Hochschule Harz	FB Automatisierung und Informatik
MFC	Grafische Nutzerschnittstellen mit MFC Thema: SDI-Programmierung mit TreeView

Versuchsziele

Vertiefung im Verständnis der SDI-Programmierung mit der Baum-Komponente TreeView unter Visual Studio 2008.

Überblick:

Diese Übung zeigt die Verwendung des „Trees“, TreeView in einem SDI-Fenster. Bekannt ist dieses GUI im linken Teil des Explorers.

Folgende Aufgaben müssen Sie durchführen:

- Erstellen eines SDI-Projektes mit MFC und einem TreeViewView (step1)
- Einstellen des Darstellungsmodus der TreeView, inklusive der Symbole
- Beispielcode in den Tree eintragen
- Erweitern:
  - Definieren eine Klasse zur Verwaltung der „Treedaten“, Nodes
  - Dialogfenster zum Einfügen, Ändern und löschen im Tree entwickeln

Grundlagen TreeView:

Die Komponente „TreeView“ erlaubt die Darstellung von Objekten in einer Baumstruktur. Bekannt durch die linke Seite im Explorer. Für n-Stufen benötigt man n-Nodes. Als rekusive Lösung ist es wiederum sehr einfach. Dazu gibt es das zweite Labor der Wahlpflichtveranstaltung MFC und .net.

Methoden für das Einfügen in den CTreeView

Konstanten:

- TVI_FIRST Einfügen des Knotens am Anfang der Liste
- TVI_LAST Einfügen des Knotens am Ende der Liste
- TVI_SORT Sortiertes Einfügen des Knotens nach des Alphabetischen Größe

1. Version:

```
HTREEITEM InsertItem(  LPCTSTR lpszItem,
                       HTREEITEM hParent = TVI_ROOT,
                       HTREEITEM hInsertAfter = TVI_LAST
);
```

Beispiele:

```
HTREEITEM h1 = GetTreeCtrl().InsertItem("1. Knoten", TVI_ROOT, TVI_LAST);
   HTREEITEM h2 = GetTreeCtrl().InsertItem("2. Knoten");  // ans Root, aber nachher
   HTREEITEM h3 = GetTreeCtrl().InsertItem("3. Knoten", TVI_ROOT, TVI_FIRST);
```

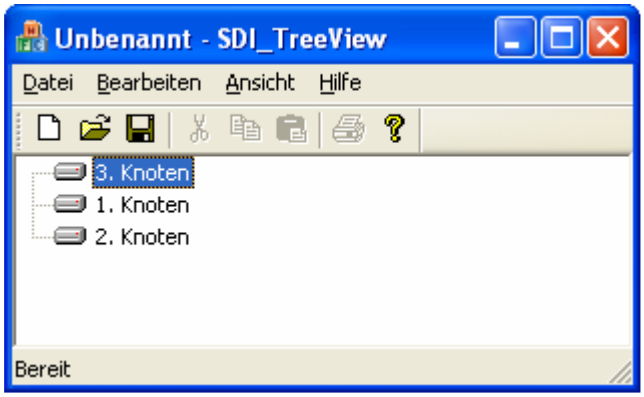


Abbildung 1 Ergebnis von bsp1

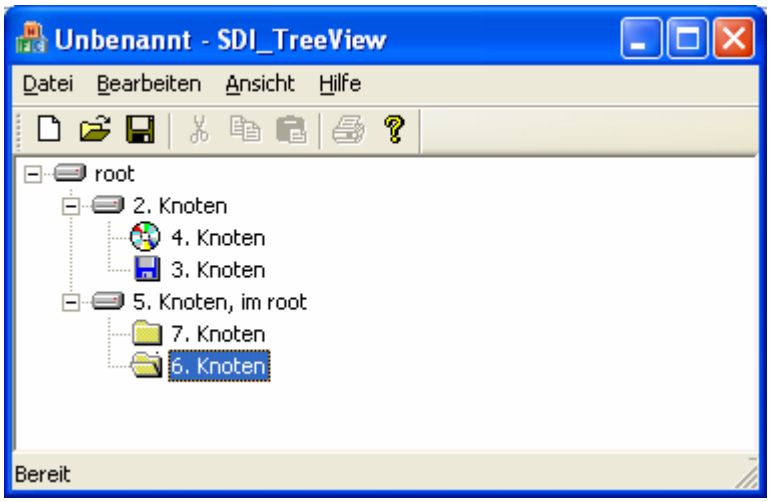
2. Version:

```
HTREEITEM InsertItem(     LPCTSTR lpszItem,
                          int nImage,
                          int nSelectedImage,
                          HTREEITEM hParent = TVI_ROOT,
                          HTREEITEM hInsertAfter = TVI_LAST
);
```

