# Eigenschaften der GUI-Elemente

## *CButton GUI-Overklasse*

- EnableWindow( bool ); // enabled des GUI-Elements
- 
- ShowWindow(SW_SHOW) // sichtbar
- ShowWindow(SW_HIDE) // unsichtbar
- 
- SetCheck(bool) // setzte / löscht den Status eines radiobuttons

## *CStatic*

- Align Text
- No Wrap
- Simple
- Static Edge
- Sunken
- Visible // ShowWindow

### Hinweis:

Alle Label-Elemente haben die defaultmäßig die ID: IDC_STATIC
Damit kann man keine Membervariable einrichten (doppelte Namen)
Erst umbenennen

## *CEdit*

- Align Text
- Border
- Lowercase
- Uppercase
- Number
- Multiline
- Password
- Read Only
- Visible // ShowWindow

## *CRadiobutton, besser als Klasse CButton*

- Bitmap
- Flat
- Multi Line
- Static Edge
- Visible // ShowWindow

- EnableWindow( bool );  // enabled des GUI-Elements
-
- ShowWindow(SW_SHOW)  // sichtbar
- ShowWindow(SW_HIDE)  // unsichtbar
-
- SetCheck(bool)  // setzte / löscht den Status eines radiobuttons

## *CCheckbox, besser als Klasse CButton*

- Bitmap
- Flat
- Multi Line
- Right Align Text
- Right to Left Reading Order
- Static Edge
- Visible  // ShowWindow

- EnableWindow( bool );  // enabled des GUI-Elements
- ShowWindow(SW_SHOW)  // sichtbar
- ShowWindow(SW_HIDE)  // unsichtbar
- SetCheck(bool)  // setzte / löscht den Status eines radiobuttons

## *CSpinButtonCtrl*

- Arrow Keys
- No Thousands
- Orientation
- Static Edge
- Wrap
- Auto Buddy
- Set Buddy Integer
- Visible  // ShowWindow

- Einfügen eines Drehfeldes
- Einfügen eines Editorfeldes
- Ändern der Number-Eigenschaften im Editorfeld
- Sofort danach Einfügen des Drehfeldes
- Ändern der beiden Buddy-Eigenschaften im Register „Formate“
- Ändern der ID auf z. B. IDC_SPIN_XTICKS
- Damit ist die Verknüpfung zum Editorfeld definiert
- Leider ist die Reihenfolge (Decrement, Increment) falsch

**Abhilfe:**

Dazu muss in der OnInitDialog-Methode folgender Code eingefügt werden:

```
m_SpinEdit.SetRange(0,12);     // (Membervariable Control)
```

## *CComboBox*

- Lowercase
- Uppercase
- Vertical Scrollbar
- Type Dropdown (list+Eingabe)
  Dropdownlist (nur liste)
- Data Semikolon als Trenner
- Sort
- Visible // ShowWindow
- GetCurSel() // Membervariable Control
- SetCurSel(int index) // Membervariable Control

- AddString
- DeleteItem mit Struktur LPDELETEITEMSTRUCT lpDeleteItemStruct;
- DeleteString
- GetCount
- GetCurSel
- InsertString
- ResetContent
- SelectString
- Clear
- Copy
- Cut
- Paste

## CLISTBOX:

- AddString
- DeleteItem mit Struktur LPDELETEITEMSTRUCT lpDeleteItemStruct;
- DeleteString
- GetCount
- GetCurSel
- GetSel
- GetSelItems
- GetText
- GetTextLen
- InsertString
- ItemFromPoint
- ResetContent
- SelItemRange
- SetCurSel
- SetItemData
- SetSel

## Init einer CLISTBOX:

In InitDialog:

Hinter // TODO: Add extra initialization here

```
m_combo_ctrl.AddString(_T("44567"));
m_combo_ctrl.SetCurSel(2);
```

## *Slider*

Methoden:

