zenon guida
Controls
v.7.11
Tutti i diritti riservati.

Tutti i diritti riservati la distribuzione e la copia - indifferentemente dal metodo - può essere consentita esclusivamente dalla ditta COPA-DATA. I dati tecnici servono solo per la descrizione del prodotto e non rappresentano in alcun modo parti legali. Modifiche - anche sotto aspetti tecnici sono a noi riservate.
Indice

1. Benvenuti nell’help COPA-DATA................................................................................................................................5

2. Controls .......................................................................................................................................................... 5

3. Generale .......................................................................................................................................................... 6
   3.1 Accesso a API zenon .................................................................................................................................... 6
   3.2 Metodi......................................................................................................................................................... 8
   3.2.1 CanUseVariables .................................................................................................................................. 9
   3.2.2 MaxVariables ....................................................................................................................................... 9
   3.2.3 VariableTypes ...................................................................................................................................... 9
   3.2.4 zeronExit ............................................................................................................................................. 10
   3.2.5 zeronExitEd ......................................................................................................................................... 10
   3.2.6 zeronInit ............................................................................................................................................. 10
   3.2.7 zeronInitEd .......................................................................................................................................... 11

4. ActiveX ......................................................................................................................................................... 11
   4.1 Sviluppare elementi Active X .................................................................................................................. 12
   4.1.1 Metodi................................................................................................................................................... 12
   4.2 Esempio LatchedSwitch (C++) ................................................................................................................ 15
   4.2.1 Interfaccia .......................................................................................................................................... 15
   4.2.2 Control............................................................................................................................................... 16
   4.2.3 Metodi................................................................................................................................................... 19
   4.2.4 Gestione e visualizzazione .................................................................................................................. 21
   4.2.5 L’interfaccia di zenon .......................................................................................................................... 24
   4.3 Esempio CD_SliderCtrl (C++) ................................................................................................................... 24
   4.3.1 Interfaccia .......................................................................................................................................... 24
   4.3.2 Control............................................................................................................................................... 25
   4.3.3 Metodi................................................................................................................................................... 27
   4.3.4 Gestione e visualizzazione .................................................................................................................. 30
   4.3.5 L’interfaccia di zenon .......................................................................................................................... 32
   4.4 Esempio .NET Control come ActiveX (C#) ............................................................................................... 32
   4.4.1 creare Windows Form Control ............................................................................................................ 32
   4.4.2 trasformare .NET User Control in Dual Control .................................................................................. 36
5. **.NET User Controls** ................................................................. 46
   5.1 Differenze nell’uso di .NET Control in Control Container e ActiveX ........................................... 47
   5.2 Esempio di .NET Control Container ......................................................................................... 47
      5.2.1 Generale .................................................................................................................................. 47
      5.2.2 Creare .Net User Control ........................................................................................................ 50
      5.2.3 aggiungeremo CD_DotNetControlContainer e .NET User Control ........................................ 59
      5.2.4 Accesso all’User Control tramite VSTA oppure VBA ............................................................ 64
   5.3 Esempio .NET Control come ActiveX (C#) .............................................................................. 68
      5.3.1 creare Windows Form Control .............................................................................................. 68
      5.3.2 trasformare .NET User Control in Dual Control ................................................................ 72
      5.3.3 lavorare nell’Editor tramite VBA con ActiveX .................................................................... 77
      5.3.4 Collegare variabili di zenon con il .NET User Control ......................................................... 78

6. **Elemento WPF** .................................................................................. 82
   6.1 Basi: ......................................................................................................................... 83
      6.1.1 WPF nella visualizzazione di processo .................................................................................. 83
      6.1.2 Trasmissione valori di zenon a WPF .................................................................................. 85
      6.1.3 Oggetti referenziati ............................................................................................................. 86
      6.1.4 Assegnazione di oggetto zenon a WPF Content ................................................................ 87
      6.1.5 Workflows ........................................................................................................................ 88
   6.2 Indicazioni per i designer ................................................................................................. 89
      6.2.1 workflow con Microsoft Expression Blend ............................................................................ 90
      6.2.2 workflow con Adobe Illustrator .......................................................................................... 94
   6.3 Engineering in zenon ................................................................................................. 103
      6.3.1 Crea elemento WPF ......................................................................................................... 104
      6.3.2 CDWPF-file collection (file collection) .............................................................................. 105
      6.3.3 Configurazione del collegamento ....................................................................................... 105
      6.3.4 Validità di file XAML ......................................................................................................... 121
      6.3.5 Elementi predefiniti .......................................................................................................... 123
      6.3.6 Esempio: Integrazione di WPF in zenon .......................................................................... 136
      6.3.7 Trattamento errori .......................................................................................................... 158
1. Benvenuti nell'help COPA-DATA

GUIDA GENERALE

Nel caso in cui non abbiate trovato delle informazioni che cercavate o se avete dei consigli relativi al completamento di questo capitolo dell'help, mandate una Mail a documentation@copadata.com (mailto:documentation@copadata.com).

