

Treeview und das Tree Control

Ziel, Inhalt

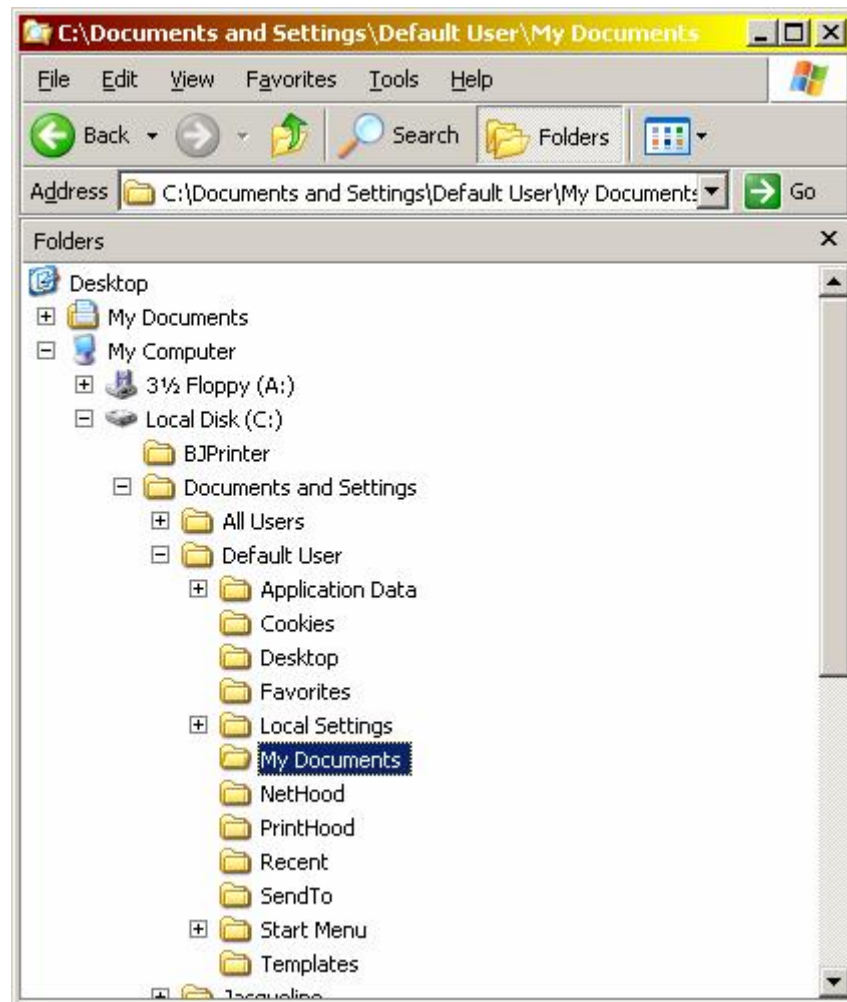
- § Wir lernen heute anhand eines Beispiels das Tree Control kennen und anwenden.

Treeview und das Tree Control	1
Ziel, Inhalt	1
Treeview und das Tree Control	2
Das Tree Control	2
Projekt mit Tree Control	2
Erzeugen des Projektes	3
Projektstruktur	4
Die Klasse CTreeView	5
Klassenhierarchie	5
Die Klasse CTreeCtrl	6
Einfügen eines Elementes in den Tree	6
Das HTREEITEM	6
Kleines Beispiel mit Hierarchie	6
Setzen der Eigenschaften des Tree Controls	7
Die Nachrichten des Tree Controls	7

Treeview und das Tree Control

Das Tree Control

Das Tree Control dient als Ansicht für Baumartige Strukturen wie zum Beispiel dem Dateibaum.



Tree Control zur Ansicht der Dateistruktur

Das Tree Control wird in der MFC von der Klasse `CTreeCtrl` gekapselt. Um die Eigenschaften des Tree Controls kennen zu lernen, erzeugen wir ein Projekt, das davon Gebrauch macht.