- Int GetPos()
- void SetPos(int i);
- SetPageSize(int i); // Anzahl der Einheiten pro Mausklick
- GetTic // Aufbau der Skala
- GetTicPos // Aufbau der Skala
- GetTicArray // Aufbau der Skala
- GetNumTicks // Aufbau der Skala
- ClearSel // Selektion löschen

## Einbau einer Slider-Change-Event

**In der Headerdatei:**
als public
afx_msg void OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);

**In der Dlg.cpp**

```
void CPagerCtrlPage::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
    int nBorderSize = m_sliderBorderSize.GetPos();
    CString sStr;
    sStr.Format(_T("%d"), nBorderSize);
    m_staticBorderSize.SetWindowText(sStr);
    AfxMessageBox(sStr);
}
```

**In der Dlg.cpp**

```
BEGIN_MESSAGE_MAP(CTestSliderDlg, CDialog)
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    //}}AFX_MSG_MAP
    ON_WM_HSCROLL()
END_MESSAGE_MAP()
```

| nSBCode: | Aktionscode: |
|---|---|
| TB_TOP | Pos1 gedrückt |
| TB_BOTTOM | Ende gedrückt |
| TB_LINEDOWN | Taste Pfeil links oder rechts |
| TB_PAGEDOWN | Taste Page Down gedrückt |
| TB_PAGEUP | Taste Page Up gedrückt |
| TB_THUMBTRACK | Regler wird mit der Maus gezogen |
| TB_THUMBPOSITION | Linke Maustaste wurde nach dem Ziehen losgelassen |
| TB_ENDTRACK | Taste oder Maustaste wurde nach dem Ziehen losgelassen |

### *Beispiele*

http://msdn.microsoft.com/en-us/library/ms386053(VS.71).aspx

# Klassen und GUI-Elemente

| Control | MFC class | Description |
|---|---|---|
| animation | CAnimateCtrl | Displays successive frames of an AVI video clip |
| button | CButton | Pushbuttons that cause an action; also used for check boxes, radio buttons, and group boxes |
| combo box | CComboBox | Combination of an edit box and a list box |
| date and time picker | CDateTimeCtrl | Allows the user to choose a specific date or time value |
| edit box | CEdit | Boxes for entering text |
| extended combo box | CComboBoxEx | A combo box control with the ability to display images |
| header | CHeaderCtrl | Button that appears above a column of text; controls width of text displayed |
| hotkey | CHotKeyCtrl | Window that enables user to create a "hot key" to perform an action quickly |
| image list | CImageList | Collection of images used to manage large sets of icons or bitmaps (image list isn't really a control; it supports lists used by other controls) |
| list (Explorer) | CListCtrl | Window that displays a list of text with icons |
| list box | CListBox | Box that contains a list of strings |
| month calendar | CMonthCalCtrl | Control that displays date information |
| progress | CProgressCtrl | Window that indicates progress of a long operation |
| rebar | CRebarCtrl | Tool bar that can contain additional child windows in the form of controls |
| rich edit | CRichEditCtrl | Window in which user can edit with character and paragraph formatting (see Classes Related to Rich Edit Controls) |
| scroll bar | CScrollBar | Scroll bar used as a control inside a dialog box (not on a window) |
| slider | CSliderCtrl | Window containing a slider control with optional tick marks |
| spin button | CSpinButtonCtrl | Pair of arrow buttons user can click to increment or decrement a value |
| static-text | CStatic | Text for labeling other controls |
| status bar | CStatusBarCtrl | Window for displaying status information, similar to MFC class CStatusBar |
| tab | CTabCtrl | Analogous to the dividers in a notebook; used in "tab dialog boxes" or property sheets |
| toolbar | CToolBarCtrl | Window with command-generating buttons, similar to MFC class CToolBar |
| tool tip | CToolTipCtrl | Small pop-up window that describes purpose of a toolbar button or other tool |
| tree | CTreeCtrl | Window that displays a hierarchical list of items |

## *CList Box*

- Border
- Group
- MultiColumn
- Selection
- Sort
- Vertical Scrollbar
- Visible // ShowWindow
- GetCurSel() // Membervariable Control
- SetCurSel(int index) // Membervariable Control

