dokument zu arbeiten, anstelle vom Arbeiten mit einem Programm.



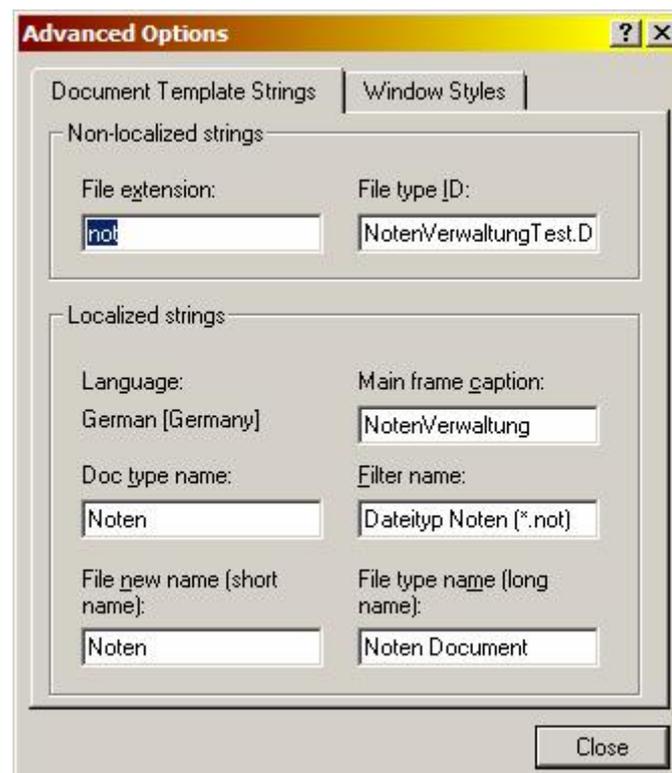
Die Auswahl der „Single document“ Eigenschaft

Bei Schritt 2 und Schritt 3 belassen wir die Grundeinstellungen, „Datenbankunterstützung“ nichts anwählen, obwohl, die Schulnoten schlussendlich in eine Datenbank gehören würden. Auch Schritt 3 ist mit den voreingestellten Werten in Ordnung. So dass wir nun zu Schritt 4 gelangen:



Hier wählen wir den Knopf für weitere Einstellungen

Folgender Dialog erscheint nun:

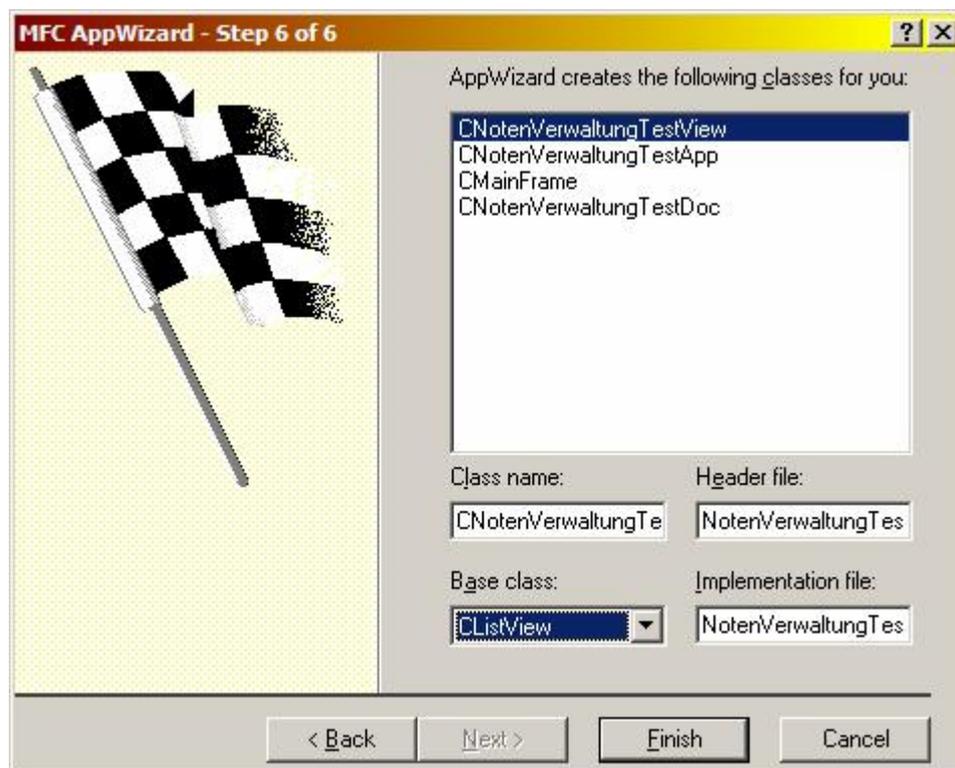


Dialog für weitere Einstellungen

Bei diesem Dialog können wir unser Programm mit einer Dateiendung verbinden. Auch können wir verschiedene weitere Einstellungen vornehmen, wie der Name des Hauptfensters („NotenVerwaltung“).