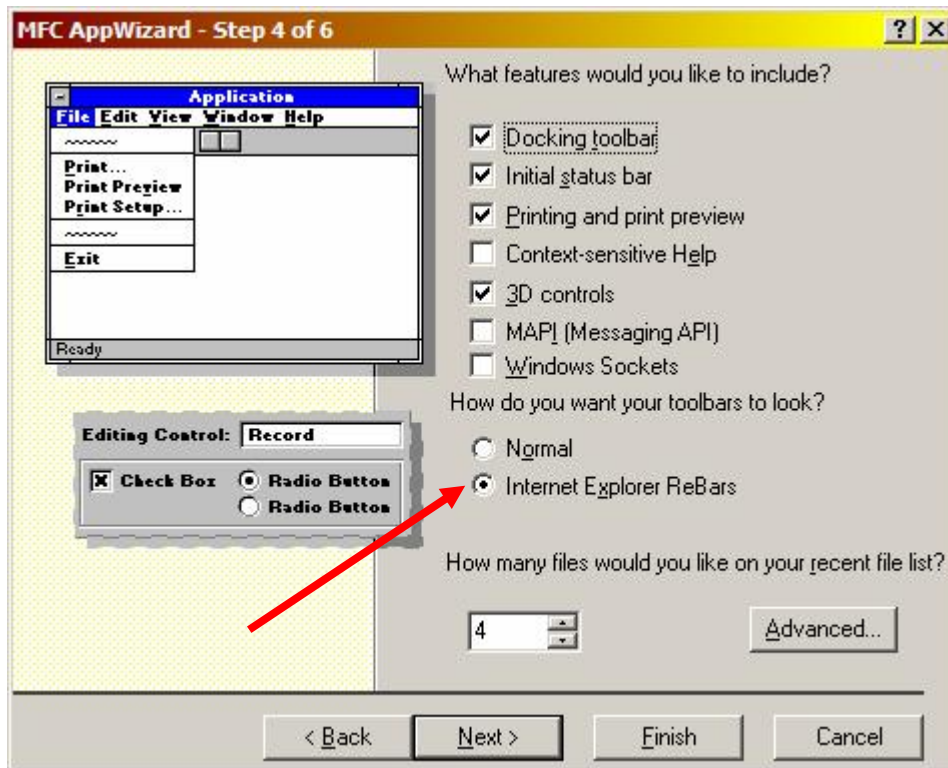
Projekt mit Tree Control

Ein Tree Control kann in einem Dialog verwendet werden, oder wir können ein Projekt erstellen, das ähnlich wie der Explorer auf der linken Seite ein Tree Control verwendet.

Erzeugen des Projektes

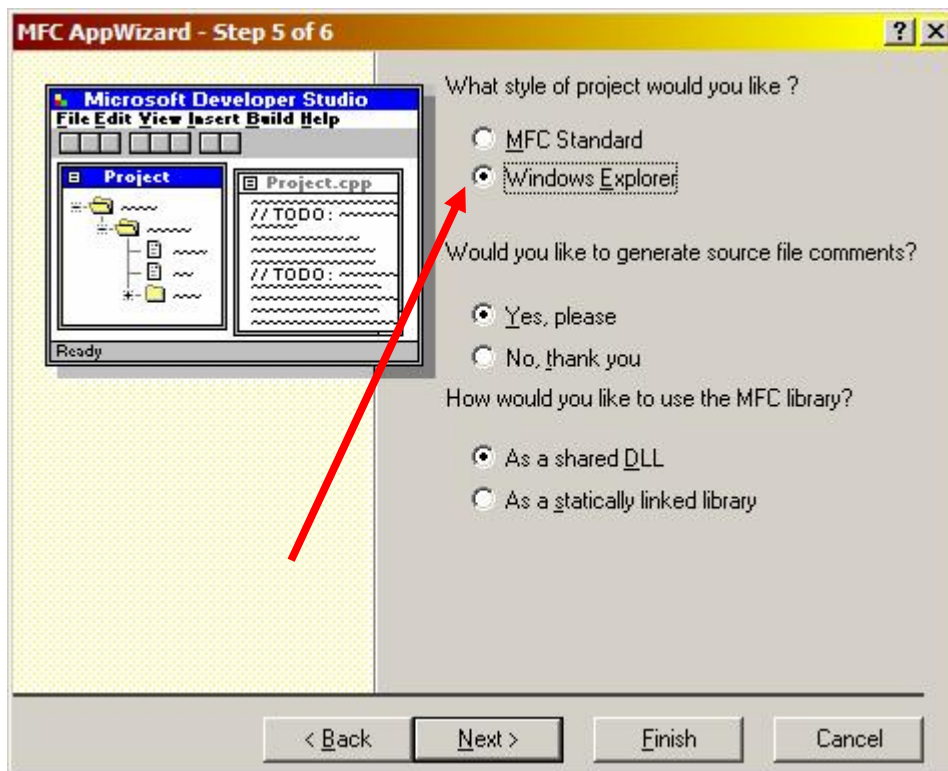
Um das Projekt zu erzeugen gehen wir wie sonst vor und erzeugen eine MFC-Anwendung.

