Projektarbeit 2

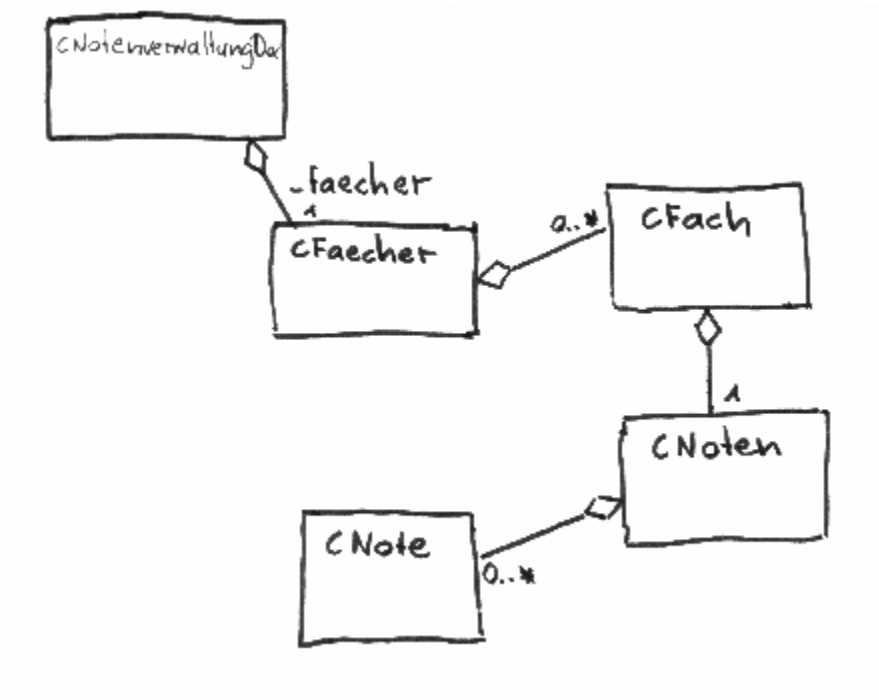
Ziel, Inhalt

- Heute sollten wir die Datenhaltung und das Modell weiter verfeinern.

Projektarbeit 2

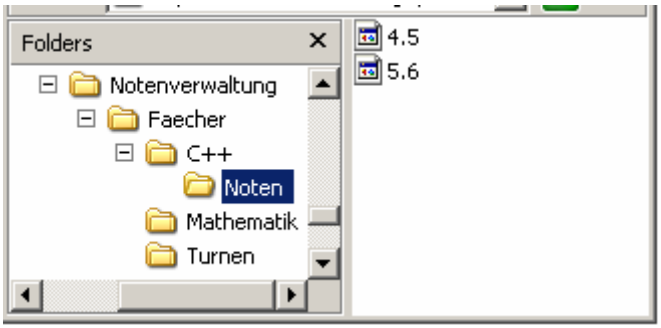
Das Modell

Das Datenmodell, das wir bereits haben sieht etwa so aus:



Einfaches Klassenmodell der Daten

Die Klasse *CFeacher* ist eine map, die eine Beziehung zwischen einem Fachnamen in Form eines CStrings und einem Objekt der Klasse CFach herstellt. Solch eine Struktur lässt sich einfach auch als Baum darstellen.

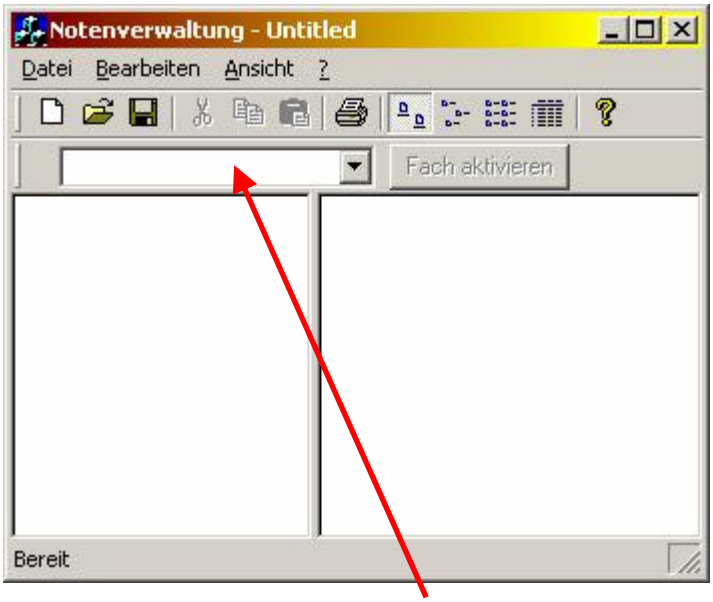


Hierarchische Struktur als Baum

Weitere Elemente im Modell

Unsere bisherigen Bemühungen haben uns in die Nähe einer Explorer-ähnlichen Darstellung gebracht. Dabei haben wir auf der linken Seite eine Baumansicht während auf der rechten Seite verschiedene Ansichten möglich sein werden.

