Einführung in die Windows - Programmierung mit der MFC

Ziel, Inhalt

- Wir erzeugen heute ein erstes MFC-Programm mit dem Assistenten des Visual Studios. Wir werden sehen, wie leicht es ist ein MFC – Programm zu erzeugen.

- Wir werden sehen, wie man ein MFC - Programm erweitert und warum die MFC ein so genanntes Framework ist.

Die MFC als Framework

Framework Klassen

Framework Klassen sind Klassen, die einen Rahmen für eine Applikation oder für bestimmte Problembereiche (Problemdomänen) bilden. Man kann solche Frameworks kaufen oder man erhält sie wie die MFC oder die VCL von Borland zusammen mit einer Entwicklungsumgebung.

Aufbau von Frameworks

Die Grundidee bei einem Framework oder von Framework Klassen ist, dass der Programmierer nicht Objekte dieser Klassen erzeugt und Aufrufe macht, sondern, dass gewisse Framework Objekte die Methoden aufrufen, die der Programmierer bereitstellen muss oder kann. Nicht der Code, der vom Programmierer geschrieben wird bestimmt den Programmablauf sondern die Framework Klassen.
Es ist für Programmierer und deren Manager manchmal schwierig zu verstehen, dass die Idee von „Reuse“ (Wiederverwendung von Code) nicht ist, dass man kleine Codestücke und Klassen wieder verwendet, sondern dass bei einem Framework ein grosser Teil vom Code bereits besteht und der Programmierer kleine Stücke Code liefern muss, um das Framework für sich arbeiten zu lassen. Im allgemeinen ist der Weg, der beschritten wird, dass der Programmierer von einer Framework Klasse ableitet und gewisse virtuelle Methoden überschreibt.

Aufbau der MFC

Die MFC bietet also Klassen an, die den Rahmen für ein Windows Programm bilden. Die Programmierer bei Microsoft haben die Schritte, die nötig sind um Programme mit Fenstern zu erzeugen genau analysiert und die immer wiederkehrenden Schritte in die Microsoft Foundation Classes (MFC) gepackt. Betrachte den Code, den wir geschrieben haben um ein einfaches Windows Programm zu erzeugen.
http://www.devmentor.ch/teaching/additional/00I/Semester5/index.html
12. Abend und 13. Abend zeigen, dass ein Windows Programm immer aus einer Nachrichtenschleife besteht. Um Fenster zu erzeugen muss man immer eine Fensterklasse erzeugen und Nachrichtenbearbeitungsfunktionen bereitstellen. Dieser Code, der immer gleich aussieht verschwindet nun in der MFC.

Document-View Architekur in der MFC

Die MFC wendet zur Abstraktion der Windows Oberfläche einige Patterns an. Eines davon kennen wir bereits, nämlich das Observer Pattern (siehe auch 10./11. Abend). Die MFC bietet an Document-Klassen zu verwenden, wo man die Daten verwaltet und View-Klassen, mit denen man die Daten dem Benutzer darstellt und auch Änderungen daran vornehmen kann. Dabei sind

die View-Klassen die Observer und die Document-Klassen sind als Subject-Klassen gedacht.