Erzeuge ein SDI-Projekt (Single Document Interface). Und verwende die Einstellungen, die der Assistent vorschlägt. Beim Schritt vier kannst du für die bessere Optik die Option „Internet Explorer ReBars“ wählen:



Auswahl der Internet Explorer ReBars

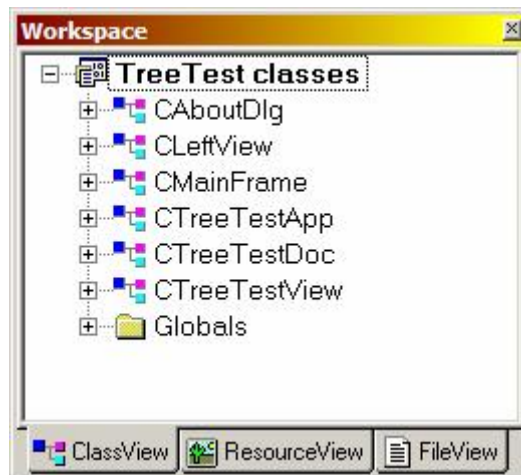
Auf der nächsten Seite ist es wichtig die Art des Projektes auf „Windows Explorer“ zu setzen.



Auswahl der Projektart „Windows Explorer“

Projektstruktur

Das Projekt enthält zusätzlich zu den bereits bekannten erzeugten Klassen die Klasse CLeftView. Diese Klasse ist von CTreeView abgeleitet.



Projektstruktur mit zusätzlicher Klasse CLeftView

Die Klasse CTreeView

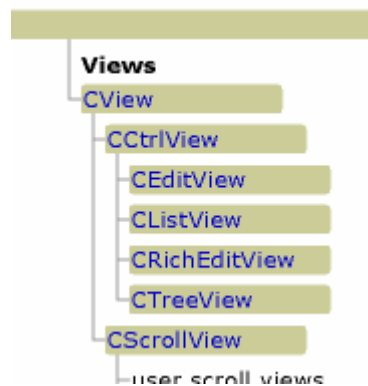
Die Klasse CLeftView ist von der Klasse CTreeView abgeleitet.

Diese Klasse ist einerseits eine normale View-Klasse, die in der MFC üblicherweise als Observer der CDocument Klasse dient. Um die Funktionalität eines TreeView Controls bereit zu stellen enthält sie ein TreeControl. Hier ein Ausschnitt aus der Deklaration der Klasse CLeftView:

```
class CLeftView : public CTreeView
{
protected: // create from serialization only
    CLeftView();
    DECLARE_DYNCREATE(CLeftView)
}
```

Klassenhierarchie

Die Klasse CTreeView ist von der Klasse CControlView abgeleitet, die ihrerseits von der Klasse CView abgeleitet ist.



Ausschnitt aus der Klassenhierarchie der MFC

Als einzige besondere Methode enthält die Klasse CTreeView die Methode GetTreeCtrl(). Damit greifen wir auf das eingebettete CTreeCtrl Objekt zu. Beachte, dass die Methode eine Referenz auf das CTreeCtrl-Objekt zurückgibt:

```
CTreeCtrl& treeControl = GetTreeCtrl();
```

Die Klasse CTreeCtrl

Hier beginnt der Spass mit dem Tree Control.