Nach Schliessen dieses Dialoges sind die Schritte 4 und 5 einfach mit „weiter>“ zu bestätigen.

Beim Schritt 6 wählen wir noch als View Typ nicht den voreingestellten Typ CView sondern die Klasse CListView. Von dieser Klasse wird unsere View Klasse abgeleitet. Die CListView beherbergt ein List Control. Das List Control wird vom Betriebssystem verwendet um die rechte Seite im Explorer darzustellen. Es kann als Icon-, Detail- und Listenansicht verwendet werden.



Als Basisklasse verwenden wir die CListView Klasse

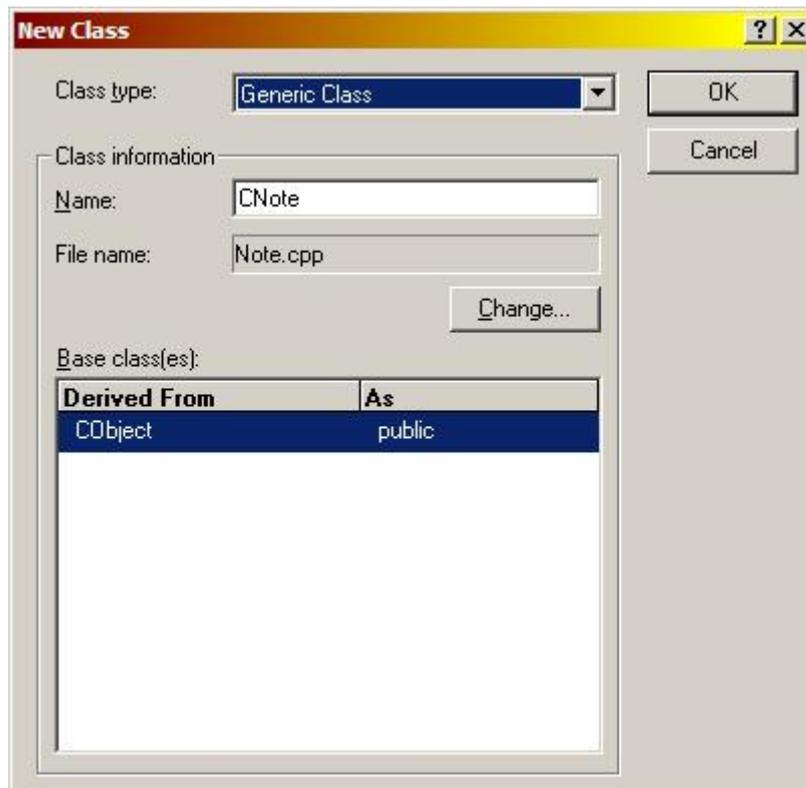
Neue Klasse CNote

Diese Klasse leiten wir von CObject ab, um das Serialieren der MFC zu verwenden. Die Klasse kann mit dem Klassenassistenten erzeugt werden.



Klassenansicht mit Context Menu

Im Assistenten wählen wir als Klassentyp nicht eine MFC Klasse, denn die Klasse CObject steht dann nicht als Basisklasse zur Verfügung. Wähle also „allgemeine Klasse“ als Typ und leite von CObject ab:



Neue Klasse mit CObject als Basisklasse

Nach dem OK erscheint eine Warnung, die man einfach bestätigen kann.

Elemente der Klasse CNote

Mit dem Klassenassistenten kann nun das Datenelement für den Notenwert ergänzt werden.



The dialog box 'Add Member Variable' has a title bar with a question mark and a close button. It contains three input fields: 'Variable Type' with 'double', 'Variable Name' with '_notenWert', and 'Access' with radio buttons for 'Public', 'Protected', and 'Private' (selected). There are 'OK' and 'Cancel' buttons on the right.

Datenelement für die Klasse CNote

Ergänze auch einen Konstruktor mit Parametern und das DECLARE_SERIAL Makro. Dazu gehört dann auch das überschreiben der Serialize Methode und eine Methode asString damit wir einen String erhalten.



The dialog box 'Add Member Function' has a title bar with a question mark and a close button. It contains two input fields: 'Function Type' with 'CString' and 'Function Declaration' with 'asString() const'. Below are radio buttons for 'Access' (Public selected, Protected, Private) and checkboxes for 'Static' and 'Virtual'. There are 'OK' and 'Cancel' buttons on the right.