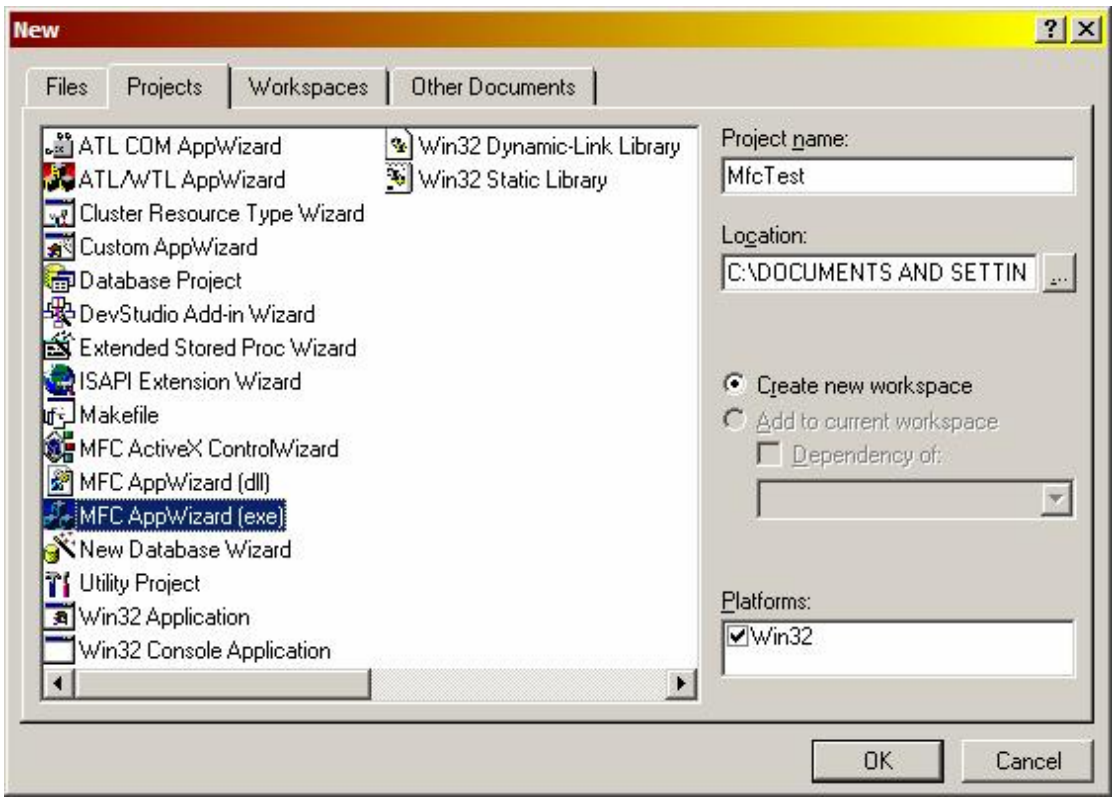
Erzeugen einer MFC Applikation

Einfache Applikation

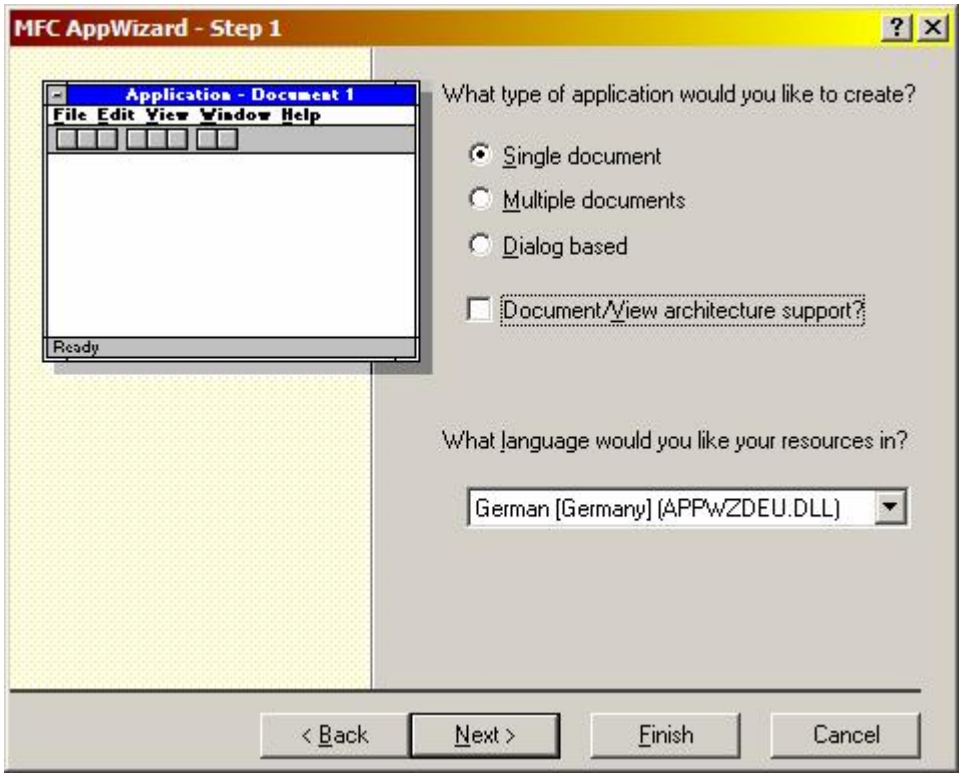
Wir erzeugen als erstes eine ganz einfache Applikation ohne Unterstützung für Document-View Architektur. Trotzdem werden wir sehen, wie die MFC zusammen mit dem Assistenten Applikationen erzeugt. **Wichtig:** Achte darauf, dass du das Service Pack 5 für das Visual Studio installiert hast. Andernfalls wird das vom Assistenten erzeugte Programm nicht funktionieren.