Einfügen eines Elementes in den Tree

Die Klasse CTreeCtrl hat mehrere Methoden, um Elemente einzufügen. Die einfachste ist folgende:

```
void CLeftView::OnInitialUpdate()
{
    CTreeView::OnInitialUpdate();

    // TODO: You may populate your TreeView with items by
    // directly accessing
    // its tree control through a call to GetTreeCtrl().
    CTreeCtrl& treeControl = GetTreeCtrl();

    treeControl.InsertItem("Baum", TVI_ROOT, TVI_LAST);
}
```

Diese Methode InsertItem nimmt als Argument einen string-Zeiger (const char*), als zweites ein HTREEITEM, das als Elternelement des neuen betrachtet wird und als letztes eine vordefinierte Konstante um die Position unter den bereits eingefügten Kindelementen zu bestimmen. Die Methode hat als Rückgabewert, der hier nicht verwendet wurde ein HTREEITEM.

Das HTREEITEM

Dieser Windows-Datentyp ist ein HANDLE auf ein Element im Baum.

Kleines Beispiel mit Hierarchie

Hier ein kleines Beispiel, das eine Hierarchie bildet.

```
void CLeftView::OnInitialUpdate()
{
    CTreeView::OnInitialUpdate();

    CTreeCtrl& treeControl = GetTreeCtrl();

    HTREEITEM baum = treeControl.InsertItem("Baum",
                                           TVI_ROOT,
                                           TVI_LAST);
    treeControl.InsertItem("Kind 1", baum, TVI_FIRST);
    treeControl.InsertItem("Kind 2", baum, TVI_LAST);
    HTREEITEM liebling =
        treeControl.InsertItem("Lieblingskind",
                               baum,
                               TVI_FIRST);
    treeControl.InsertItem("KindesKind", liebling, TVI_LAST);
}
```

Beim Testen bemerkst du sicher, dass das Tree Control zwar alle Elemente enthält, aber dieser Umstand nicht erkennbar ist. Es fehlen die Pluszeichen, die wir uns vom Windows-Explorer gewohnt sind.

Setzen der Eigenschaften des Tree Controls

Das Tree Control kennt spezielle Window Styles. Window Styles kennen wir seit wir das erste Fenster erzeugt haben. Die Window Stile, die wir dort angewendet haben, waren Konstanten, die mit WS_ beginnen. Beim List Control haben wir zusätzlich Control spezifische Stile kennen gelernt. Auch das Tree Control kennt seine besonderen Window Stile, die wir in der Dokumentation finden können. Der Stil, der Plus- und Minusboxen erzeugt heisst TVS_HASBUTTONS.

Um den Window Stil zu ändern bei einem bestehenden Fenster zu ändern, verwenden wir die Methoden GetWindowLong und SetWindowLong folgendermassen:

```
CTreeCtrl& treeControl = GetTreeCtrl();
LONG style = GetWindowLong(treeControl.m_hWnd, GWL_STYLE);
style |= TVS_HASBUTTONS | TVS_LINESATROOT;
SetWindowLong(treeControl.m_hWnd, GWL_STYLE, style);
```

Die Nachrichten des Tree Controls

Das Tree Control versendet einige Nachrichten, die nur für ein Tree Control relevant sind. der Assistent gibt uns die Möglichkeit, diese Nachrichten in der Klasse CLeftView (CTreeView) zu behandeln. Hier zum Beispiel die Nachricht TVN_SELCHANGED. Diese Nachricht wird gesandt, wenn ein Element angewählt wurde.

```
void CLeftView::OnTvnSelchanged(NMHDR *pNMHDR,
                                LRESULT *pResult)
{
    LPNMTREEVIEW pNMTreeView =
        reinterpret_cast<LPNMTREEVIEW>(pNMHDR);
    // TODO: Add your control notification handler code here
    *pResult = 0;
}
```

Diese Nachrichten werden in der Form einer WM_NOTIFY Nachricht versandt. Die MFC setzt diese Nachrichten mit dem ON_NOTIFY oder ON_NOTIFY_REFLECT Makro in der Message Map um.