Diese Variante erlaubt die Definition der Bildsymbole. Dazu muss aber eine Bitmap in das Projekt eingefügt werden. Die Pixelhöhe beträgt 15 Pixel, die Höhe beträgt 16 Pixel. Der Parameter „parent“ dient als Anker beim Einfügen. Ohne eine Angabe, werden die Knoten immer ins root eingetragen. Ein Baum entsteht aber so nicht.

Beispiel:

```
void CSDI_TreeViewView::bsp2() {
   HTREEITEM root = GetTreeCtrl().InsertItem("root", 0,0, TVI_ROOT, TVI_LAST);
   HTREEITEM h2 = GetTreeCtrl().InsertItem("2. Knoten",root);
   HTREEITEM h3 = GetTreeCtrl().InsertItem("3. Knoten", 1,2,h2, TVI_FIRST);
   HTREEITEM h4 = GetTreeCtrl().InsertItem("4. Knoten", 2,3,h2, TVI_FIRST);
   HTREEITEM h5 = GetTreeCtrl().InsertItem("5. Knoten, im root",root);
   HTREEITEM h6 = GetTreeCtrl().InsertItem("6. Knoten", 4,5,h5, TVI_FIRST);
   HTREEITEM h7 = GetTreeCtrl().InsertItem("7. Knoten", 4,5,h5, TVI_FIRST); }
```



Das obige Beispiel zeigt die Verwendung des Parameters „parent“. Mit diesem kann man einen Baum komplett aufbauen. Der Knoten 7 wurde zwar als letztes eingetragen, aber die Konstante „TVI_FIRST“ verhindert dieses. Mit TVI_LAST würde der Eintrag an Ende eingetragen.

3. Version:

```
HTREEITEM InsertItem(
  UINT nMask,                    // Integer-Attribut
  LPCTSTR lpszItem,              // Text im Baum
  int nImage,                    // Bildindex, wenn nicht angeklickt
  int nSelectedImage,            // Bildindex, wenn angeklickt
  UINT nState,                   // setzt den Status, siehe unten
  UINT nStateMask,
  LPARAM lParam,
  HTREEITEM hParent,
  HTREEITEM hInsertAfter
);
```

Konstanten:

nMask:

- TVIF_CHILDREN The cChildren member is valid.
- TVIF_HANDLE The hItem member is valid.
- TVIF_IMAGE The iImage member is valid.
- TVIF_PARAM The lParam member is valid.
- TVIF_SELECTEDIMAGE The iSelectedImage member is valid.
- TVIF_STATE The state and stateMask members are valid.
- TVIF_TEXT The pszText and cchTextMax members are valid

nState:

Konstante	**Beschreibung**
TVIS_BOLD	The item is bold
TVIS_CUT	The item is selected as part of a cut-and-paste operation
TVIS_DROPHILITED	The item is selected as a drag-and-drop target
TVIS_EXPANDED	The item's list of child items is currently expanded; that is, the child items are visible. This value applies only to parent items
TVIS_EXPANDEDONCE	The item's list of child items has been expanded at least once. The TVN_ITEMEXPANDING and TVN_ITEMEXPANDED notification messages are not generated for parent items that have this state set in response to a TVM_EXPAND message. Using TVE_COLLAPSE and TVE_COLLAPSERESET with TVM_EXPAND will cause this state to be reset. This value applies only to parent items.
TVIS_EXPANDPARTIAL	A partially expanded tree view item. In this state, some, but not all, of the child items are visible and the parent item's plus symbol is displayed
TVIS_SELECTED	The item is selected. Its appearance depends on whether it has the focus. The item will be drawn using the system colors for selection

nStateMask:
Zeigt die Bits, die im Parameter state gültig sind.
Mögliche Werte:

- TVIS_OVERLAYMASK
- TVIS_STATEIMAGEMASK

hParent:
Handle of the inserted item’s parent.

hInsertAfter:
Handle of the item after which the new item is to be inserted.

Beispiel:

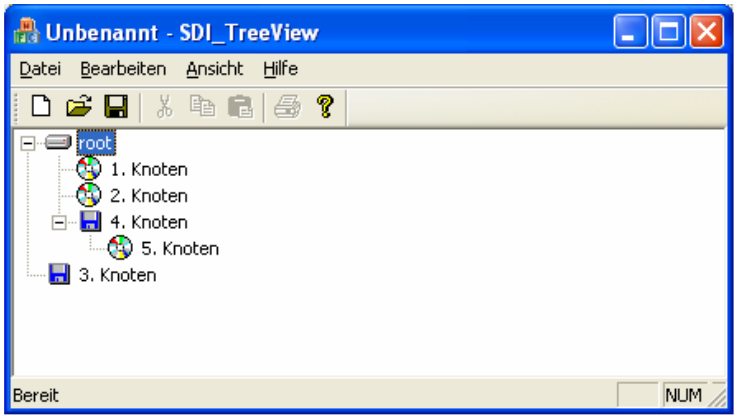


Abbildung 2 Beispiel3

```
void CSDI_TreeViewView::bsp3() {
   HTREEITEM root = GetTreeCtrl().InsertItem("root", 0,0, TVI_ROOT, TVI_LAST);
   HTREEITEM h1 = GetTreeCtrl().InsertItem(
          TVIF_IMAGE | TVIF_SELECTEDIMAGE| TVIF_TEXT,
          "1. Knoten",
          2, 3, 0,0,  NULL,root, NULL);
   HTREEITEM h2 = GetTreeCtrl().InsertItem(
          TVIF_IMAGE | TVIF_SELECTEDIMAGE| TVIF_TEXT,
          "2. Knoten",
          2, 3, 0,0,  NULL,root, NULL);
   HTREEITEM h3 = GetTreeCtrl().InsertItem(
          TVIF_IMAGE | TVIF_SELECTEDIMAGE| TVIF_TEXT | TVIF_HANDLE,
          "3. Knoten",
          1, 4, TVIS_BOLD, TVIS_STATEIMAGEMASK,   NULL,NULL, h2);
   HTREEITEM h4 = GetTreeCtrl().InsertItem(
          TVIF_IMAGE | TVIF_SELECTEDIMAGE| TVIF_TEXT | TVIF_HANDLE,
          "4. Knoten",
          1, 4, 0,0,  NULL,root,NULL);
   HTREEITEM h5 = GetTreeCtrl().InsertItem(
          TVIF_IMAGE | TVIF_SELECTEDIMAGE| TVIF_TEXT,
          "5. Knoten",
          2, 2, 0,0,  NULL,h4,NULL);
}
```

Die Bitmuster bzgl. der Bilder funktionieren. Das Muster für Texte ist nicht notwendig.

4. Version:

Diese Methode benutzt eine verschachtelte Struktur mit den Einträgen

- HTREEITEM hParent;
- HTREEITEM hInsertAfter;
- TVITEM item;

Struktur:

```
   typedef struct tagTVINSERTSTRUCT {
      HTREEITEM hParent;
      HTREEITEM hInsertAfter;
   #if (_WIN32_IE >= 0x0400)
      union
      {
         TVITEMEX itemex;
         TVITEM item;
      } DUMMYUNIONNAME;
   #else
      TVITEM item;
   #endif
   } TVINSERTSTRUCT, FAR *LPTVINSERTSTRUCT;
```