Die Methode asString einfügen

```
class CNote : public CObject
{
public:
    CNote();
    CNote(double notenWert);
    virtual ~CNote();

    void Serialize(CArchive& ar);

private:
    double _notenWert;

    DECLARE_SERIAL(CNote)
};
```

Die .cpp Datei sieht so aus (nicht vergessen IMPLEMENT_SERIAL):

```
IMPLEMENT_SERIAL(CNote, CObject, 1)

CNote::CNote()
:_notenWert(4.0)
{
}

CNote::CNote(double notenWert)
:_notenWert(notenWert)
{
}

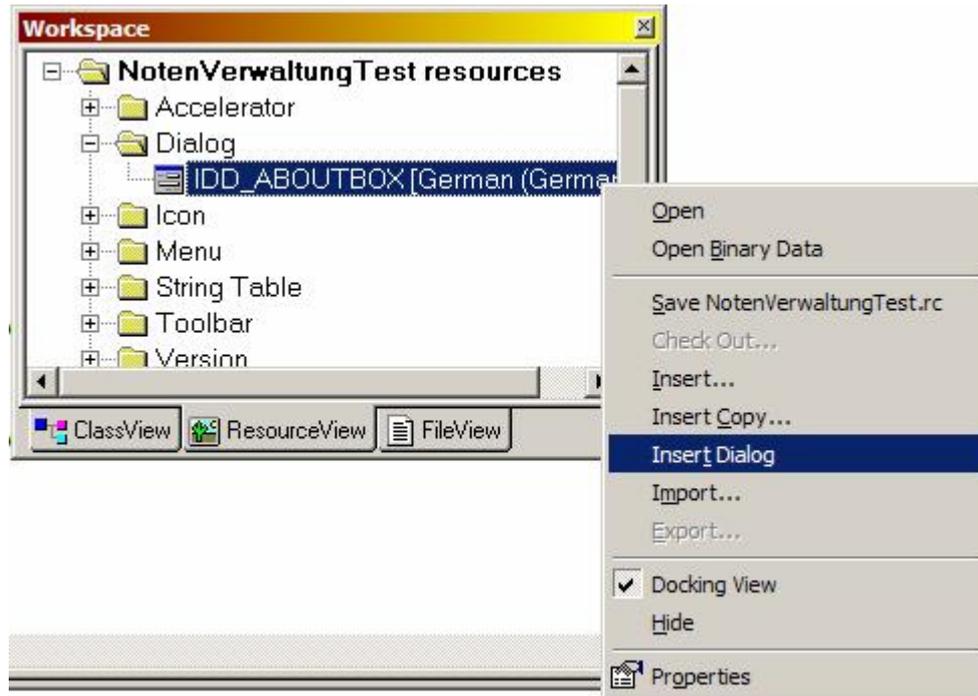
CNote::~CNote()
{
}

void CNote::Serialize(CArchive& ar)
{
    if(ar.IsStoring())
    {
        ar << _notenWert;
    }
    else
    {
        ar >> _notenWert;
    }
}

CString CNote::asString() const
{
    CString string;
    // notenwert als float wert mit
    // zwei Kommastellen
    string.Format("%.2f", _notenWert);
    return string;
}
```

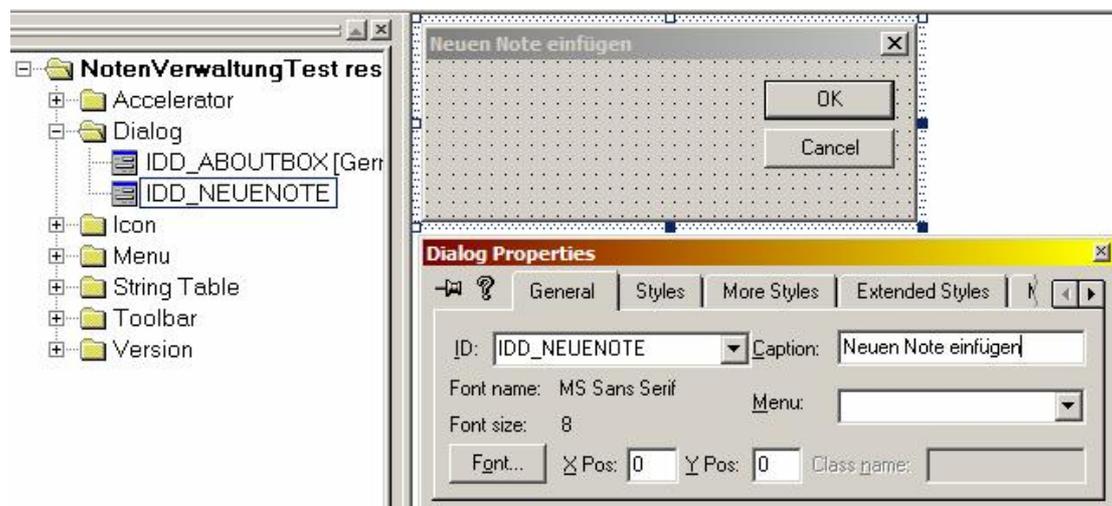
Dialog zum Erzeugen von Noten

Im Ressourceneditor erzeugen wir einen Dialog:



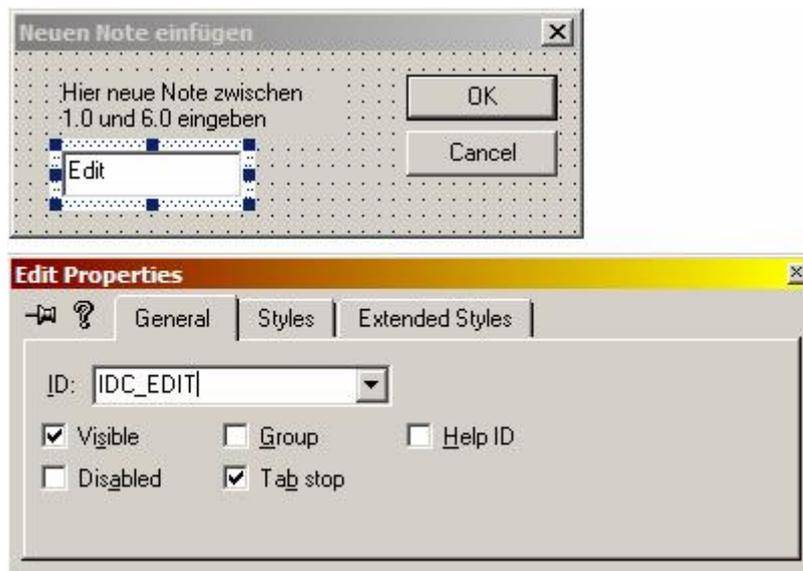
In der Ressourcenansicht kann ein neuer Dialog erzeugt werden

Mit den Eigenschaften zu den einzelnen Controls wählst du die Bezeichnungen wie IDD_NEUENOTE und andere Eigenschaften:



Editieren der Dialog Ressource

Füge eine Editbox ein und einen Text, der dem Benutzer helfen soll.



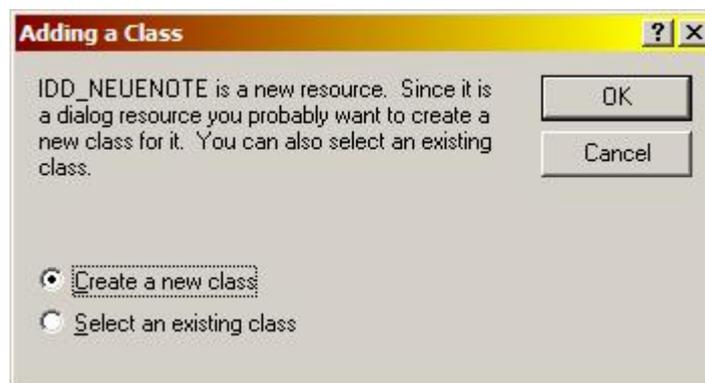
Eigenschaften für die Editbox

Hier noch wie man in einem statischen Text einen neue Zeile erzeugt:

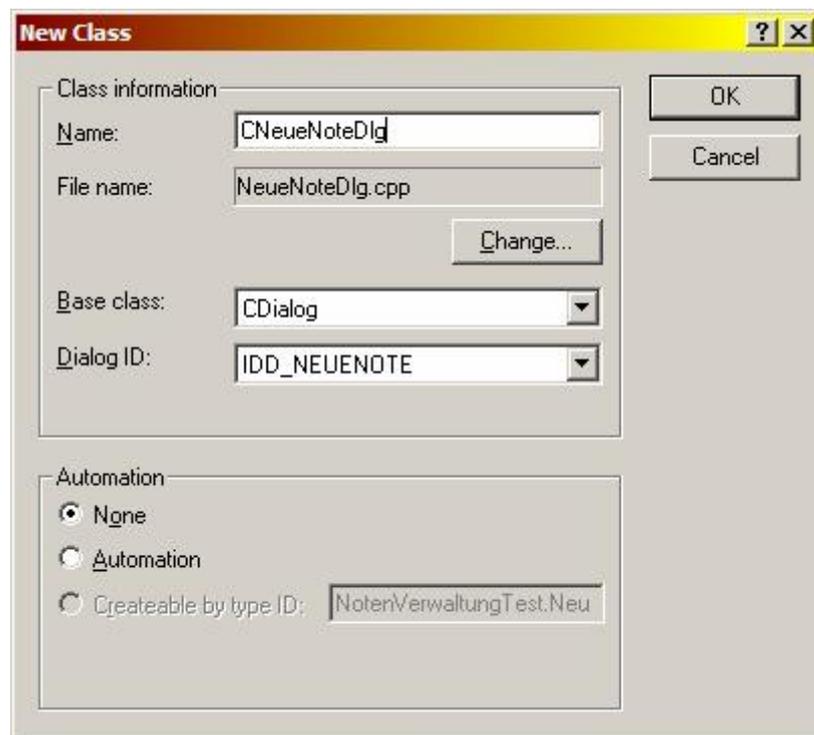


Mehrzeiliger Text mit \n

Durch Doppelklick auf den Dialog will der Klassenassistent einen neue Klasse erzeugen, was wir dankend annehmen:

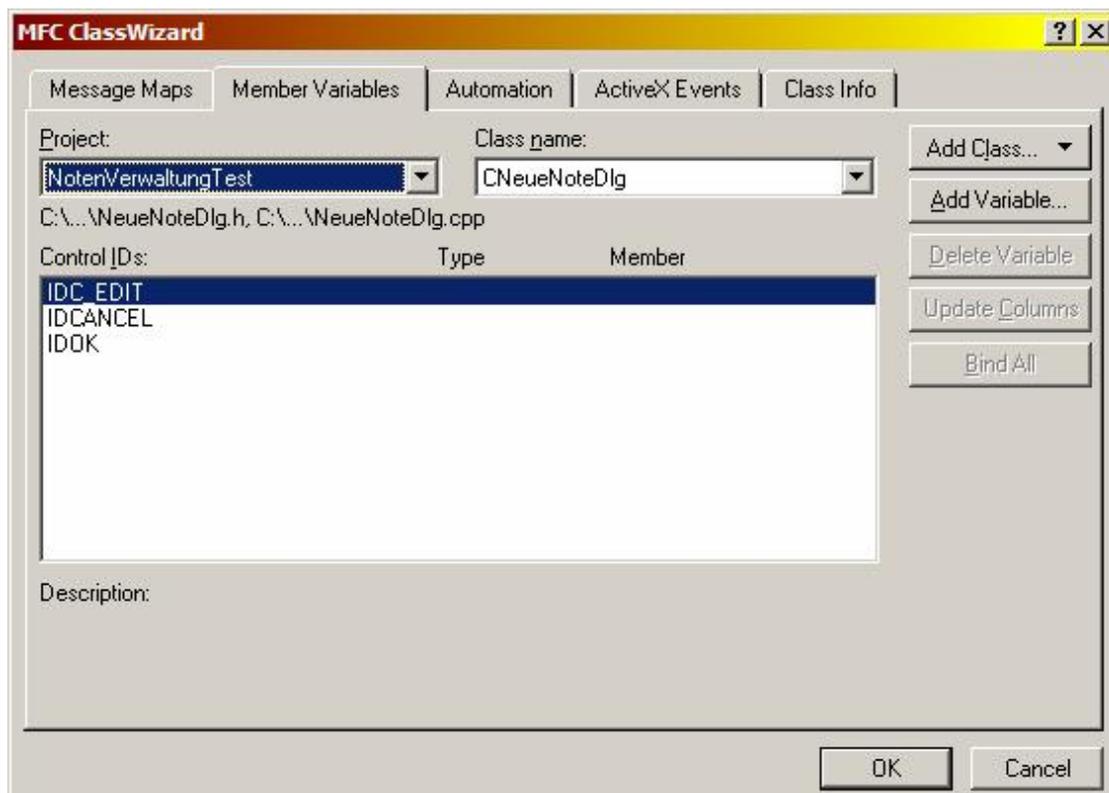


Dieser Dialog erscheint durch Doppelklick auf den Dialog im Ressourceneditor



Der Assistent hilft beim Erzeugen der Klasse

Jetzt gibt es die Möglichkeit in dieser neuen Klasse ein Datenelement für die Editbox einzubauen:

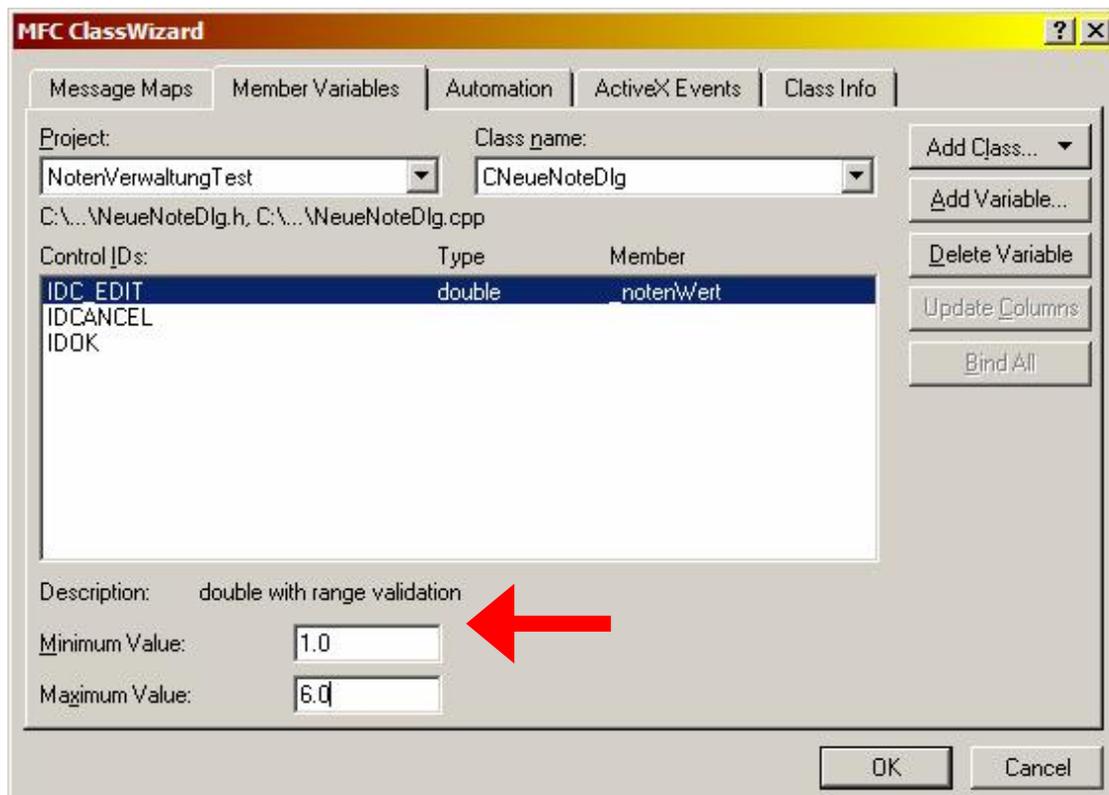


Hier gibt es die Möglichkeit eine Variable für IDC_EDIT zu erzeugen



Datenelement als double - Wert hinzufügen

Wichtig ist, dass wir für das IDC_EDIT nicht ein Control erzeugen (Category) sondern einen Wert mit dem Datentypen double.
Jetzt ist es möglich Grenzen für diese Variable zu definieren.



Unten können nun Grenzen für die Variable angegeben werden

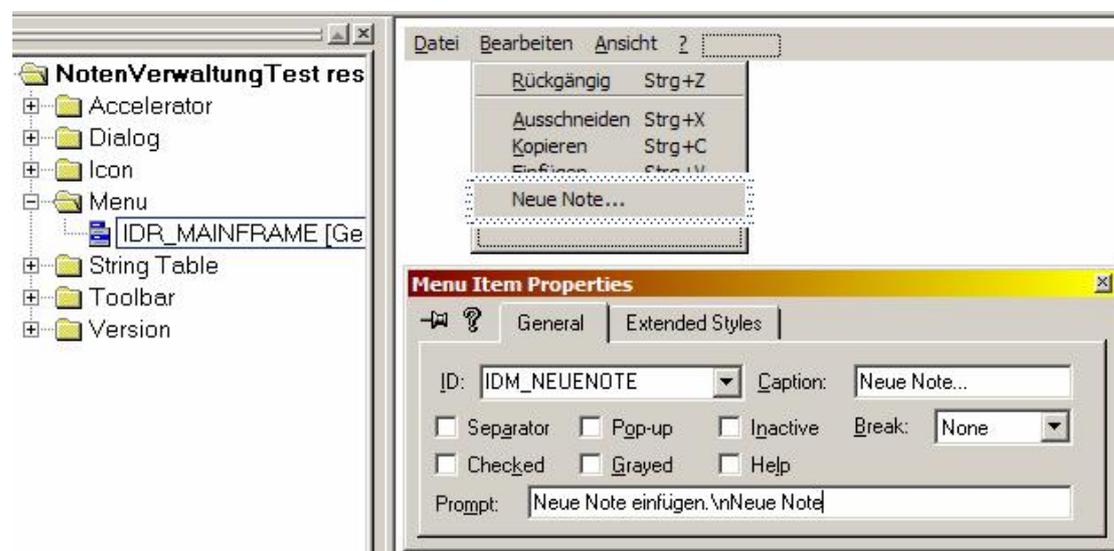
Zugriff auf den Notenwert im Dialog

Mit einer einfachen Get-Methode können wir auf den Notenwert zugreifen. Mit dem Klassenassistenten kann diese in die neue Dialog-Klasse eingefügt werden.

```
double CNeueNoteDlg::getNotenWert() const
{
    return _notenWert;
}
```

So sieht der Code in der Dialog-Klasse aus.

