

Grundlagen in C# und .net

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- Raum 2.202
- Tel. 03943 / 659 338



Inhalt

- TabbedPane (Register)
- ListView
- Tree
- Tabelle
- MDI-Programme
- Erweiterte Grafik
- Threads und Semaphore
- Datenbanken



Weitere GUI-Elemente / Techniken

- Clipboard
- LINQ
- Drucken, Preview
- Drag & Drop
- Templates, Generische Datentypen
- Eigene GUI-Komponenten
- Windows-Server
- Installation

Ein- und Ausgabe

Hauptklassen:

- Stream
 - Byteweise lesen und schreiben
- FileStream
 - Byteweise lesen und schreiben
- StreamReader / StreamWriter
 - Lesen und schreiben mit Datentypen, auch Strings
- BinaryFormatter
 - Lesen und schreiben eines Objektes, serialize
- XmlSerializer
 - Lesen und schreiben mittels XML
- Package: System.IO;

FileStream: Konstruktor

- Konstruktor(String sFilename, FileMode mode)
- Konstruktor(String , FileMode , FileAccess)
 - FileMode.Append Öffnen anhängen oder neu erzeugen
 - FileMode.Create Wird immer neu erzeugt
 - FileMode.CreateNew Wenn nicht existiert, dann neu erzeugt
 - FileMode.Open Öffnen der Datei
 - FileMode.OpenOrCreate Öffnen der Datei oder neu erzeugen, R/W
 - FileMode.Truncate Öffnen und löschen R/W

 - FileAccess.Read
 - FileAccess.Write
 - FileAccess.ReadWrite



FileStream: Properties und Methoden

- Properties
 - Name Dateiname
 - Length
 - Position
- Methoden
 - Close
 - Lock
 - UnLock
 - Read / ReadByte
 - Write / WriteByte
 - Seek
 - SetLength
 - Synchronized



StreamReader: Konstruktor

- Konstruktor(Stream)
- Konstruktor(String)
- Konstruktor(Stream, Boolean bigEndian)
- Konstruktor(Stream, Encoding)
- Konstruktor(Stream, Encoding, Boolean bigEndian)

- Encoding
 - System.Text.Encoding.ASCII
 - System.Text.Encoding.BigEndianUnicode
 - System.Text.Encoding.Default
 - System.Text.Encoding.Unicode
 - System.Text.Encoding.UTF32
 - System.Text.Encoding.UTF7
 - System.Text.Encoding.UTF8

StreamReader: Properties und Methoden

- Properties
 - EndOfStream

- Methoden
 - Close
 - Peek
 - Read
 - ReadBlock
 - ReadToEnd
 - ReadLine
 - Synchronized

StreamWriter: Konstruktor

- Konstruktor(Stream)
- Konstruktor(String)
- Konstruktor(Stream, Boolean bigEndian)
- Konstruktor(Stream, Encoding)
- Konstruktor(Stream, Encoding, Boolean bigEndian)

- Encoding
 - System.Text.Encoding.ASCII
 - System.Text.Encoding.BigEndianUnicode
 - System.Text.Encoding.Default
 - System.Text.Encoding.Unicode
 - System.Text.Encoding.UTF32
 - System.Text.Encoding.UTF7
 - System.Text.Encoding.UTF8

StreamWriter: Properties und Methoden

- Properties
 - NewLine Holen und Setzen des Zeilenumbruchs

- Methoden
 - Close
 - Flush
 - Write
 - WriteLine
 - Synchronized

BinaryFormatter:

- Dient zum Laden und Speichern eines Objektes
- Konstruktor()

- Properties
 - Binder
 - FilterLevel
 - TypeFormat

- Methoden
 - Serialize Speichern
 - Deserialize Laden



XmlSerializer:

- Dient zum Laden und Speichern eines Objektes im XML-Format
- Konstruktor()
- Konstruktor(Type)
- Konstruktor(Type, String Filename oder String URL)
- Konstruktor(Type, Type [])
- Konstruktor(Type, XMLRootAttribute)
- Konstruktor(Type, XmlAttributeOverrides)

- Type = Klasse, à la Vector <.>
- XmlSerializer serializer = new XmlSerializer(typeof(OrderedItem));

- using System.IO;
- using System.Xml;
- using System.Xml.Serialization;



Beispiel Einlesen einer ASCII-Datei

```
FileStream inFile = new FileStream(sFilename, FileMode.Open);

StreamReader inStream = new StreamReader(inFile);
while (! inStream.EndOfStream) {
    String value = inStream.ReadLine();
}

inStream.Close();
inFile.Close();
```

- using System.IO;



Beispiel Schreiben einer ASCII-Datei

```
FileStream oFile = new FileStream(sFilename, FileMode.Create);

StreamWriter oStream = new StreamWriter(oFile);

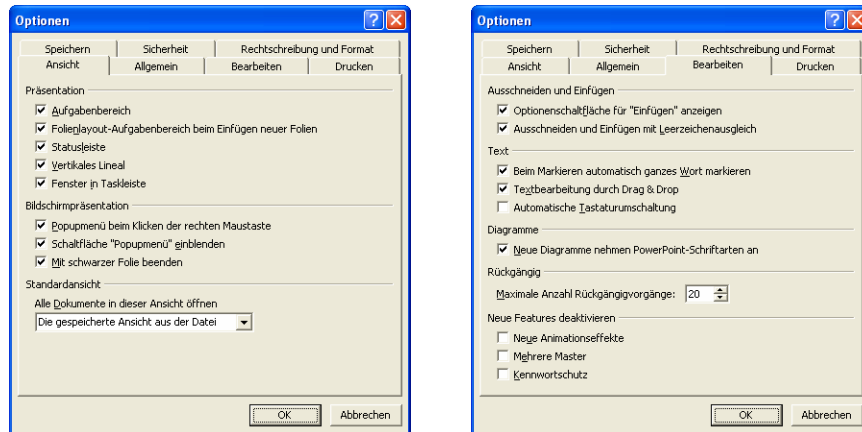
foreach (String value in liste) {
    oStream.WriteLine(value);
}

oStream.Close();
oFile.Close();
```



TabbedPane

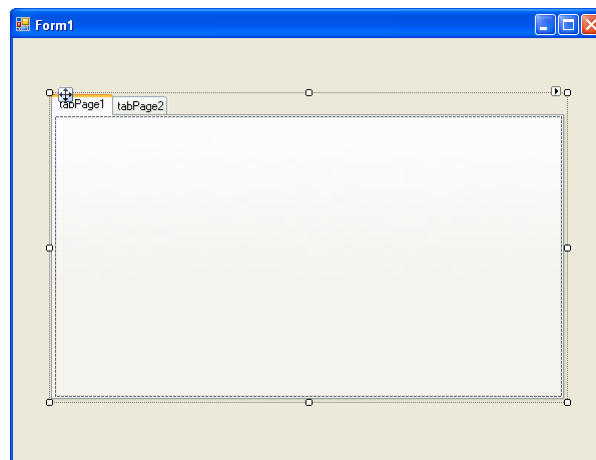
Registerkarten dienen der Gruppierung von Dialogfenster in umfangreichen Dialogfenstern