TVITEM hat folgende Attribute:

- mask = TVIF_TEXT;
- pszText // Knotenname = _T("Eintrag");
- iImage // Index des Bildsymbols
- iSelectedImage // Index des Bildsymbols, wenn angeklickt
- state // Status
- stateMask
- lParam // long pointer zu einer Struktur, einem Objekt
- cChildren // Flag that indicates whether the item has associated child items.
  // This member can be one of the following values.
  - Zero The item has no child items.
  - One The item has one or more child items.
  - I_CHILDRENCALLBACK
- cchTextMax // Size of the buffer pointed to by the pszText member, in characters.
  // If this structure is being used to set item attributes, this member is ignored.
- hItem // handle to the item

Methode:

```
HTREEITEM InsertItem(  LPTVINSERTSTRUCT lpInsertStruct );
```

Der Vorteil liegt in der Vordefinition der Struktur. Nur die notwendigen Daten werden geändert (Text, Symbole).

Beispiel:

```
TVINSERTSTRUCT tvInsert;
tvInsert.hParent = parent;
tvInsert.hInsertAfter = NULL;
tvInsert.item.mask = TVIF_TEXT;
tvInsert.item.pszText = _T("Ein Knoten");
tvInsert.item.iImage=1;
tvInsert.item.iSelectedImage=11;
tvInsert.item.state
tvInsert.item.stateMask
tvInsert.item.lParam=NULL;     // oder ein Pointer auf eine Klasse resp. Struktur
```

Aufgaben:

1) Grundprojekt erstellen:

Erstellen eines SDI-Projektes mit CTreeView

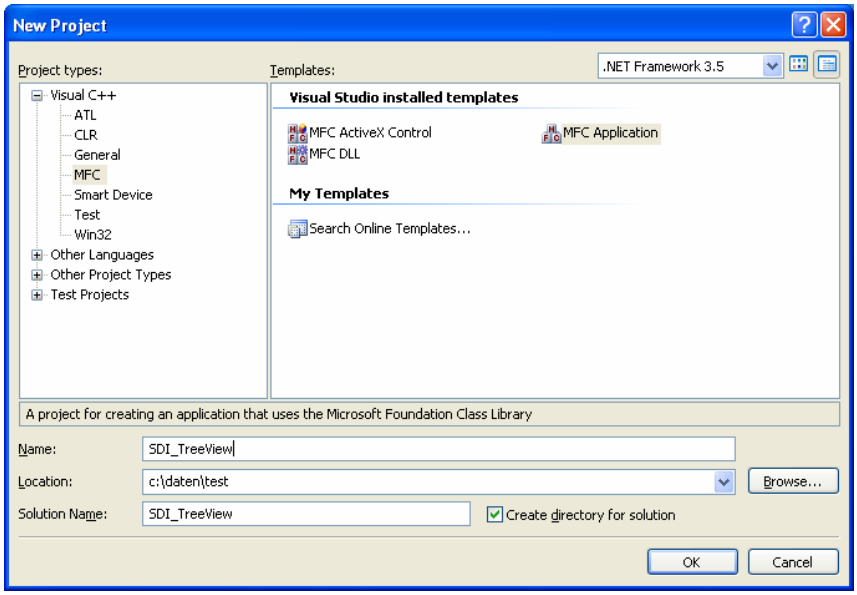


Abbildung 3 MFC-Anwendung erstellen

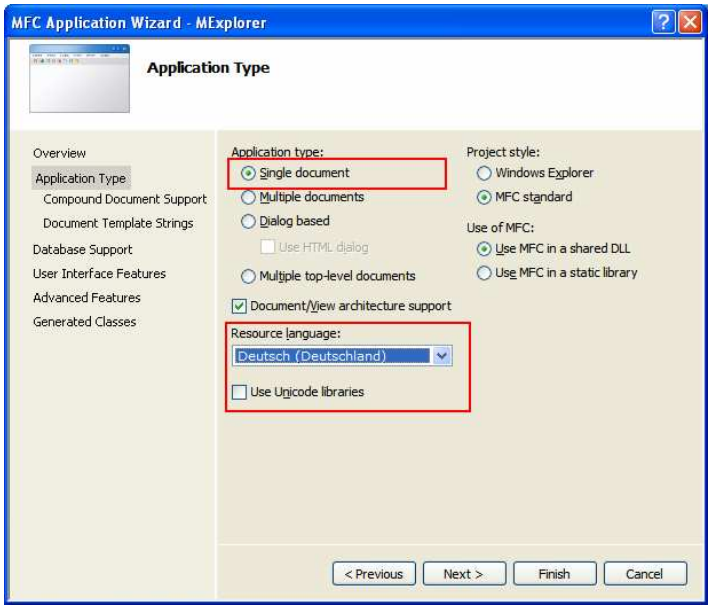


Abbildung 4 SDI-Anwendung

Für einfache Dialoge benötigt man keine Document/View Achitektur. Schaden kann aber auch. Die Speicherung ist im jedem Fall einfacher.

Wichtig: Hier Änderung von CView in CTreeView

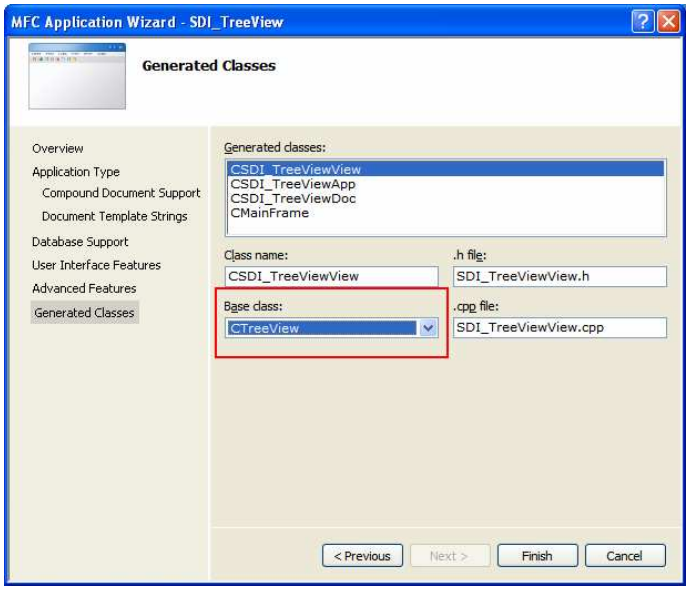


Abbildung 5 Ende des Wizards

Step1

2) Tree-Style definieren in Datei CSDI_TreeViewView.cpp

```
BOOL CSDI_TreeViewView::PreCreateWindow(CREATESTRUCT& cs) {
    if (!CTreeView::PreCreateWindow (cs))
        return FALSE;

    cs.style |= TVS_HASLINES | TVS_LINESATROOT | TVS_HASBUTTONS |
        TVS_SHOWSELALWAYS;
    return TRUE;
}
```

3) Eintragen einer Methode zum Einfügen eines Knotens

Datei: CSDI_TreeViewView.h:
void insertTree(int step, HTREEITEM hParent, CString sPath);

Datei: CSDI_TreeViewView.cpp:
void insertTree(int step, HTREEITEM hParent, CString sPath) {

Die Daten einer Zeile müssen in eine Klasse bzw. einer Struktur gespeichert werden. Die Methode „OnGetdispinfo“ benutzt diese Daten, um für jede Zeile die Tabelle aufzubauen.

```
typedef struct tableITEMINFO {
   CString     sName;
   unsined int Matrnr;
   CString    sPruefung;
   double note;
} tableITEMINFO;
```

Beispiel-Menüs

Beispiel1:

Einfaches Beispiel zum Eintragen von Knoten

Beispiel2:
Beispiel zum Eintragen von Knoten mit Symbolen
Jeweils Symbol zum Anzeigen und
Symbol beim Anklicken