### Init einer Listbox:

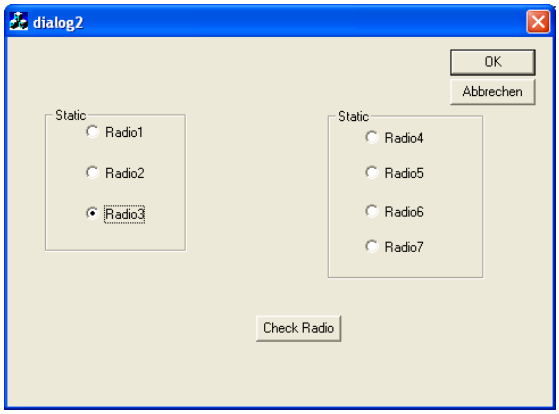
In InitDialog:

Hinter // TODO: Add extra initialization here
m_liste_ctrl.AddString(_T("IB"));

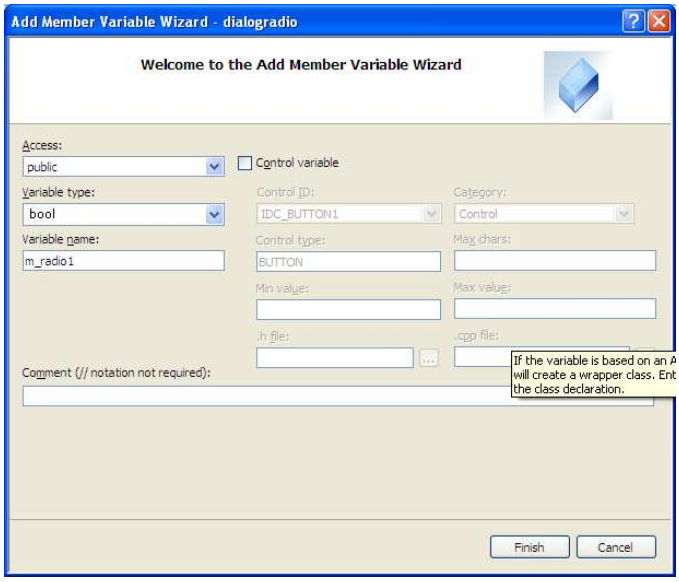
## *UpdateData(bRichtung)*

- TRUE ⇒ Aus GUI nach Membervariable
- FALSE ⇒ Aus Membervariable nach GUI

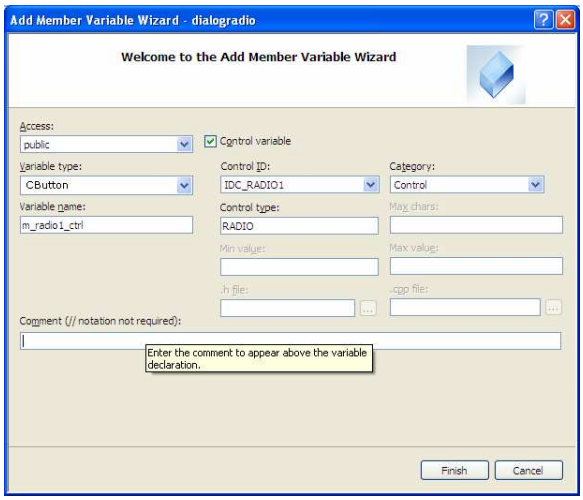
# *2. Übung*



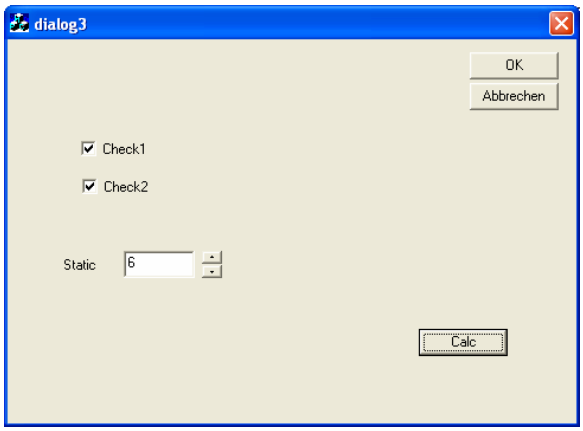

**Einfügen einer Member-Variablen Wert**



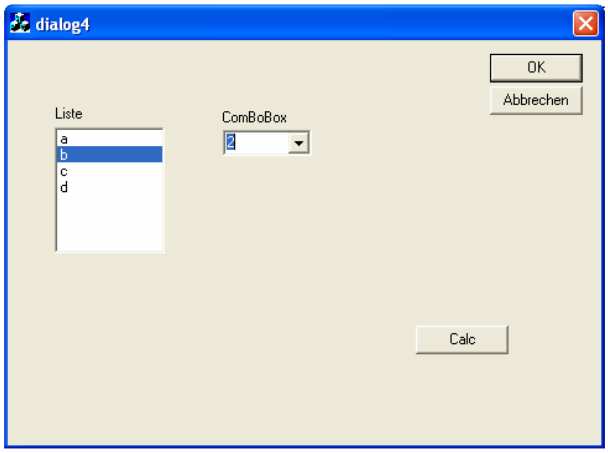

**Einfügen einer Member-Variablen Control**



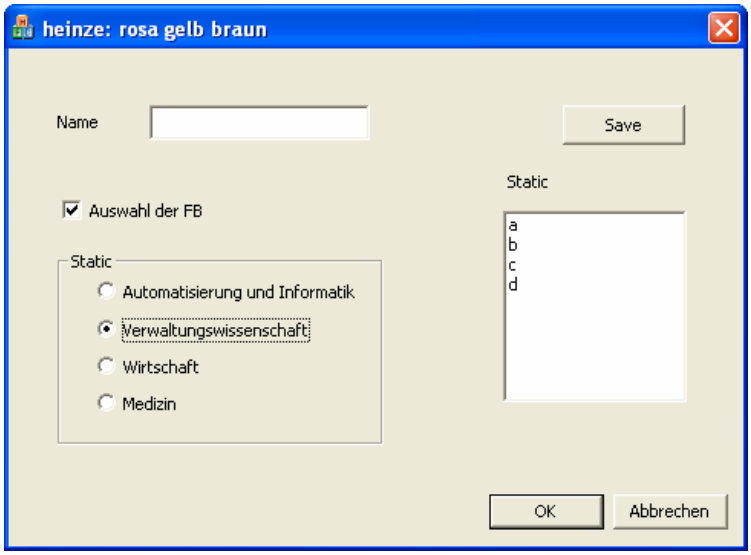
## 3. Übung




## 4. Beispiel




## 5. Übung




Checkbox kontrolliert enabled der Radiobuttons

```
void CheinzeDlg::OnPaint()
{
   if (IsIconic())
   {
      CPaintDC dc(this); // Gerätekontext zum Zeichnen

      SendMessage(WM_ICONERASEBKGND, reinterpret_cast<WPARAM>
                  (dc.GetSafeHdc()), 0);

      // Symbol in Clientrechteck zentrieren
      int cxIcon = GetSystemMetrics(SM_CXICON);
      int cyIcon = GetSystemMetrics(SM_CYICON);
      CRect rect;
      GetClientRect(&rect);
      int x = (rect.Width() - cxIcon + 1) / 2;
      int y = (rect.Height() - cyIcon + 1) / 2;

      // Symbol zeichnen
      dc.DrawIcon(x, y, m_hIcon);
   }
   else
   {
      CPaintDC dc(this); // Gerätekontext zum Zeichnen
      CRect rect;
      GetClientRect(&rect);
      CBrush brush1( RGB(255,0,255) );
      dc.SelectObject(&brush1);
      dc.Rectangle(&rect);

      CDialog::OnPaint();
   }
}
```