Der MFC-Applikations Assistent

Erzeuge ein neues Projekt, wähle den MFC Applikationsassistenten und gib dem Projekt einen tollen Namen.



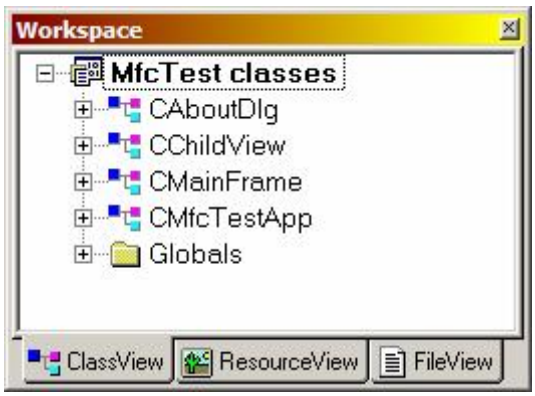
Beim nächsten Schritt wähle ein Applikation, die nur ein Document und nur eine View erzeugt und stelle den Support für Document-View Architektur ab.



Eigentlich kannst du an dieser Stelle bereits „Finish“ oder Beenden drücken, die Applikation, die jetzt so erstellt wird genügt unseren Ansprüchen.

Die erzeugten Klassen

Der Assistent erzeugt nun einige Klassen, die von MFC-Framework Klassen abgeleitet sind.



Die Applikationsklasse

Die Klasse CMfcTestApp Klasse, dient als Abstraktion unserer Applikation. Sie ist von der MFC Klasse CWinApp abgeleitet. Diese Framework Klasse kümmert sich um das Aufstarten und Beenden der Applikation sowie um die Nachrichtenschleife, die so vor uns verborgen bleibt. Der Assistent erzeugt die Klasse und überschreibt für den Anwender die Methode InitInstance. In dieser Methode wird das Hauptfenster erzeugt und dargestellt. Danach übernimmt die Basisklasse wieder und startet die Nachrichtenschleife.

Das Rahmenfenster

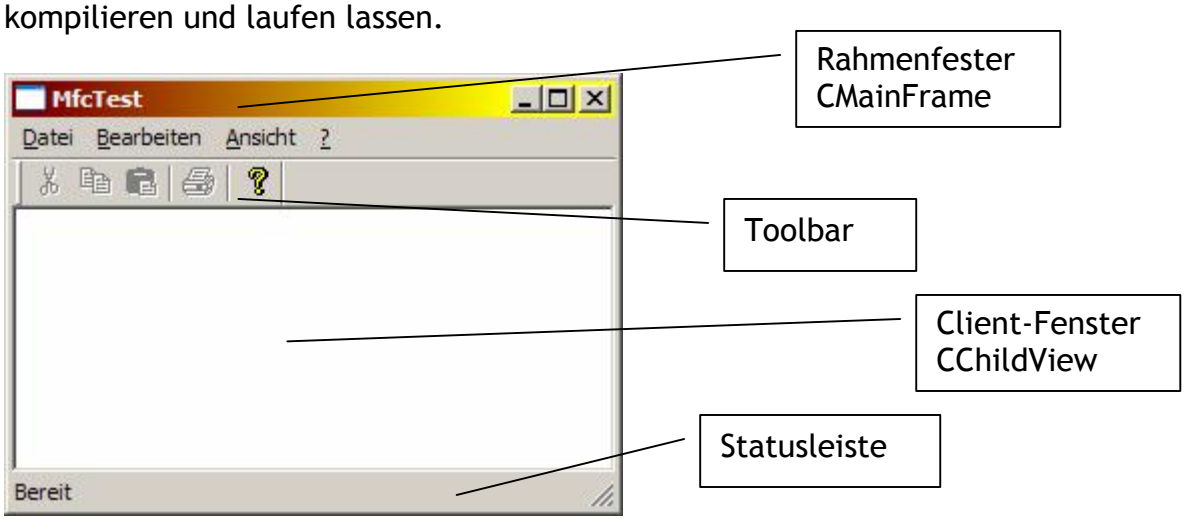
Die Klasse CMainFrame ist die Klasse, die von der Klasse CFrameWnd ableitet. Dieses Objekt kümmert sich um das Rahmenfenster. Dazu gehört das Menu, die Toolbar und der Statusbar unten am Fenster. Es erzeugt auch ein Objekt der Klasse CChildView. Die Toolbar ist das Objekt m_wndToolBar vom Datentypen CToolBar. Die Statusbar ist das Objekt m_wndStatusBar vom Datentypen CStatusBar.