Beispiel3:
Verwendung des komplexen Konstruktors

Beispiel4:
Verwendung der Struktur TVINSERTSTRUCT

Beispiel5:
Verwendung der Struktur TVINSERTSTRUCT

Beispiel6:
Aufbau eines Baumes, mit Übergabe einer Struktur, in der individuelle Informationen gespeichert sind.
Folgende Schritte sind dafür notwendig:

1) Deklaration der Struktur „treeITEMINFO“ in der Header-Datei

2) Eintragen der onSelChange Methode in der Headerdatei:

```
//{{AFX_MSG(CSDI_TreeViewView)
afx_msg void OnSelchanged(NMHDR* pNMHDR, LRESULT* pResult);
//}}AFX_MSG
DECLARE_MESSAGE_MAP()
```

3) Eintragen der onSelChange Methode in der MAP-Loop

```
BEGIN_MESSAGE_MAP(CSDI_TreeViewView, CTreeView)
  ON_NOTIFY_REFLECT(TVN_SELCHANGED, OnSelchanged)
END_MESSAGE_MAP()
```

4) Eintragen der onSelChange Methode in der Quellcodedatei:

```
void CSDI_TreeViewView::OnSelchanged(NMHDR* pNMHDR, LRESULT* pResult)
{
  if (m_selChange) {        // verhindert bei den normalen Beispielen einen Absturz
    ITEMINFO *p1;
    CString sStr;
    NM_TREEVIEW* pNMTreeView = (NM_TREEVIEW*)pNMHDR;
    p1 = (ITEMINFO *)pNMTreeView->itemNew.lParam;
    sStr.Format("Name: %s\nid: %d",p1->sName,p1->id);
    AfxMessageBox( sStr );
  }
  *pResult = 0;
}
```

Eintragen von Menüs in ein MFC-Projekt

- Auswahl des Registers „Recource“

- Auswahl des Eintrags „Menü“

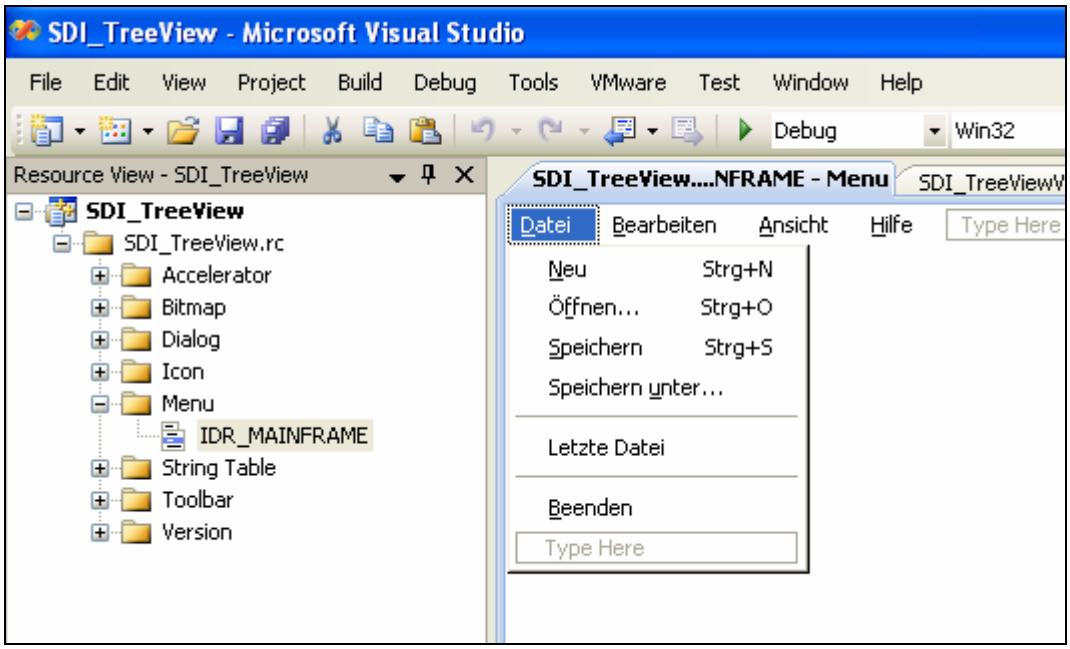


Abbildung 6 Menü Erstellung

- Menü Ansicht anklicken
- Taste „Einfügen“ drücken
- Mit rechter Maustaste „Properties“ auswählen
- Eintragen der Caption „Tree“
- Mit der Maus in die unteren Elemente klicken und einen Text eintragen

Wichtig:

Die Menüs sind jetzt schon im Programm,.aber noch disabled, da sie keine Klick-Event-Methoden haben.