Registerkarten

Ablauf

- Projekt erstellen
- TabControl ins Dialogfenster eintragen



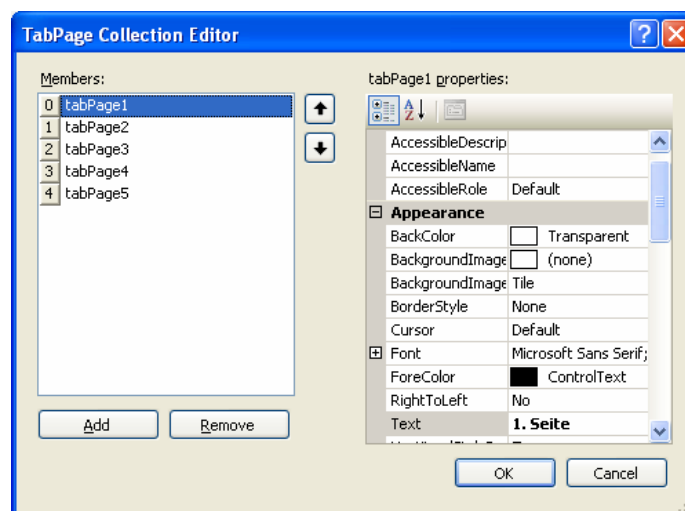
Registerkarten

Ablauf

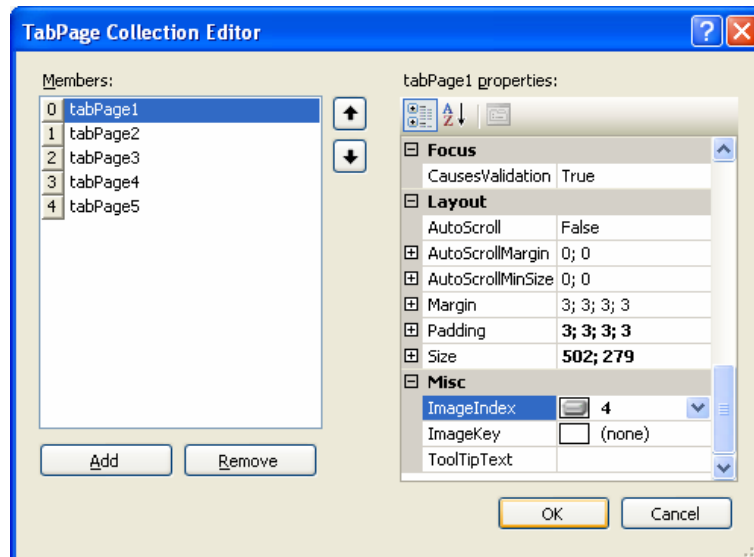
- Unterschiede
 - Klick in die Kopfzeile (Aktiviert das gesamte Element)
 - Klick in eine Seite (Aktiviert die jeweilige Seite)
 - Attribut Text: Überschrift der Seite
 - Erst dann kann man Elemente einfügen
- Eigenschaften:
 - Multiline: Die Registerbeschriftungen werden gescrollt oder umgebrochen
 - Sizemode: Ausrichtung der Karten
 - Padding: Rand
 - SelectedIndex: setzt und zeigt das aktuelle Register (Plausibilitätsprüfung)
 - Visible
 - ImageList: Liste zu Symbolen, in den Überschriften

Registerkarten

Bearbeiten der Register (Reihenfolge): Eintrag TabPages im Propertiefenster



Registerkarten



Registerkarten

- tabPage Events
 - Enter
 - Leave
 - Click
 - DoubleClick
 - PreviewKeyDown (fängt Tasten ab !)

Einstellungenfenster

Einstellungsfenster sind modale Dialoge, die Eingaben für Programme entgegen nehmen

Ablauf:

- Dialogfenster im Projekt erstellen: FEinst
- Einbau der grafischen Elementen
- Schalter BnOk erhält den DialogResult Ok
- Schalter BnEsc erhält den DialogResult Cancel
- Ok-Click
 - Plausibilität abfragen

Aufruf:

- Erstellen des Dialogs: FEinst frame = new FEinst();
- Setzen der Attribute
- frame.ShowDialog();
- Abfragen der auf Ok
- Abfragen und Setzen der Werte



FB Automatisierung und Informatik: GUI mit Visual Studio: .net

21

ListView

Die ListView-Komponente zeigt Daten in einer „Tabelle“ an

Eigenschaften

- Tabellen / Listendarstellung (Kleine, Große Symbole, Liste, Report, Details)
- Darstellung mit Symbolen
- Eintragen von Daten
- Top, Left, Width, Height
- Dock (None, Left, Top, Bottom, Right, Full)
- FullrowSelect
- GridLines (anzeigen)
- Zeilen sind selektierbar
- Sortierbar (siehe Beispiel listView1: .Sorting = SortOrder.Ascending)



FB Automatisierung und Informatik: GUI mit Visual Studio: .net

22

ListView

```
listView1.View = View.Details;           // Symbol, Report vs. Details
listView1.LabelEdit = true;              // EditModus
listView1.AllowColumnReorder = true;     // Spalten verschieben
listView1.CheckBoxes = true;             // CheckBox
listView1.FullRowSelect = true;          // zeilenweise markieren
listView1.GridLines = true;              // Gitterlinien
listView1.Sorting = SortOrder.Ascending; // Sortiere (Alphanumerisch oder separat)

// Spalten -2 Breite wird autom. Ausgerechnet, -1 sehr klein
listView1.Columns.Add("Nr", 100, HorizontalAlignment.Left);
listView1.Columns.Add("Name", 75, HorizontalAlignment.Left);
listView1.Columns.Add("Matrnr", -2, HorizontalAlignment.Left);
listView1.Columns.Add("Note", -2, HorizontalAlignment.Center);
// es wird immer eine zusätzliche Spalte angezeigt
```



ListView

```
// Symbole (small und large)
ImageList imageListSmall = new ImageList();
ImageList imageListLarge = new ImageList();

// Initialize the ImageList objects with bitmaps.
imageListSmall.Images.Add(Bitmap.FromFile("C:\\Daten\\MySmallImage1.bmp"));
imageListSmall.Images.Add(Bitmap.FromFile("C:\\Daten\\MySmallImage2.bmp"));
imageListLarge.Images.Add(Bitmap.FromFile("C:\\Daten\\MyLargeImage1.bmp"));
imageListLarge.Images.Add(Bitmap.FromFile("C:\\Daten\\MyLargeImage2.bmp"));

// Zuweisung
listView1.LargeImageList = imageListLarge;
listView1.SmallImageList = imageListSmall;
```