Das Client-Fenster

Der Bereich im Innern eines Fensters wird auch als Client-Bereich bezeichnet. Dieser Client Bereich wird bei der MFC durch ein Objekt der Klasse CWnd behandelt. Der Assistent erzeugt die Klasse CChildView als Ableitung davon, so dass der Programmierer Einfluss auf die Darstellung des Fensters hat. Dieses Fenster nimmt tatsächlich nur den Bereich im innern des Rahmenfensters, also die weisse Fläche.

Das Programm

Ohne eine einzige Zeile Code zu schreiben, können wir das Projekt bereits kompilieren und laufen lassen.

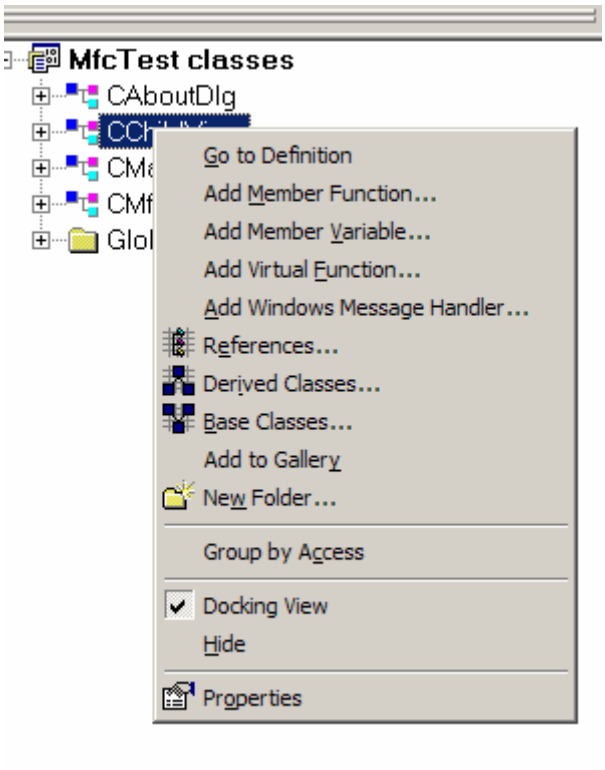


Der Assistent hat wo nötig einige Methoden der Framework Klassen überschrieben, so dass wir nun direkt die Applikation nach unseren Wünschen anpassen können.