Mit rechter Maustaste erreicht man den Eintrag „Add Event Handler“. Dabei ist es wichtig, in welcher Klasse der Mausevent eingetragen werden soll.

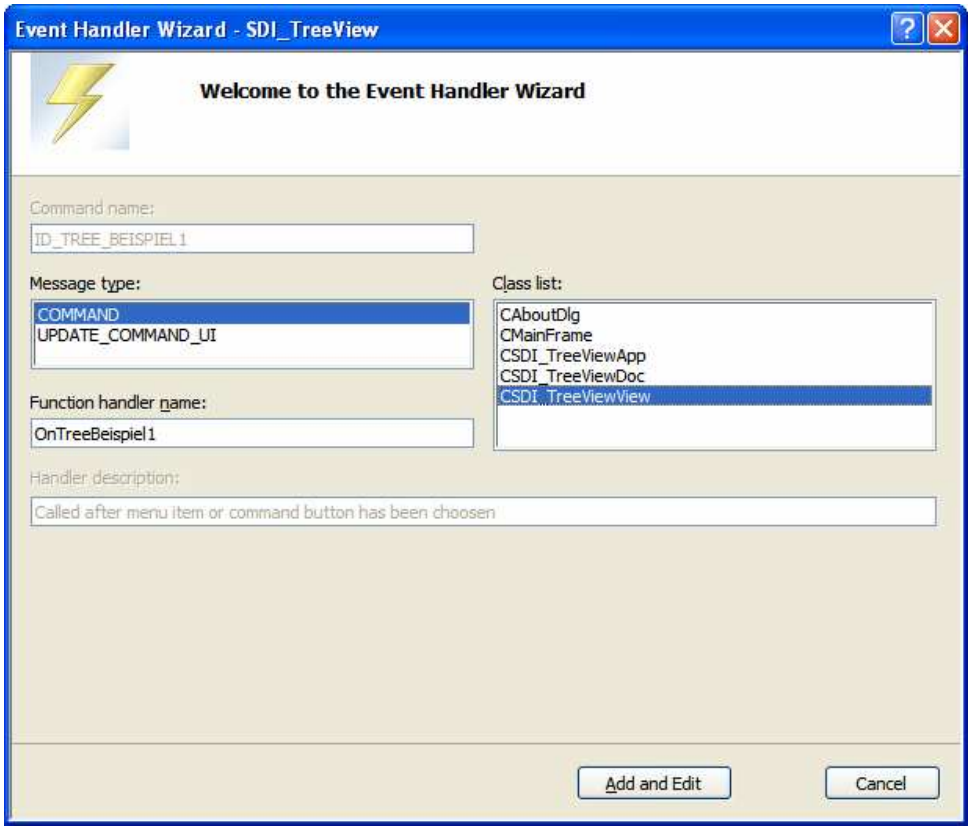


Abbildung 7 Erzeugung eines Mausevent in einer Klasse

Eintragen von Symbolen in einem CTreeView

Eintragen in der Headerdatei:

```
CImageList m_ilDrives;
```

Eintragen in der Quellcodedatei:

```
m_ilDrives.Create (IDB_DRIVEIMAGES, 16, 1, RGB (255, 0, 255));
GetTreeCtrl ().SetImageList (&m_ilDrives, TVSIL_NORMAL);
```

Die angegebene Farbe bezieht sich auf die Transparenzfarbe bzgl. der Bitmap.

Konstanten:

- LVSIL_NORMAL
- LVSIL_SMALL