ListView

```
ListViewItem item1 = new ListViewItem("1",0); // 0 = 1. Bild
item1.Checked = true;
item1.SubItems.Add("Meyer");    // next Column
item1.SubItems.Add("14721");
item1.SubItems.Add("4.0");
listView1.Items.Add(item1);
```

```
ListViewItem item2 = new ListViewItem("2",1); // 1 = 2. Bild
item2.SubItems.Add("Schulze");
item2.SubItems.Add("16234");
item2.SubItems.Add("1.3");
listView1.Items.Add(item2);
```



ListView

Sortieren (1)

Vorgehensweise

- Klasse erstellen, abgeleitet von IComparer
 - Überschreiben der Methode `int compare(object, object)`
 - `compareResult = listViewX.SubItems[iSort].Text.CompareTo(listviewY.SubItems[iSort].Text);`
 - Private Variablen für Sortfeld, auf- bzw. Absteigend
 - set- und get Methoden
- Sortierung aktivieren
 - `lvwColumnSorter = new ListViewColumnSorter();`
 - `this.listView1.ListViewItemSorter = lvwColumnSorter;`



ListView

Sortieren (2)

- **ListView ColumnClick Event eintragen**

- Check, welche Spalte
- ev. Richtung umkehren
- sonst Absteigend definieren

- **Aufruf der Sortierung**

- `this.listView1.Sort();`



ListView

Object Anbindung

- **Das ListView-Object kann mittels Tag-Attribut eine Referenz erhalten**

- `ListViewItem item = new ListViewItem( std.name, 0);`
- `item.SubItems.Add( std.matrnr.ToString() );`
- `item.Tag = std;`
- `listView1.Items.Add(item);`

- **Aufruf des Doppelklicks-Events**

- `CStudent std;`
- `for (int i = 0; i < listView1.SelectedItems.Count; i++) {`
- `std = (CStudent) listView1.SelectedItems[i].Tag;`
- `}`



Tabelle (DataGridView)

■ Eigenschaften

- Anbindung an eine Datenbanktabelle
- Funktioniert aber auch ohne DBS
- Spalten und Zeilen
- Attribut Text: Überschrift der Seite
- Erst dann kann man Elemente einfügen
- Columns[?].Name Spaltennamen
- Selectionmode = DataGridViewSelectionMode.FullRowSelect
- MultiSelect = true oder false
- Formatierung der einzelnen Zellen

Tabelle (DataGridView)

■ Eigenschaften

- Anbindung an eine Datenbanktabelle
- Funktioniert aber auch ohne DBS
- Spalten und Zeilen
- Attribut Text: Überschrift der Seite
- Erst dann kann man Elemente einfügen
- Man kann auch neue Daten einfügen (DBS-Tabelle)
- Abhilfe: Attribut: ReadOnly auf true setzen

■ Ablauf:

- Erstellen eines Projektes
- Einfügen eines DataGridViews

Tabelle (DataGridView)

```
Grid1.RowCount = 1;
```

```
Grid1.ColumnCount = 5;
```

```
Grid1.Columns[0].Name = "MitarbeiterNr";
```

```
Grid1.Columns[1].Name = "Name";
```

```
Grid1.Columns[2].Name = "Vorname";
```

```
Grid1.Columns[3].Name = "Abteilung";
```

```
Grid1.Columns[4].Name = "Gehalt";
```

Attribut: Columns im Eigenschaftsdialog

Tabelle (DataGridView)

Zuweisung per Zeile

```
string[] row0 =
```

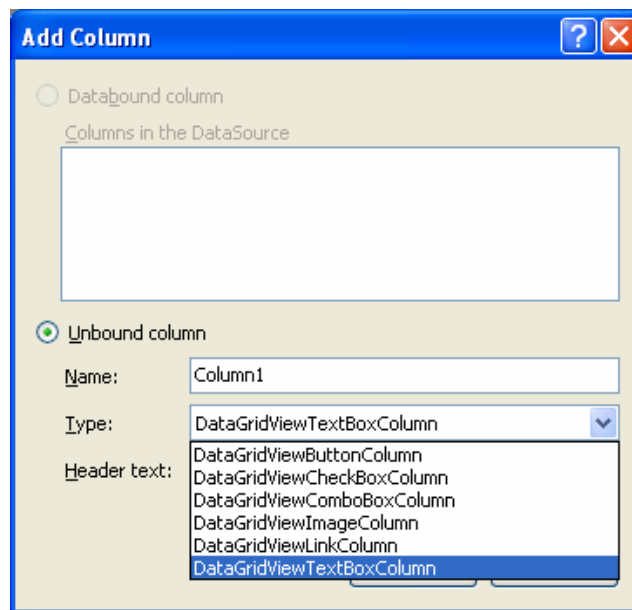
```
{ "1", "Meier", "Andreas", "Finanzen", "1.203" };
```

```
Grid1.Rows.Add(row0);
```

```
object[] row0 =
```

```
{ "1", "Meier", "Andreas", "Finanzen", "1.203", Color.Red, true };
```

```
Grid1.Rows.Add(row0);
```



Spalte mit einer ComboBox

```
DataGridViewComboBoxColumn comboBoxColumn =  
    new DataGridViewComboBoxColumn();
```

```
comboBoxColumn.Items.AddRange(  
    Color.Red, Color.Yellow, Color.Green,  
    Color.Blue, Color.Black);
```

```
comboBoxColumn.ValueType = typeof(Color);  
comboBoxColumn.Name="Lieblingsfarbe";  
Grid1.Columns.Add(comboBoxColumn);
```

Spalte mit einer Checkbox

```
DataGridViewCheckBoxColumn checkBoxColumn = new
DataGridViewCheckBoxColumn();
checkBoxColumn.HeaderText = "Extern";
checkBoxColumn.AutoSizeMode =
    DataGridViewAutoSizeColumnMode.DisplayedCells;
checkBoxColumn.FlatStyle = FlatStyle.Standard;
checkBoxColumn.ThreeState = true; // ???
checkBoxColumn.CellTemplate.Style.BackColor = Color.Beige;

Grid1.Columns.Add(checkBoxColumn);
```

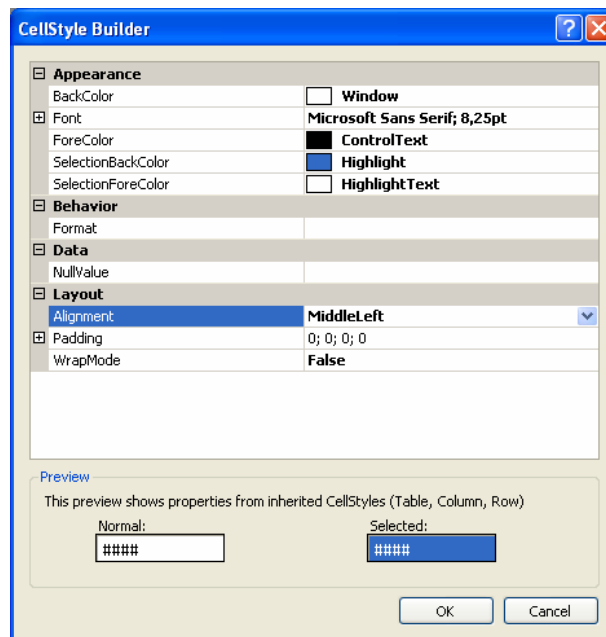
Tabelle (DataGridView)