Unsere Bedienungsoberfläche enthält bereits ein Auswahlfeld, mit dem wir ein Fach zum „aktuellen“ Fach machen. Wir werden sehen, dass wir das aktuelle Fach auch in unserem CDocument Objekt halten sollten.



Unsere Oberfläche mit dem Auswahlfeld für das Fach

Datenelement „Aktuelles Fach“

Wir ergänzen also am besten ein Datenelement _aktuellesFach in unserer CDocument Klasse. Es spricht im Moment nichts dagegen dieses Datenelement als CString anzulegen.

Erzeugen eines neuen Faches

Zum erzeugen eines neuen Faches können wir eine Technik anwenden, die immer häufiger in Windows Verwendung findet. Der Trick dabei ist, dass man im einen Auswahlfeld den speziellen Eintrag „Neu…“ einträgt. Bei uns könnte dieser lauten: „Neues Fach…“.
Natürlich können wir im Menu Bearbeiten auch einen solchen Menupunkt einfügen.

Füllen der Combobox

Die Combobox ist ein Datenelement des Dialogbars, der sich im CMainFrame Objekt befindet. Es sollte möglich sein, vom diesem Objekt aus, die nötigen Einträge zu machen.
Dabei können wir die nette Technik verwenden, mit der wir ein Betriebssystem-Fenster mit einem C++ Objekt verbinden um auf die Kapselung der MFC zurückgreifen zu können. Ergänze also unten in der OnCreate Methode von CMainFrame folgenden Code:

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
      if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
            return -1;

WEITER UNTEN:

    // mit GetDlgItem holen wir uns
    // einen CWnd Zeiger auf ein Objekt
    // in einem Dialog
    CWnd* comboboxWindow = m_wndDlgBar.GetDlgItem(IDC_FACH);
    if(comboboxWindow)
    {
        CComboBox combobox;
        // verbinden mit Windows Fenster
        combobox.Attach(comboboxWindow->m_hWnd);
        combobox.InsertString(-1, "Neues Fach...");
        // wieder trennen, denn der
        // Destruktor will sonst
        // das Windows Handle zerstören
        combobox.Detach();
    }

      return 0;
}
```

Neues Fach definieren

Wir sind eigentlich schon bald soweit, dass wir ein Fach erzeugen können. Ein neues Fach liesse sich über einen Dialog erzeugen.

So könnte ein Dialog zum Erstellen eines neuen Faches aussehen.

Erzeuge für diesen Dialog eine neue Klasse, die von CDialog abgeleitet ist. Nimm dazu die Hilfe des Assistenten in Anspruch, der durch Doppelklick auf den Dialog im Ressourcen-Editor aktiviert wird.
Diese Klasse könnte „CNeuesFachDialog“ heissen. Verbinde in dieser Klasse die ID’s der Editboxen mit Variablen vom Datentyp CString. Bei mir heissen die beiden _fachname und _lehrername. Da beide Datenelemente selbstverständlich privat sind, musst du auch gleich zwei Zugriffsmethoden definieren (können diese const sein?).
Zusätzlich können wir auch einen Menupunkt in das Bearbeiten Menu einbauen, mit dem wir diesen Dialog aufrufen können. Danach werden wir diesen Befehl auch ausführen, wenn der Benutzer in der Combobox im Hauptfenster den Punkt „Neues Fach…“ auswählt.
Den Befehl kann unsere Document-Klasse direkt behandeln.

```
void CNotenverwaltungDoc::OnBearbeitenNeuesfacherzeugen()
{
    CNeuesFachDialog dlg;
    if(IDOK == dlg.DoModal())
    {
        CString fachname = dlg.getFachname();
        // neues Fach nur, falls der
        // Name etwas enthält.
        if(0 != fachname.GetLength())
        {
            CString lehrername = dlg.getLehrername();
            _faecher.neuesFach(fachname, lehrername);
        }
    }
}
```

Das neue Fach erzeugen wir in der Kollektion CFaecher mit einer neuen Methode „neuesFach“.

Die Klasse CFaecher

Meine Klasse CFaecher sieht mittlerweile so aus:

```
const CFach& CFaecher::getFach(const CString& name) const
{
    Faecher::const_iterator it = _faecher.find(name);
    if(_faecher.end() == it)
    {
        // dieses Fach existiert nicht
        // wir werfen den Namen des
        // nicht gefundenen Faches als Excpetion
        throw name;
    }

    const CFach* fach = it->second;
    return *fach;
}
```

```
CFach& CFaecher::getFach(const CString& name)
{
    Faecher::iterator it = _faecher.find(name);
    if(_faecher.end() == it)
    {
        // dieses Fach existiert nicht
        // wir werfen den Namen des
        // nicht gefundenen Faches als Excpetion
        throw name;
    }

    CFach* fach = it->second;
    return *fach;
}

CFach& CFaecher::neuesFach(const CString& name,
                           const CString& lehrername)
{
    CFach* fach = 0;