SUPPORTO ALLA PROGETTAZIONE

Se avete delle domande concernenti progetti concreti, potete rivolgervi per E-Mail al support@copadata.com (mailto:support@copadata.com).

LICENZE E MODULI

Nel caso in cui doveste constatare che avete bisogno di altri moduli o licenze, rivolgetevi ai nostri dipendenti all'indirizzo sales@copadata.com (mailto:sales@copadata.com).

2. Controls

In zenon possono essere implementati dei Control propri. A tal scopo sono a disposizione:

- .NET User Controls (A pagina: 46) (sul tema implementazione in zenon, vedi anche .NET Controls nella parte della guida concernente le Immagini.)
- ActiveX (A pagina: 11) (sul tema implementazione in zenon vedi anche ActiveX nella parte della guida concernente le Immagini.)
3. Generale

Control per zenon possono essere realizzati mediante ActiveX, .NET e WPF. Tramite VBA/VSTA si può accedere alla zenon API.

3.1 Accesso a API zenon

Sotto zenon un ActiveX Control può essere esteso con delle funzioni specifiche, per poter accedere a API zenon.

ACCESSO A API ZENON

- selezionate nelle Project References con Add References... la zenon-RT libreria oggetti
- inserite le funzioni estese nel codice classe del control.
FUNZIONI ESTESE ACTIVEX ZENON

// Viene aperto al momento dell’avvio del Control nel runtime di zenon.
public bool zenonInit(zenon.Element dispElement)...

// Viene aperto al momento della distruzione del Control nel runtime di zenon.
public bool zenonExit()...

// Supporta il Control Variablenverknüpfungen (collegamentivariabili).
public short CanUseVariables()...

// Tipi di dato supportati da Com Control.
public short VariableTypes()...

// Numero massimo delle variabili che possono essere collegate con il Control.
public short MaxVariables()...

ESEMPIO

Il COM Object di una variabile di zenon viene salvato temporaneamente in un Member per poterlo poi accedere nel Paint Event del Control:

zenon.Variable m_cVal = null;
public bool zenonInit(zenon.Element dispElement)
{
    if (dispElement.CountVariable > 0) {
        try {
            m_cVal = dispElement.ItemVariable(0);
            if (m_cVal != null) {
                object obRead = m_cVal.get_Value((object)-1);
                UserText = obRead.ToString();
            }
        } catch { }
    }
    return true;
}

public bool zenonExit()
{
    try {
        if (m_cVal != null) {
            System.Runtime.InteropServices.Marshal.FinalReleaseComObject(m_cVal);
            m_cVal = null;
        }
    }
catch { } 
return true;

public short CanUseVariables()
{
    return 1; // le variabili sono supportate
}

public short VariableTypes()
{
    return short.MaxValue; // tutti i tipi di dato vengono supportati
}

public short MaxVariables()
{
    return 1; // al massimo una variabile deve essere collegata con il Control
}

private void SamplesControl_Paint(object sender, PaintEventArgs e)
{
    // zenon Variables has changed
    try {
        if (m_cVal != null) {
            object obRead = m_cVal.get_Value((object)-1);
            UserText = obRead.ToString();
        }
    } catch { }

3.2 Metodi

ActiveX Control e .NET-Control che usano variabili di zenon, necessitano di determinati metodi.
3.2.1 CanUseVariables

Prototipo: `short CanUseVariables();`

Questo metodo fornisce come risultato 1 oppure 0

<table>
<thead>
<tr>
<th>Value</th>
<th>Descrizione</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>Il Control può usare delle variabili di zenon. È possibile indicare per l’elemento dinamico (tramite il pulsante <code>Variabile</code>) solamente delle variabili di zenon dei tipi indicati tramite il metodo <code>VariableTypes</code> (A pagina: 9) e nella quantità indicata nel metodo <code>MaxVariables</code> (A pagina: 9).</td>
</tr>
<tr>
<td>0:</td>
<td>Il Control non può usare variabili di zenon oppure non possiede questo metodo. Possono essere indicate variabili di tutti i tipi senza limite di numero. Esse possono essere usate nel runtime, ma solo tramite VBA.</td>
</tr>
</tbody>
</table>

3.2.2 MaxVariables

Prototipo: `short MaxVariables();`

In questa sede si stabilisce il numero delle variabili che possono essere selezionate dalla lista di variabili. Se il risultato è 1, nella lista variabili viene disattivata la possibilità di multiselect. Il sistema emette un avvertimento nel caso in cui si tenti di selezionare più variabili contemporaneamente.