Spalten lassen sich sortieren (ohne weiteren Quellcode)
Methode sort



	MitarbeiterNr	Name	Vorname	Abteilung	Gehalt
	2	Schmidt	Ralf	Marketing	5.104
	4	Opel	Hans-Dieter	Informatic	3.455
	66	Bischoff	Hugo	Forschung	5.434
	1	Meier	Andreas	Finanzen	1.203
	78	Schulze	Sabine	Controlling	7.655
▶	77	Ernst	Uwe	Controlling	7.656
	3	Bender	Hans	CEO	11.001
*					

DefaultCellStyle



Bestimmen der selektierten Zelle / Reihe

```
for (int i=0; i<(Grid1.SelectedCells.Count); i++)
{
    int col = Grid1.SelectedCells[i].ColumnIndex;
    int row = Grid1.SelectedCells[i].RowIndex;
    sStr1 = sStr1.Text + " " + col.ToString()+"/"+row.ToString();
    sStr2 = Grid1.SelectedCells[i].FormattedValue.ToString();
}
```

Bestimmen der selektierten Zelle / Reihe

	Mitarbe...	Name	Vorname	Abteilung	Gehalt	Liebling...	Extern2
1		Meier	Andreas	Finanzen	1.203	Red	☒
2		Schmidt	Ralf	Marketing	5.104	Blue	☒
3		Bender	Hans	CEO	11.001	Yellow	☒
4		Opel	Hans-Dieter	Informatic	3.455	Blue	☐
66		Bischoff	Hugo	Forschung	5.434	Black	☐
77		Ernst	Uwe	Controlling	7.656	Red	☒
78		Schulze	Sabine	Controlling	7.655	Yellow	☒
*							☐

Bestimmen der selektierten Zelle / Reihe

Wenn man keine Datenbank-Anbindung einträgt, kann man zusätzliche Reihen eintragen (Tab-Taste am Ende)

Folgende Attribute verhindern dieses Verhalten:

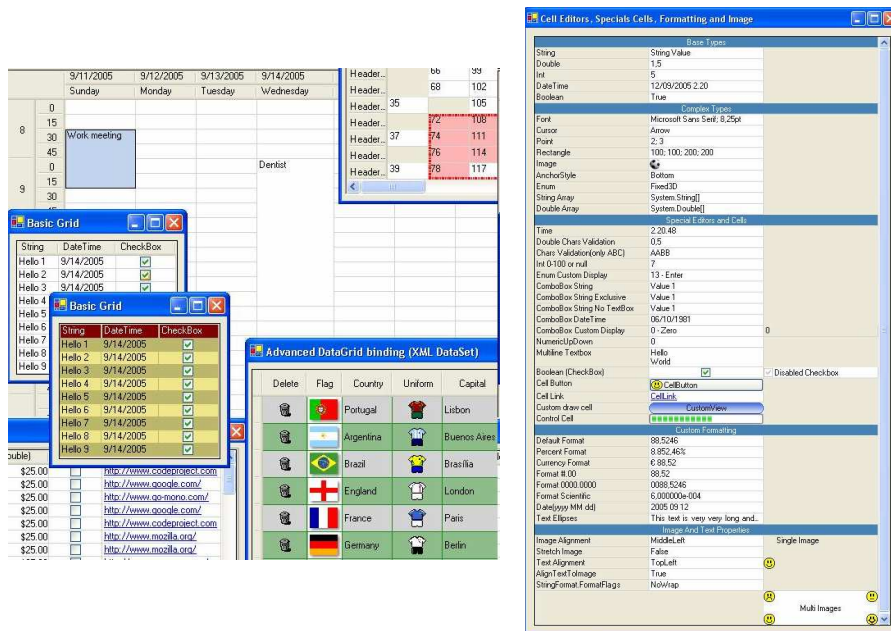
- AllowUserToAddRows=false
- AllowUserToDeleteRows=false

Dann wird aber eine "neue" Zeile oben eingetragen

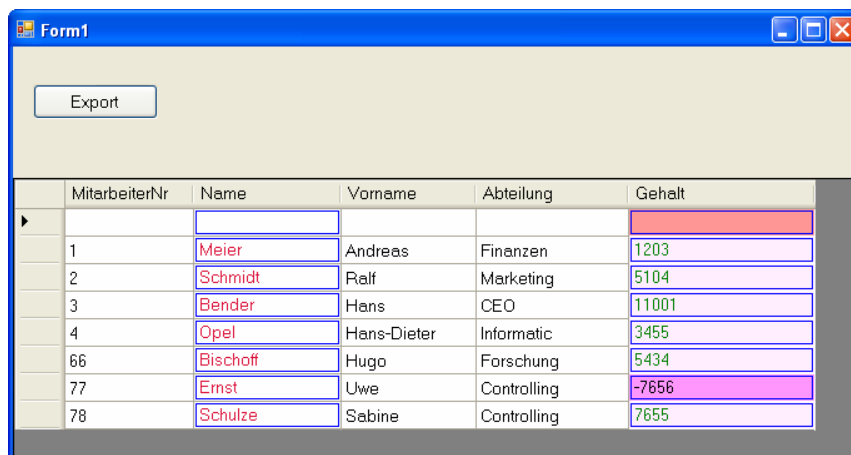
Alternative:

<http://www.componentone.com/>

<http://www.codeproject.com/KB/grid/csharpgridcontrol.aspx>



Selbstdefinierte Formatierung



Multi Dokument Interface (MDI)

Ein MDI-Fenster zeigt Daten in mehreren eigenständigen Fenster an.