Menupunkt einfügen

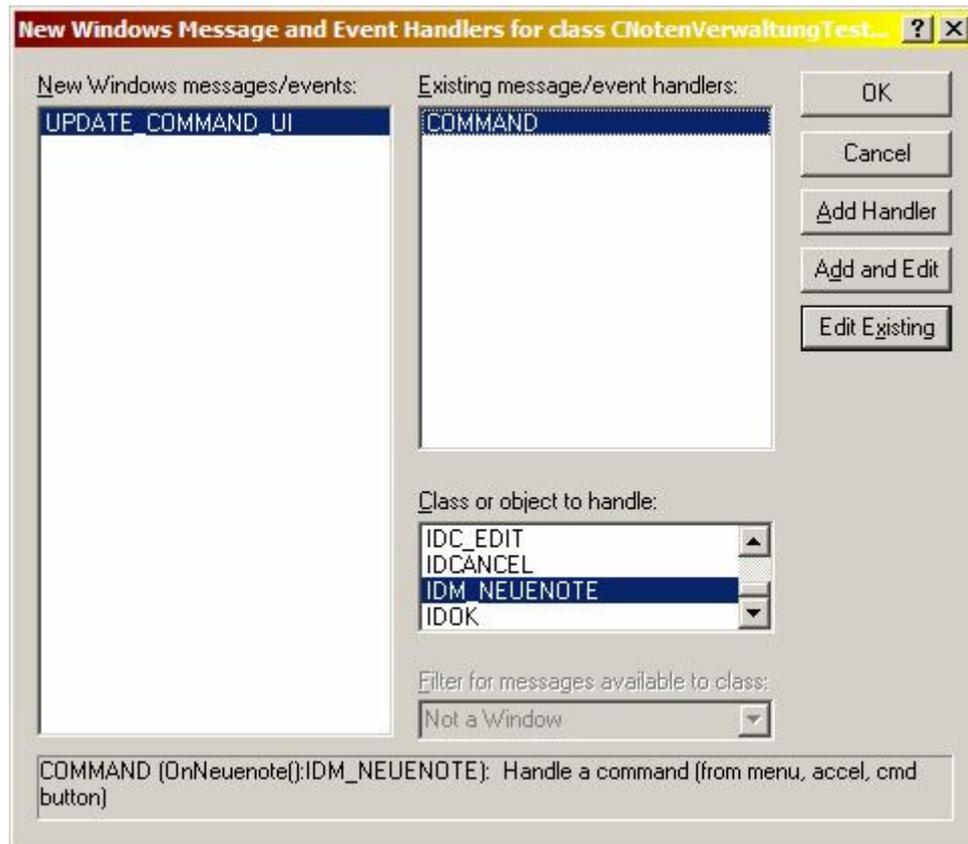


Im Ressourceneditor einen Menupunkt einfügen

Diesen Menupunkt bearbeiten wir in der Document - Klasse.

Menupunkt behandeln

Durch Rechtsklick auf die Document Klasse kann man eine „Nachrichtenbehandlungsroutine einfügen“ und der Assistent erscheint. Wähle rechts unten das IDM_NEUENOTE.



Die Behandlungsroutine für den Menupunkt einfügen

Document Klasse ergänzen

Der Code in der Document Klasse (hier CNotenVerwaltungTestDoc) sieht so aus:

zuerst der nötige #include

```
#include "NeueNoteDlg.h"
```

Und weiter unten:

```
void CNotenVerwaltungTestDoc::OnNeuenote()
{
    CNeueNoteDlg dlg;
    if(IDOK == dlg.DoModal())
    {
        double notenWert = dlg.getNotenWert();
    }
}
```

Es fehlt noch das Erzeugen eines CNoten - Objektes und das speichern in eine Kollektion.

CObArray in der Document Klasse

Mit dem Assistenten fügen wir ein Objekt der Klasse CObArray in die Document Klasse ein:



Datenelement _noten in die Document-Klasse

Ergänze noch folgendes #include:

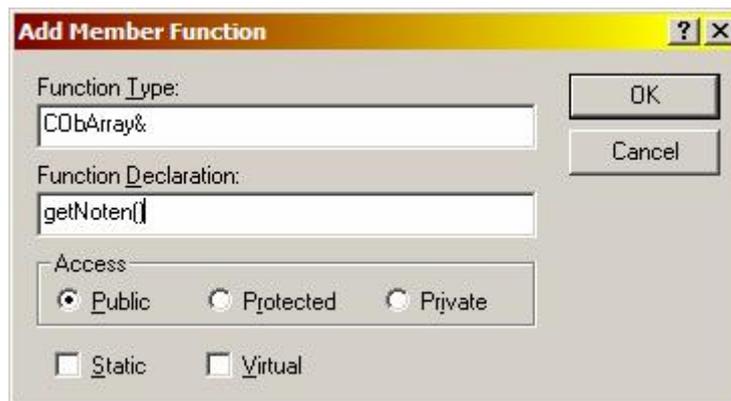
```
#include "NeueNoteDlg.h"  
#include "Note.h"
```