## *2. Übung*

```
void CDialog2Dlg::OnBnCalc()
{
     int i1, i2;
     UpdateData(true);
     i1 = m_Radio1;
     i2 = m_Radio2;
     CString sStr;
     sStr.Format("i1: %d\ni2: %d",i1,i2);
     AfxMessageBox(sStr);
}

     int iButton = GetCheckedRadioButton(IDC_RADIO1, IDC_RADIO3) - IDC_RADIO1;
     CString sStr;
     sStr.Format(_T("Radiowert: %d"),iButton);
     AfxMessageBox( sStr );


void CdialogradioDlg::OnBnClickedCheck1()
{
     // TODO: Add your control notification handler code here
   CButton * pBtn1 = (CButton *) GetDlgItem(IDC_RADIO1);
   CButton * pBtn2 = (CButton *) GetDlgItem(IDC_RADIO2);
   CButton * pBtn3 = (CButton *) GetDlgItem(IDC_RADIO3);
     UpdateData(true);
     // nur eine
   if (m_checkbox) {
       pBtn1->EnableWindow(true);
       pBtn2->EnableWindow(true);
       pBtn3->EnableWindow(true);
      }
      else {
       pBtn1->EnableWindow(false);
       pBtn2->EnableWindow(false);
       pBtn3->EnableWindow(false);
       //m_radio1_ctrl.EnableWindow(false);
      }

}
```

## *3. Übung*

```
void CDialog3Dlg::OnBnCalc()
{
     BOOL b1, b2;
     int zahl;
     UpdateData(true);
     b1 = m_Check1;
```

```
    b2 = m_Check2;
    zahl = m_Zahl;
    CString sStr;
    sStr.Format("Check1: %d\nCheck2: %d\nZahl: %d",b1,b2,zahl);
    AfxMessageBox(sStr);


}
```

## 4. Übung

**InitDialog**

```
 CListBox * pListe1 = (CListBox *) GetDlgItem(IDC_LISTE1);
 CComboBox * pListe2 = (CComboBox *) GetDlgItem(IDC_LISTE2);

    pListe1->AddString("a");
    pListe1->AddString("b");
    pListe1->AddString("c");
    pListe1->AddString("d");

    // CListBoxm_liste1_ctrl;
    m_liste1_ctrl.AddString("3333");

    pListe2->AddString("1");
    pListe2->AddString("2");
    pListe2->AddString("3");
    pListe2->AddString("4");
    pListe2->AddString("5");
    pListe2->AddString("6");

    CDialog::OnInitDialog();
```

**Membervariablen**

- m_liste1
- m_liste2

**void CDialog4Dlg::OnBnCalc()**

```
{
    UpdateData(true);
    CString sStr1;
    CString sStr2;
    sStr1 = m_Liste1;
    sStr2 = m_Liste2;
    AfxMessageBox(sStr1 + "\n" + sStr2);
}
```