Eigenschaften:

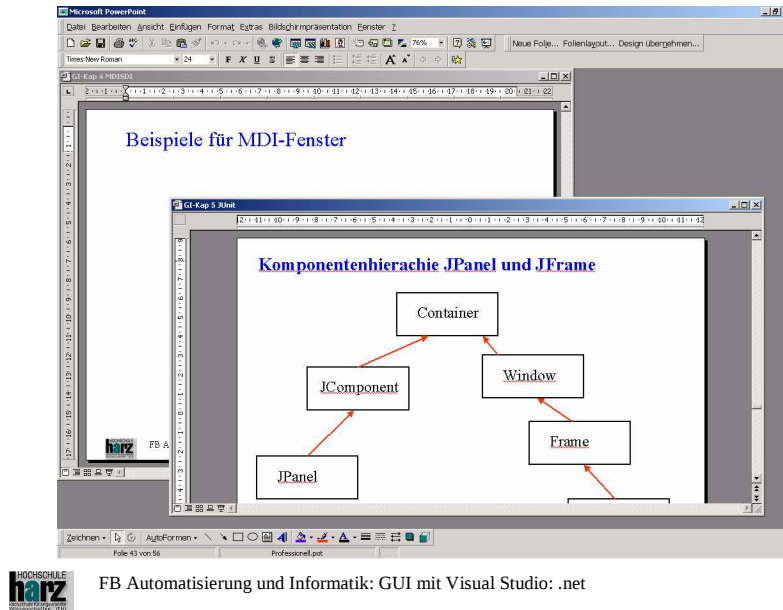
- Menüleiste
- Schalterleisten
- Statuszeile
- Kein Schalter „Ok“ oder „Abbruch“
- Wechselnde Menüleisten (Groupindex)
- Mehrere Fenster für eine Datei
- Komplex

Multi Dokument Interface (MDI)

Weitere Eigenschaften:

- Eigenständige Clientfenster (Klassen)
- Mainframe
- Menü für die Client-Fenster
- Letzte Dateien
- Fensterdarstellung
 - cascade, tile, arrange icons, next, previous
- Programminfo
- Hilfe
- Erzeugen Dialogfenster zur speziellen Eingabe

Beispiele für MDI-Fenster (Powerpoint)



FB Automatisierung und Informatik: GUI mit Visual Studio: .net

45

Allgemeine MDI- Regeln

- Die Menüs des Mainfensters enthalten nur die Inhalte für das MainFenster.
 - Menü Datei
 - Menü Bearbeiten
 - Menü Ansicht
 - Menü Fenster
 - Menü Hilfe
- Die Schalterleisten dienen sowohl für das Mainfenster als auch für die unterschiedlichsten Clientfenster.
 - Datei öffnen, speichern, Drucken
 - Zwischenablage

Menü Datei:

■ Neu	Create Client
■ Öffnen	Create Client
■ Speichern	Message to Client
■ Speichern unter	Message to Client
■ Drucken	Message to Client
■ Drucker einstellen	
■ Beenden	Message to all Clients

Menü Bearbeiten:

■ Rückgängig	Message to Client
■ Wiederholen	Message to Client
■ Ausschneiden	Message to Client
■ Kopieren	Message to Client
■ Einfügen	Message to Client
■ Alles markieren	Message to Client
■ Löschen	Message to Client
■ Suchen	Message to Client
■ Ersetzen	Message to Client

Menü Fenster:

- Vorheriges Fenster
- Nächstes Fenster
- Überlappen
- Nebeneinander
- Untereinander
- Symbole arrangieren

Multiple Document Interface

- **Ablauf**
- Erstellen eines neuen Projektes
- Löschen von form1 (rechte Maustaste: Delete)
- Menü Projekt: Eintrag: "Add new Item" (STRG+Shift+A)
- Auswahl Windows Form dann "MDI Parent Form"
- **DANN klappt es auch mit den Submenüs (siehe Seite 50)**
- WindowsState auf Maximized setzen (optional)
- **Eigenschaft IsMDIContainer auf true setzen**
- Menüleiste einfügen
- Nun die Hauptmenüs einfügen (Datei, Fenster, Hilfe), nur wenn manuell
 - &Datei MainFile
 - &Fenster MainWindow
 - &Hilfe MainHelp

Multiple Document Interface

- **Ablauf**
- Nun die weiteren Menüs einfügen (Datei)
 - &Neu MnNew
 - &Öffnen MnOpen
 - &Speichern MnSave
 - Speichern &unter MnSaveAs
 - &Drucken MnPrint
 - Schließen MnCloseClient
 - Schließen alle MnCloseAllClient
 - &Beenden MnClose



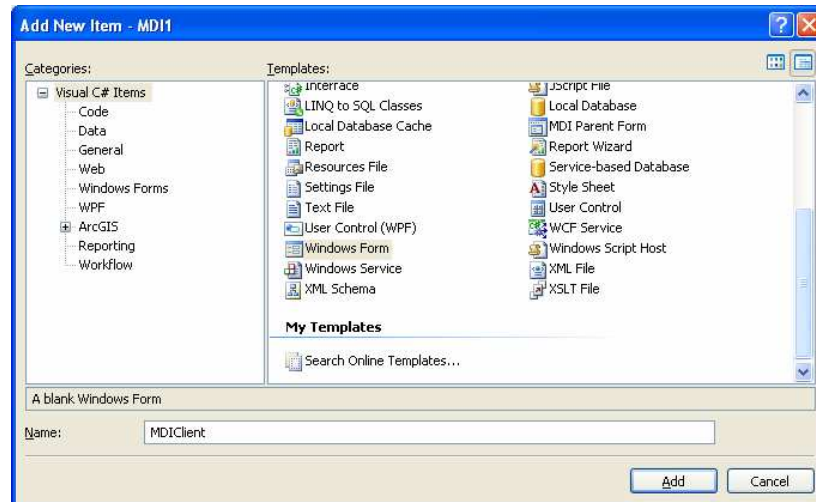
Multiple Document Interface

- **Ablauf**
- Nun die weiteren Menüs einfügen (Fenster)
 - Überlappend MnCascade
 - &Nebeneinander MnTileVertical
 - &Übereinander MnTileHorizontal
- **Eintragen der Fensterliste**
- Anklicken der Komponente "Menustrip" unten
- Eigenschaftsdialog aufrufen
- In der Liste [MdiWindowListItem](#) das Fenstermenü auswählen
- Nun die weiteren Menüs einfügen (Hilfe)
 - Programminfo MnInfo



Multiple Document Interface

■ MDI Clientfenster: Name MDIClient



Multiple Document Interface

■ Eintragen der Events

■ Neu

- o MDIClient childForm = new MDIClient();
- o childForm.MdiParent = this;
- o childForm.Text = "Window " + childFormNumber++;
- o childForm.Show();

■ Fenster

- o LayoutMdi(MdiLayout.Cascade);
- o LayoutMdi(MdiLayout.TileVertical);
- o LayoutMdi(MdiLayout.TileHorizontal);
- o LayoutMdi(MdiLayout.ArrangeIcons);

Main Container

```
private void MnNew_Click(object sender, EventArgs e)
{
    MDIClient childForm = new MDIClient();
    childForm.MdiParent = this;
    childForm.Text = "Window " + childFormNumber++;
    childForm.Show();
}
```