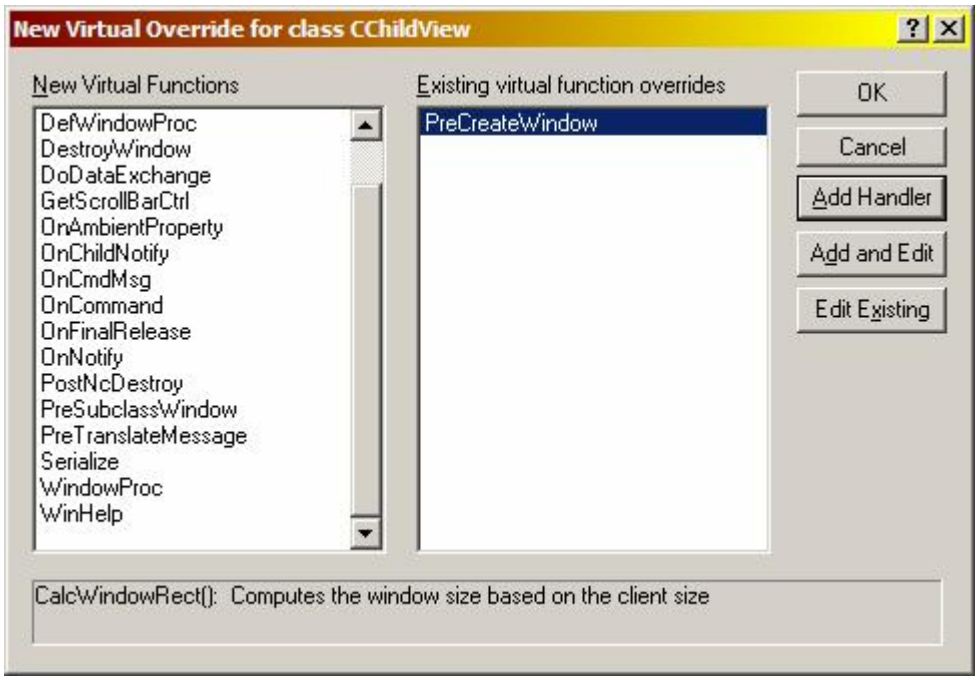
Erweitern der Applikation

Der Assistent

Der Assistent erzeugt wie gesagt einige Methoden, die einige der Basisklassen-Methoden überschreiben. Das sind vor allem Methoden, die wie InitInstance den Ablauf des Programmes bestimmen. Da aber Windows ein Ereignisbasiertes Betriebssystem ist, gibt es auch die Möglichkeit Meldungen, die an das Programm gesendet werden von einigen der Klassen abzufangen und zu bearbeiten.
Dabei hilft uns der Assistent auch nach dem Erzeugen des Programms. Je nach Klasse, die man erweitern will kann einem der Assistent verschiedene Möglichkeiten anbieten, die – seiner Meinung nach – Sinn machen. Durch Rechtsklick auf eine Klasse in der Klassenansicht erscheint ein Kontext-Menu mit der Möglichkeit virtuelle Methoden oder Methoden zum behandeln von Ereignissen einzufügen.