3.2.3 VariableTypes

Prototipo: `short VariableTypes();`

Il valore restituito da questo metodo viene usato come maschera per i tipi di dato utilizzabili nella lista delle variabili. Il valore è una connessione `AND` dei seguenti valori (definiti in `zenon32/dy_type.h`):
3.2.4 zenonExit

Prototipo: boolean zenonExit();

Questo metodo viene chiamato in causa dal runtime di zenon al momento della chiusura dell’ActiveX Control.

In questo caso dovrebbero essere resi disponibili tutti i Dispatch Pointer su variabili.

3.2.5 zenonExitEd

Corrisponde a zenonExit (A pagina: 10) e viene eseguito nell’editor al momento della chiusura dell’ActiveX.

In tal modo è possibile reagire anche nell’editor a modifiche nell’ActiveX, o a cambiamenti di valore.

Informazioni: è disponibile attualmente solo per ActiveX.

3.2.6 zenonInit

Prototipo: boolean zenonInit(IDispatch* dispElement);
Con l’aiuto di questo metodo, viene consegnato (nel runtime) all’ActiveX Control una lancetta sulla Dispatch Interface dell’elemento dinamico. Grazie a questa lancetta si può passare alle variabili di zenon collegate all’elemento dinamico.

La successione delle variabili consegnate la definite nella Configurazione dell’elemento ActiveX con i pulsanti In giù oppure In su. Il dialogo Immissione elemento si apre quando cliccate due volte sull’elemento ActivX, oppure se selezionate nelle proprietà dell’elemento al nodo Visualizzazione la proprietà Impostazioni ActiveX.

3.2.7 zenonInitEd

Corrisponde a zenonInit (A pagina: 10) e viene eseguito nell’editor al momento dell’apertura dell’ActiveX (due clic sull’ActiveX).

Informazioni: è disponibile attualmente solo per ActiveX.

4. ActiveX

Con ActiveX si può estendere la funzionalità del Runtime e dell’Editor di zenon.

In questa guida trovate quanto segue:

- Sviluppare elementi Active X (A pagina: 12)
- Esempio LatchedSwitch (C++) (A pagina: 15)
- Esempio CD_SliderCtrl (C++) (A pagina: 24)
- Esempio .NET Control come ActiveX (C#) (A pagina: 32)

Le informazioni sull’elemento dinamico ActiveX le trovate in questa guida nella parte dedicata alle immagini e in particolare nel capitolo ActiveX.

**ACTIVEX PER WINDOWS CE**

Se un ActiveX Control deve funzionare sotto Windows CE, il modello apartement deve essere settato su Threading. Se è settato su Free, il Control non funzionerà nel Runtime di zenon.
4.1 Sviluppare elementi Active X

L’elemento dinamico ActiveX in zenon può trasferire delle variabili all’ActiveX Control senza che sia necessario usare la via più lunga attraverso VBA per gestire il Control.

Il Control può definire autonomamente quante variabili di zenon possono essere utilizzate e di che tipo esse possono essere. Inoltre le proprietà del Control possono essere definite tramite l’elemento dinamico.

A tal scopo, l’interfaccia (Dispatch-Interface) del Control deve supportare una serie di determinati metodi (A pagina: 12).

4.1.1 Metodi

Ogni ActiveX-Control che può usare delle variabili di zenon deve contenere i seguenti metodi.

- CanUseVariables (A pagina: 9)
- MaxVariables (A pagina: 9)
- VariableTypes (A pagina: 9)
- zeronExit (A pagina: 10)
- zeronExitEd (A pagina: 10)
- zeronInit (A pagina: 10)
- zeronInitEd (A pagina: 11)

Non ha alcuna importanza quale Dispatch ID hanno i metodi nell’interfaccia. All’apertura dei metodi, in zenon viene tratto dall’interfaccia l’ID corretto.