Clientfenster

```
private void MnOpen_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();
    openFileDialog.InitialDirectory = Environment.CurrentDirectory;
    openFileDialog.Filter = "Text Files (*.txt)|*.txt|All Files (*.*)|*.*";
    if (openFileDialog.ShowDialog(this) == DialogResult.OK)
    {
        string FileName = openFileDialog.FileName;
        MDIClient childForm = new MDIClient(FileName); // 2. Konstruktor
        childForm.MdiParent = this; // setzen ISMDIContainer
        childForm.Text = "Window " + childFormNumber++;
        childForm.Show();
    }
}
```



```

public MDIClient()
{
    InitializeComponent();
    this.sFilename = "";
}
public MDIClient(string sFilename)
{
    InitializeComponent();
    this.sFilename = sFilename;
}
private void MDIClient_Load(object sender, EventArgs e)
{
    Editor.Dock = DockStyle.Fill;
    if (!sFilename.Equals(""))
    {
        loadFile(sFilename);
    }
}

```

Clientfenster

using System.IO;

```

private void loadFile(string sFilename)
{
    string sStr;
    FileStream oFile = new FileStream(sFilename, FileMode.Open);
    StreamReader inStream = new StreamReader(oFile);
    sStr = inStream.ReadToEnd();
    Editor.Text = sStr;
}

```

Multiple Document Interface

■ Eintragen

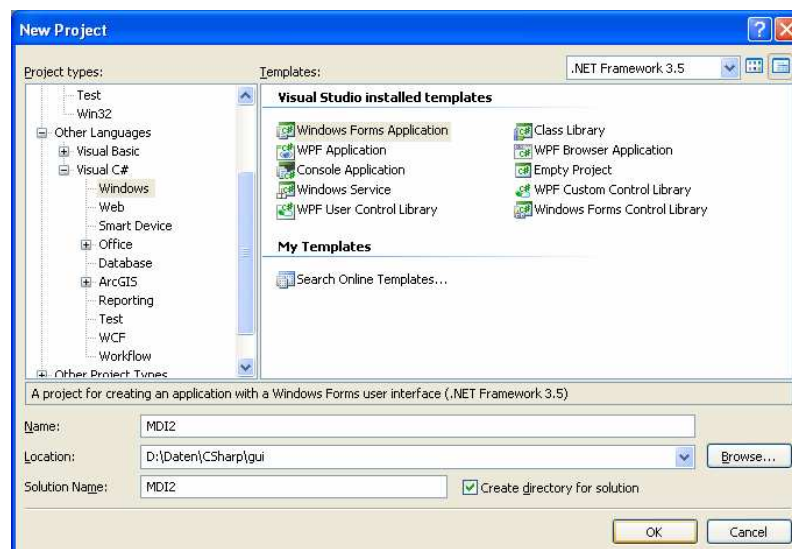
- o &Bearbeiten

MainEdit

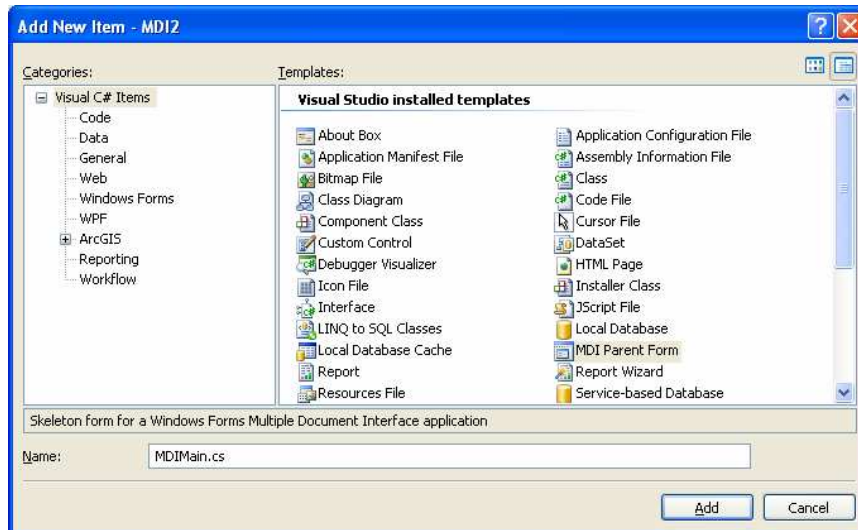
■ Eintragen der Einträge im Bearbeiten

- | | | |
|-------------------|--------|-------------|
| o Rückgängig | Strg+Z | MnUndo |
| o Ausschneiden | Strg+X | MnCut |
| o Kopieren | Strg+C | MnCopy |
| o Einfügen | Strg+V | MnPaste |
| o Löschen | | MnDelete |
| o Alles markieren | Strg+A | MnSelectAll |

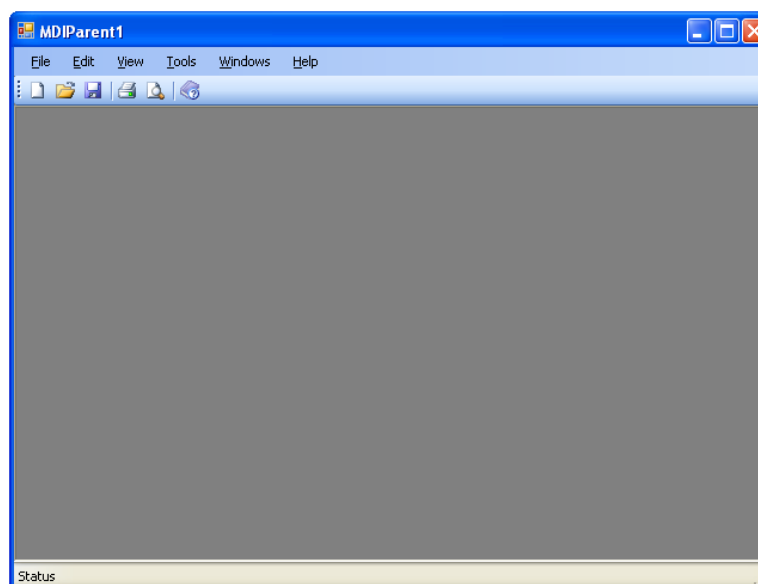
MDI-Fenster automatisch erstellen: MDI2



Menü Projekt, Eintrag "Add Component"



MDI Parent Form



Multiple Document Interface

- Menü Projekt
- Add Class
- Auswahl Code
- Auswahl Interface
- Name IMDIClient

```
using System;
using System.Collections.Generic;
using System.Text;
```

```
namespace MDI2
```

```
{
```

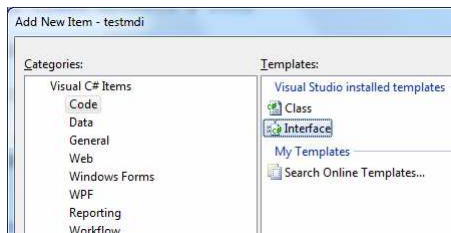
```
    interface IMDIClient
```

```
    {
```

```
        void save();
```

```
    }
```

```
}
```



```
public partial class MDIClient : Form, IMDIClient
```

```
{
```

```
    private void MnSave_Click(object sender, EventArgs e)
```

```
    {
```

```
        if (!sFilename.Equals("")) saveFile(); // ohne SaveAs
```

```
    }
```

```
    public void save()
```

```
    {
```