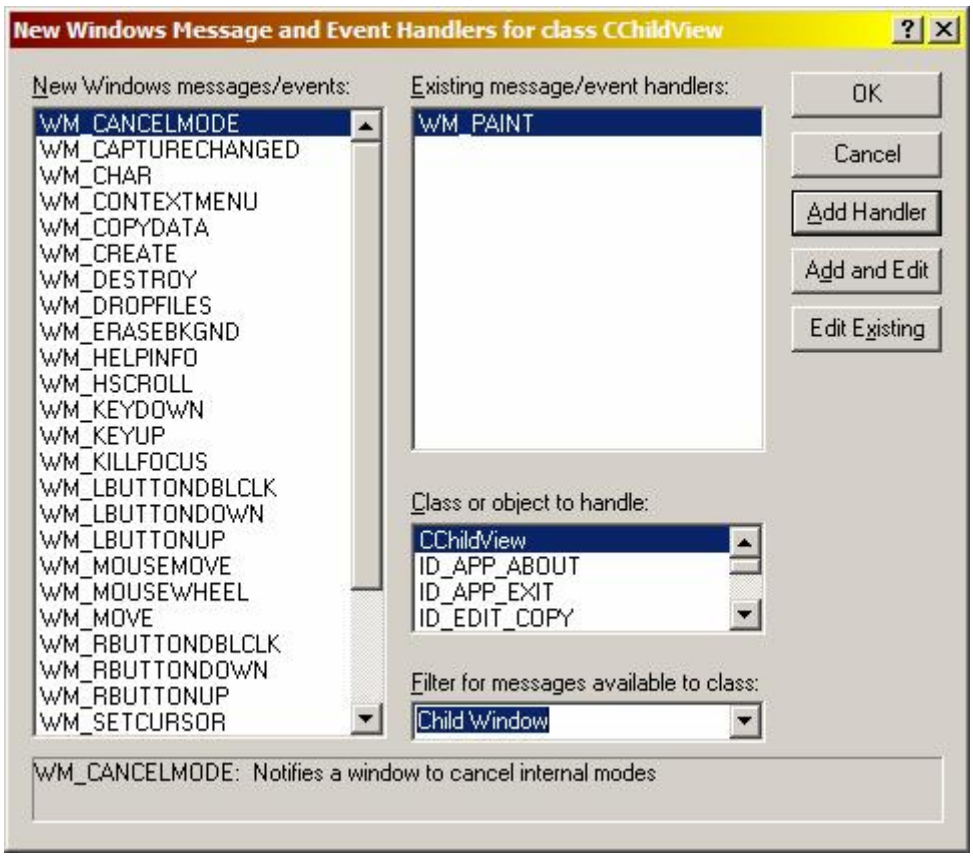
Virtuelle Methoden überschreiben



Bei den virtuellen Methoden bei der Klasse CChildView sehen wir, dass PreCreateWindow bereits überschrieben ist. Diese Methode wird beispielsweise aufgerufen bevor das Fenster erzeugt. Das Framework gibt die Möglichkeit im Entstehungsprozess des Fensters einzugreifen.

Ereignisbehandlung

Interessanter ist für uns die Möglichkeit auf besondere Ereignisse des Betriebssystems zu reagieren. Dazu gehören Mausereignisse, Tastenereignisse und ähnliches.
Eine Methode, die in der Klasse CChildView bereits vorhanden ist, ist die OnPaint Methode. Diese Methode wird aufgerufen sobald das Fenster gezeichnet werden muss was aufgrund der WM_PAINT Nachricht geschieht.



Das ist der Dialog, der erscheint wenn man einen Ereignis behandeln will. Unten rechts besteht die Möglichkeit eine Gruppe von Ereignissen auszuwählen. Da das CChildView ein Kindfenster ist, ist die Wahl von Child Window richtig. Die Listbox darüber erlaubt die Auswahl von einem Objekt, das man behandeln will. Links erscheint eine Auswahl der möglichen Windows-Ereignisse. Es ist leider nur die Auswahl, für die der Assistent gerüstet ist. Viele weitere Ereignisse kann man nicht mit dem Assistenten behandeln.

Was der Assistent erzeugt

Als Beispiel betrachten wir die WM_MOUSEMOVE Nachricht. Der Assistent erzeugt in der Header-Datei die Methodendeklaration:

```
afx_msg void OnMouseMove(UINT nFlags, CPoint point);
```

Diese Deklaration befindet in einem Block, der vom Assistenten verwaltet wird und im Editor meistens eine andere Farbe hat. In der Quellcodedatei erscheinen an zwei Orten neuer Code. Bei Klassen, die Ereignisse behandeln können gibt es eine so genannte Message Map. In der Quellcode Datei finden wir einerseits diesen Eintrag:

```
BEGIN_MESSAGE_MAP(CChildView,CWnd )
      //{{AFX_MSG_MAP(CChildView)
      ON_WM_PAINT()
      ON_WM_MOUSEMOVE()
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

Andererseits hat der Assistent folgenden Code erzeugt

```
void CChildView::OnMouseMove(UINT nFlags, CPoint point)
{
      // TODO: Add your message handler code here
      // and/or call default

      CWnd::OnMouseMove(nFlags, point);
}
```

Hier kann man nun den eigenen Code einfügen.
Was die einzelnen Parameter bedeuten kann man mit der Hilfe herausfinden. Auch wie die Ereignisse (WM_XXX) behandelt werden und wann diese gesendet werden findet man mit der Hilfe heraus. Je besser man Englisch versteht um so mehr nützt die Hilfe.

Einfügen von neuen Methoden und Datenelementen

Der Assistent hilft auch neue Methoden und Datenelemente in die Klassen einzufügen. Auch diese Funktion ist über Rechtsklick in der Klassenansicht erreichbar. Der Assistent fügt automatisch die notwendigen Deklarationen und Definitionen ein.
Selbstverständlich kann man die Klassen weiterhin von Hand in der Header- und in der Quellcodedatei anpassen.