    Faecher::iterator it = _faecher.find(name);
    if(_faecher.end() != it)
    {
        fach = it->second;
    }
    else
    {
        fach = new CFach(lehrername);
        _faecher[name] = fach;
    }
    return *fach;
}
```

CMainFrame Ergänzungen

Behandeln der aktuellen Fachauswahl

Das CMainFrame-Objekt beinhaltet ja den Dialogbar mit der Combobox. Es sollte möglich sein durch Abfangen der Änderungsnachricht das aktuelle Fach im Document zu setzen oder ein neues Fach erzeugen zu lassen.
Das CCombobox Objekt schickt uns die Nachricht CBN_SELCHANGE.

Beliebige Control Meldungen empfangen

Um Meldungen von einem beliebigen Control zu Empfangen, können wir folgenden Eintrag in die Message Map der Klasse CMainFrame machen:

```
    ON_CONTROL(CBN_SELCHANGE, IDC_FACH, OnSelectFach)
END_MESSAGE_MAP()
```

Das Makro um Control-Meldungen zu erhalten heisst also ON_CONTROL. Das erste Argument ist dann der Control Code, den wir empfangen wollen, das zweite die ID des Controls, das bei mir IDC_FACH heisst und das dritte eine Methode, die ich folgendermassen implementiert habe:

```
void CMainFrame::OnSelectFach()
{
    CWnd* comboboxWindow = m_wndDlgBar.GetDlgItem(IDC_FACH);
    if(comboboxWindow)
    {
        CComboBox combobox;
        // verbinden mit Windows Fenster
        combobox.Attach(comboboxWindow->m_hWnd);
        CString fachname;
        combobox.GetWindowText(fachname);
        if(fachname == neuesFachString)
        {
            CNotenverwaltungView* pView = GetRightPane();
            CNotenverwaltungDoc* doc = pView->GetDocument();
            doc->OnBearbeitenNeuesfacherzeugen();
        }
        // wieder trennen, denn der
        // Destruktor will sonst
        // das Windows Handle zerstören
        combobox.Detach();
    }
}
```

Vergiss nicht diese Methode in der Header-Datei zu deklarieren.

Ergänzungen an der Dokument-Klasse

Auch diese Klasse muss noch ergänzt werden, damit wir endlich neue Fächer erzeugen können, eines als aktuelles Fach setzen können und auch um die Combobox korrekt zu füllen.

Neues Datenelement „AktuellesFach“

Am besten ist es wir Ergänzen ein neues Datenelement vom Datentyp CString mit dem Namen „_aktuellesFach“. Dieses Datenelement soll gesetzt und abgefragt werden können. Auch soll es beim Serialisieren mit abgespeichert werden.

Neue Methode „fillFachArray“

Diese Methode dient dem CMainFrame Objekt um ein Array von CString-Objekte zu erhalten, in dem alle Fachnamen drin stehen. So kann die Combobox auf dem neuesten Stand gehalten werden.
Die Klasse CStringArray ist eine MFC Klasse, die gebraucht werden kann um CString als Array zu speichern. Da die CFach-Objekte in der CFaecher Klasse gespeichert werden, leiten wir diesen Aufruf direkt an diese Klasse weiter:

```
void CNotenverwaltungDoc::fillFachArray(CStringArray&
fachArray) const
{
    _faecher.fillFachArray(fachArray);
}
```

Und in der Klasse CFaecher:

```
void CFaecher::fillFachArray(CStringArray& fachArray) const
{
    Faecher::const_iterator it = _faecher.begin();
    Faecher::const_iterator end = _faecher.end();
    for( ; it != end; ++it)
    {
        fachArray.Add(it->first);
    }
}
```

Bei dieser Gelegenheit sollten wir auch gleich den Destruktor von CFaecher noch ausprogrammieren, das heisst wir müssen dort die mit new allozierten Daten wieder freigeben:

```
CFaecher::~CFaecher()
{
    Faecher::iterator it = _faecher.begin();
    Faecher::iterator end = _faecher.end();
    for( ; it != end; ++it)
    {
        CFach* fach = it->second;
        delete fach;
    }
}
```

Zurück zum Füllen den Combobox

Nun endlich sollten wir in der Lage sein die Combobox aktuell zu halten.

Methode CMainFrame::OnSelectFach

```
void CMainFrame::OnSelectFach()
{
    CWnd* comboboxWindow = m_wndDlgBar.GetDlgItem(IDC_FACH);
    if(comboboxWindow)
    {
          CNotenverwaltungView* pView = GetRightPane();
        CNotenverwaltungDoc* doc = pView->GetDocument();

        CComboBox combobox;
        // verbinden mit Windows Fenster
        combobox.Attach(comboboxWindow->m_hWnd);
        CString fachname;
        combobox.GetWindowText(fachname);
        if(fachname == neuesFachString)
        {
            doc->OnBearbeitenNeuesfacherzeugen();
            CStringArray fachnamen;
            doc->fillFachArray(fachnamen);
            // Combobox leeren
            combobox.ResetContent();
            for(int i = 0; i < fachnamen.GetSize(); ++i)
            {
                combobox.InsertString(-1, fachnamen[i]);
            }
            combobox.InsertString(-1, neuesFachString);
        }
        else
        {
            doc->setAktuellesFach(fachname);
        }
        // wieder trennen, denn der
        // Destruktor will sonst
        // das Windows Handle zerstören
        combobox.Detach();
    }
}
```