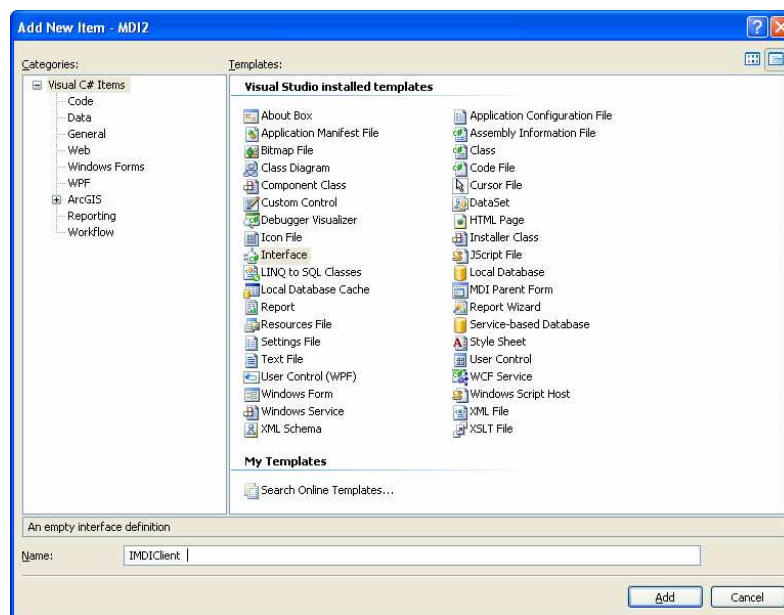
```
        // hier Abfrage auf SaveAs
```

```
        saveFile();
```

```
    }
```

MDI Container: Save Event

```
private void saveToolStripMenuItem_Click(object sender, EventArgs e)
{
    Form f = this.ActiveMdiChild;
    if (f == null)
    {
        MessageBox.Show("null");
        return;
    }
    IMDIClient fm = (IMDIClient)f;
    fm.save();
}
```



Tree / Baum

■ Eigenschaften

- Darstellung hierarchischer Elemente
- Unterscheidung Verzweigung / Blätter
- Symbole (Geschlossen / aufgeklappt)
- Häufig in Verbindung mit einer ListView / MDI-Fenster
- Erweiterte Eigenschaften in Knoten (aufklappbar, Node-Id))

■ Beispiele:

- Dateisystem (Explorer)
- Darstellen der Objektstruktur einer Datei
- Darstellen aller Teiler einer Zahl

Tree: Beispiel



Tree: 1. Beispiel

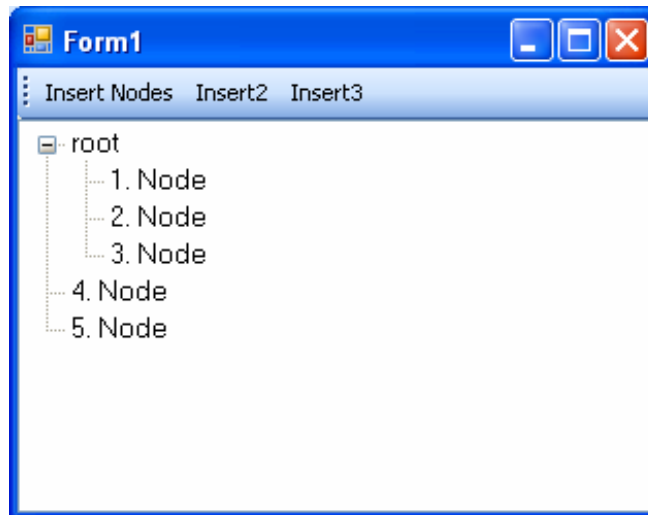
```
treeView1.Nodes.Clear();
```

```
treeView1.Nodes.Add(new TreeNode("1. Node"));  
treeView1.Nodes.Add(new TreeNode("1. Node"));  
treeView1.Nodes.Add(new TreeNode("2. Node"));  
treeView1.Nodes.Add(new TreeNode("3. Node"));
```

Tree: 1. Beispiel

```
TreeNode node;  
TreeNode root;  
treeView1.Nodes.Clear();  
root = new TreeNode("root");  
treeView1.Nodes.Add(root);  
node=new TreeNode("1. Node");  
    root.Nodes.Add(node);  
node=new TreeNode("2. Node");  
    root.Nodes.Add(node);  
node=new TreeNode("3. Node");  
    root.Nodes.Add(node);  
treeView1.Nodes.Add(new TreeNode("4. Node"));  
treeView1.Nodes.Add(new TreeNode("5. Node"));
```

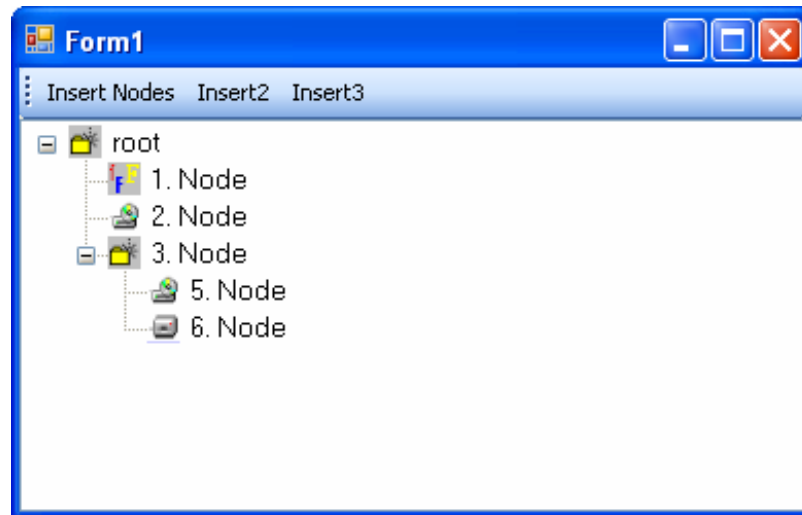
Tree: Beispiel



Tree: 3. Beispiel

```
TreeNode node, node2;
TreeNode root;
treeView1.BeginUpdate(); // Unterdrücken des Neuzeichnens
treeView1.ImageList = imageList1; // an ImageList binden
treeView1.Nodes.Clear();
root = new TreeNode("root",1,2); // Symbole (markiert und nicht markiert)
treeView1.Nodes.Add(root);
node = new TreeNode("1. Node",2,3);
root.Nodes.Add(node);
node = new TreeNode("2. Node",3,4);
root.Nodes.Add(node);
node2 = new TreeNode("3. Node",1,3);
treeView1.EndUpdate(); // Neu Zeichnen
root.ExpandAll();
```