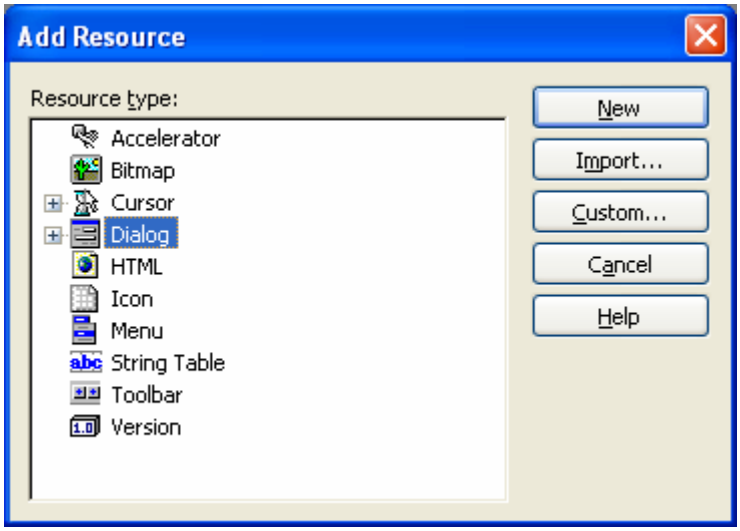
# Neuer Modale Dialog

## *Dialog-Erstellen*

Recourcen Register

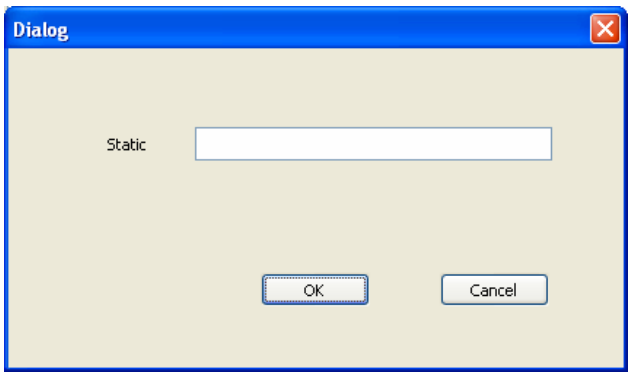
Dialog Eintrag

Add Dialog



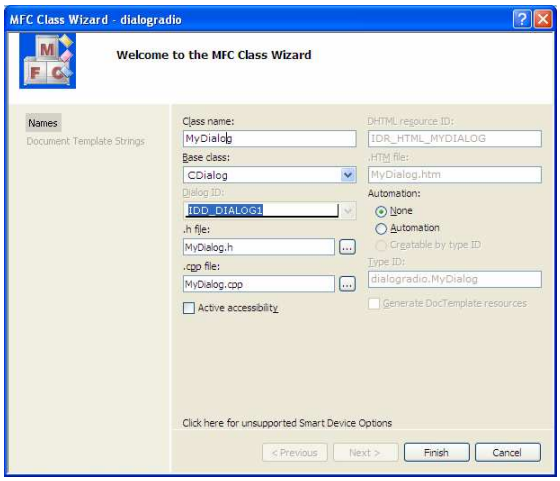

### Schalter New

## *Eintragen der GUI-Elemente*




## *Erzeugen einer Klasse*

- Ins Dialogfenster klicken




## Wechseln ins erste Dialogfenster

Eintragen eines Schalters myDialog
OnClick Event erzeugen

#include "MyDialog.h"

## Eintragen in Click

```
void CdialogradioDlg::OnBnClickedButton3()
{
  MyDialog dlg;
      //dlg.m_Label = "Radius";
      //dlg.m_Zahl = X;
      if (dlg.DoModal() == IDOK) {
            //X = dlg.m_Zahl;
            //pDoc->setX(X);
            //pDoc->UpdateAllViews( this );    // oder NULL
            //Invalidate();
      }  // if
}
```

member-variablen definieren
Vorher setzen, nachher auslesen

```
void CdialogradioDlg::OnBnClickedButton3()
{
  MyDialog dlg;
      dlg.m_edit = _T("Bitte Namen eingeben");
      if (dlg.DoModal() == IDOK) {
            AfxMessageBox(dlg.m_edit);
      }  // if
      else {
            AfxMessageBox(_T("Schade, dass Sie abgebrochen haben") );
      }
}
```

isWindowVisible

**CWinApp**

- m_pszAppName — Specifies the name of the application.
- m_hInstance — Identifies the current instance of the application.
- m_hPrevInstance — Set to NULL in a 32-bit application.
- m_lpCmdLine — Points to a null-terminated string that specifies the command line for the application.
- m_nCmdShow — Specifies how the window is to be shown initially.
- m_bHelpMode — Indicates if the user is in Help context mode (typically invoked with SHIFT+F1).
- m_pActiveWnd — Pointer to the main window of the container application when an OLE server is in-place active.
- m_pszExeName — The module name of the application.
- m_pszHelpFilePath — The path to the application's Help file.
- m_pszProfileName — The application's .INI filename.
- m_pszRegistryKey — Used to determine the full registry key for storing application profile settings.