Und der Code in der Document - Klasse, besteht aus Destruktor und der OnNeueNote Methode:

```
CNotenVerwaltungTestDoc::~CNotenVerwaltungTestDoc()  
{  
    const int count = _noten.GetSize();  
    for(int i = 0; i < count; ++i)  
    {  
        CObject* object = _noten[i];  
        delete object;  
    }  
}  
  
void CNotenVerwaltungTestDoc::OnNeuenote()  
{  
    CNeueNoteDlg dlg;  
    if(IDOK == dlg.DoModal())  
    {  
        double notenWert = dlg.getNotenWert();  
        CObject* object = new CNote(notenWert);  
        _noten.Add(object);  
        // Dokument als geändert markieren  
        SetModifiedFlag();  
        // Views benachrichtigen  
        UpdateAllViews(0);    }  
}
```

Zugriff auf die Noten

Es fehlt noch eine Möglichkeit auf das Array zuzugreifen:



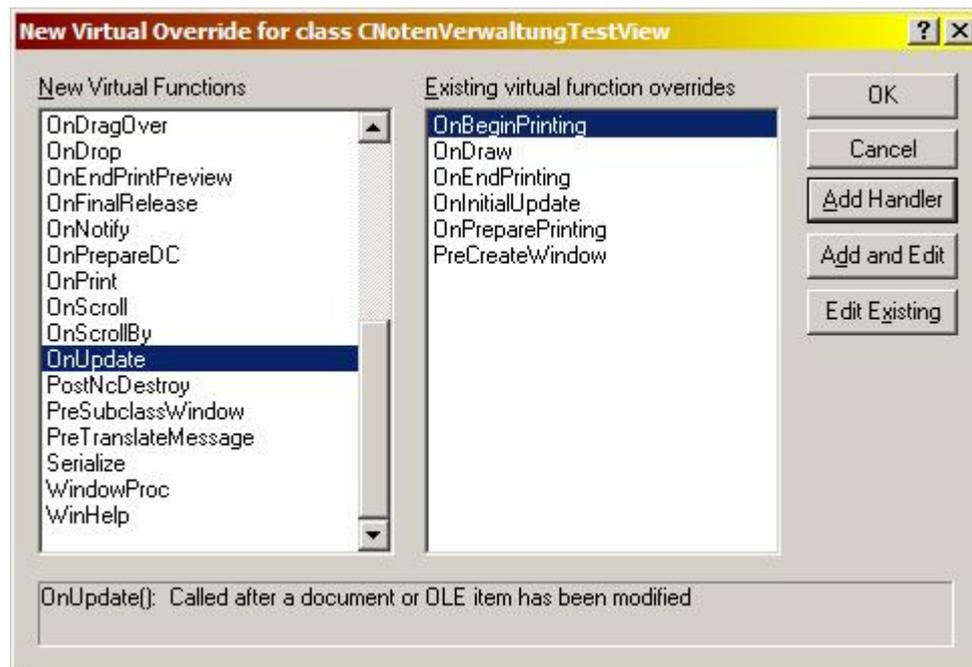
Zugriff auf das Noten Array

Und hier der Code:

```
CObArray& CNotenVerwaltungTestDoc::getNoten()
{
    return _noten;
}
```

Anpassen der View Klasse

Jetzt fehlt noch der Code in der View. Überschreibe zuerst die Methode OnUpdate in unserer View Klasse.



Hier die OnUpdate Methode für die View ergänzen

Füge ein `#include` für die `CNote` Klasse ein:

```
#include "Note.h"
```

Und hier die Methode `OnUpdate`:

```
void CNotenVerwaltungTestView::OnUpdate(CView* pSender, LPARAM  
lHint, CObject* pHint)  
{  
    CListCtrl& list = GetListCtrl();  
    list.DeleteAllItems();  
    CNotenVerwaltungTestDoc* doc = GetDocument();  
    CObArray& array = doc->getNoten();  
    const int count = array.GetSize();  
    for(int i = 0; i < count; ++i)  
    {  
        CObject* object = array[i];  
        if(object->IsKindOf(RUNTIME_CLASS(CNote)))  
        {  
            CNote* note = (CNote*)object;  
            CString notenString = note->asString();  
            list.InsertItem(0, notenString);  
        }  
    }  
}
```

Viel Spass.