Tree: 3. Beispiel



Klasse als "Pointer" an jedem TreeNode

class CStudent

```
{  
    public string name;  
    public int matrnr;  
  
    public CStudent(string name, int matrnr)  
    {  
        this.name = name;  
        this.matrnr = matrnr;  
    }  
}
```

```

private void BnInsert_Click(object sender, EventArgs e)
{
    TreeNode node, node1, node2, node3;
    TreeNode root;
    treeView1.BeginUpdate(); // Unterdrücken des Neuzeichnens
    treeView1.ImageList = imageList1; // an ImageList binden
    treeView1.Nodes.Clear();
    root = new TreeNode("root", 1, 2); // Symbole geschlossen offen
    treeView1.Nodes.Add(root);

    node = addNode(root, "1. Node", 12345,1,2); // Erzeugt Instanz CStudent
    node1 = addNode(root, "2. Node", 13345,2,3);
    node2 = addNode(root, "3. Node", 13345, 1, 4);
    node = addNode(node2, "4. Node", 43345, 3, 4);
    node3 = addNode(node2, "5. Node", 23345, 6, 1);
    treeView1.EndUpdate(); // Neu Zeichnen
    root.ExpandAll();
}

```



Klasse als "Pointer" an jedem TreeNode

```

private TreeNode addNode(TreeNode parent,
    string name, int matrnr, int sym1, int sym2)
{
    CStudent std;
    TreeNode node;
    std = new CStudent(name, matrnr);
    node = new TreeNode(name, sym1, sym2);
    node.Tag = std;
    parent.Nodes.Add(node);
    return node;
}

```



```

private void treeView1_AfterSelect(object sender, TreeViewEventArgs e) {
    TreeNode node = treeView1.SelectedNode;
    if (node != null)
    {
        CStudent std;
        string sStr1, sStr2, sStr3;
        std = (CStudent) node.Tag;
        sStr1 = "Pfad : " + node.FullPath;
        if (std != null) // wenn root angeklickt !
        {
            sStr2 = "Name : " + std.name;
            sStr3 = "Matrnr: " + std.matrnr.ToString();
            MessageBox.Show(sStr1 + "\r\n" + sStr2 + "\r\n" + sStr3, "Tree Click");
        }
        else
        {
            MessageBox.Show(sStr1, "Tree Click (root click)");
        }
    }
}

```

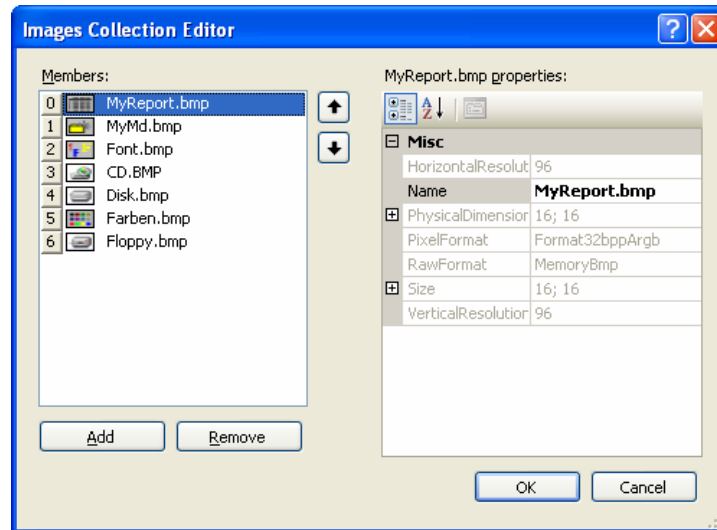


Tree: Aufgabe

- Erstellen eines neuen Projektes
- Einfügen eines Trees
- Doppelklick, form_load erstellen
 - treeView1.Dock = DockStyle.Fill;
- Imagelist einfügen
- Anklicken
- Property-Fenster
- Eintrag "Collection"
- Einfügen der Symbole (Zip-Datei, siehe Homepage)
- Toolstrip einfügen
- Schalter einfügen
- Doppelklick: **Bestimme alle Teiler einer Zahl (1 bis 50)**



Tree: Aufgabe



Tree: Aufgabe

```
private TreeNode addNode(TreeNode parent,
    string name, int sym1, int sym2)
{
    TreeNode node = new TreeNode(name, sym1, sym2);
    parent.Nodes.Add(node);
    return node;
}

private void insertTeiler(TreeNode parent, int nr)    {
    int i, j;
    TreeNode node1, node2;
    ...
}
```

Tree: Aufgabe

```
private void BnInsert_Click(object sender, EventArgs e)
{
    TreeNode root;
    int i,j;
    treeView1.BeginUpdate(); // Unterdrücken des Neuzeichnens
    treeView1.ImageList = imageList1; // an ImageList binden
    treeView1.Nodes.Clear();
    root = new TreeNode("root", 1, 2); // Symbole geschlossen offen
    treeView1.Nodes.Add(root);
    // Aktion
    treeView1.EndUpdate(); // Neu Zeichnen
    root.ExpandAll();
}
```



Test auf ganzzahlige Werte

```
static bool IsNumeric(string sStr) {
    int erg;
    bool isNumber;
    isNumber = Int32.TryParse(Convert.ToString(sStr),
        System.Globalization.NumberStyles.Any,
        System.Globalization.NumberFormatInfo.InvariantInfo,
        out erg);
    return isNumber;
}
}
```



C# Sprache: Literatur

Softwareentwicklung mit C#

Hanspeter Mössenböck
dpunkt.Verlag, ISBN 3-89864-406-5

Visual C+ 2005

Günter Born, Benjamin Born
Entwickler.press, ISBN 978-3-939084-40-2

Handbuch der .NET-Programmierung

Rolf Wenger
Microsoft Press Deutschland, 1664 Seiten,
ISBN: 3866454198



FB Automatisierung und Informatik: GUI mit Visual Studio: .net

83

C# Sprache: Literatur

Visual C# 2008 von Andreas Kuehnel

Das umfassende Handbuch
Buch: Visual C# 2008

Visual C# 2008

geb., mit DVD
1.366 S., 49,90 Euro
Galileo Computing
ISBN 978-3-8362-1172-7

Datenbank-Programmierung mit Visual C# 2008

Walter Doberanz, Thomas Gewinnus
Microsoft Press
ISBN 978-3-86645-421-7



FB Automatisierung und Informatik: GUI mit Visual Studio: .net

84

Links

- <http://www.guidetocsharp.de>
- <http://msdn.microsoft.com/de-de/library/kx37x362.aspx>
- <http://www.componentone.com/>
- <http://www.devexpress.com/Index.xml>
- <http://www.devexpress.com/Products/NET/Controls/WinForms/Grid/>