CanUseVariables

Prototipo: short CanUseVariables();

Questo metodo fornisce come risultato 1 oppure 0.
### Valor e Descrizione

<table>
<thead>
<tr>
<th>Valor e</th>
<th>Descrizione</th>
</tr>
</thead>
</table>
| 1:      | Il Control può usare delle variabili di zenon.  
È possibile indicare per l’elemento dinamico (tramite il pulsante **Variabile**) solamente delle variabili di zenon dei tipi indicati tramite il metodo **VariableTypes** (A pagina: 9) e nella quantità indicata nel metodo **MaxVariables** (A pagina: 9). |
| 0:      | Il Control non può usare variabili di zenon oppure non possiede questo metodo.  
Possono essere indicate variabili di tutti i tipi senza limite di numero. Esse possono essere usate nel runtime, ma solo tramite VBA. |

### MaxVariables

Prototipo: `short MaxVariables();`

In questa sede si stabilisce il numero delle variabili che possono essere selezionate dalla lista di variabili.

Se il risultato è 1, nella lista variabili viene disattivata la possibilità di multiselect. Il sistema emette un avvertimento nel caso in cui si tenti di selezionare più variabili contemporaneamente.

### VariableTypes

Prototipo: `short VariableTypes();`

Il valore restituito da questo metodo viene usato come maschera per i tipi di dato utilizzabili nella lista delle variabili. Il valore è una connessione **AND** dei seguenti valori (definiti in `zenon32/dy_type.h`):
### Valore 1 | Valore 2 | Corrispondenza
--- | --- | ---
WORD | 0x0001 | Posizione 0
BYTE | 0x0002 | Posizione 1
BIT | 0x0004 | Posizione 2
DWORD | 0x0008 | Posizione 3
FLOAT | 0x0010 | Posizione 4
DFLOAT | 0x0020 | Posizione 5
STRING | 0x0040 | Posizione 6
EIN_AUSGABE | 0x8000 | Posizione 15

**zenonExit**

Prototipo: `boolean zenonExit();`

Questo metodo viene chiamato in causa dal runtime di zenon al momento della chiusura dell’ActiveX Control.

In questo caso dovrebbero essere resi disponibili tutti i Dispatch Pointer su variabili.

**zenonExitEd**

Corrisponde a zenonExit (A pagina: 10) e viene eseguito nell’editor al momento della chiusura dell’ActiveX.

In tal modo è possibile reagire anche nell’editor a modifiche nell’ActiveX, o a cambiamenti di valore.

**Informazioni:** è disponibile attualmente solo per ActiveX.

**zenonInit**

Prototipo: `boolean zenonInit(IDispatch* dispElement);`

Con l’aiuto di questo metodo, viene consegnato (nel runtime) all’ActiveX Control una lancetta sulla Dispatch Interface dell’elemento dinamico. Grazie a questa lancetta si può passare alle variabili di zenon collegate all’elemento dinamico.
La successione delle variabili consegnate la definite nella Configurazione dell’elemento ActiveX con i pulsanti \textit{In giù} oppure \textit{In su}.

Il dialogo \textit{Immissione elemento} si apre quando cliccate due volte sull’elemento ActivX, oppure se selezionate nelle proprietà dell’elemento al nodo \textit{Visualizzazione} la proprietà \textit{Impostazioni ActiveX}.

\textbf{zenonInitEd}

Corrisponde a \textit{zenonInit} (A pagina: 10) e viene eseguito nell’editor al momento dell’apertura dell’ActiveX (due clic sull’ActiveX).

\textbf{Informazioni}: è disponibile attualmente solo per ActiveX.

\section*{4.2 Esempio LatchedSwitch (C++)}

Di seguito verrà descritto a titolo di esempio un ActiveX Control che realizza un interruttore bloccabile tramite due variabili Bit. La prima variabile rappresenta l’interruttore, la seconda il blocco. La variabile dell’interruttore può essere modificata tramite l’ActiveX Control solo nel caso in cui il valore della variabile di blocco sia pari a 0.

Lo stato dell’elemento viene visualizzato tramite 4 Bitmaps che possono essere selezionati nell’editor di zenon facendo uso del dialogo properties del Control.

\subsection*{4.2.1 Interfaccia}

Il Control LatchedSwitch possiede la seguente Dispatch Interface:

\begin{verbatim}
[ uuid(EB207159-D7C9-11D3-B019-080009FBEAA2),
helpstring(Dispatch interface for LatchedSwitch Control), hidden ]
dispinterface _DLatchedSwitch
{
    properties:
    // NOTE - ClassWizard will maintain method information here.
    // Use extreme caution when editing this section.
    //((AFX_ODL_PROP(CLatchedSwitchCtrl)
    [id(1)] boolean SollwertDirekt;
    [id(2)] IPictureDisp* SwitchOn; // container for the bitmaps
    [id(3)] IPictureDisp* SwitchOff;
    [id(4)] IPictureDisp* LatchedOn;
}
\end{verbatim}
ActiveX

methods:
// NOTE - ClassWizard will maintain method information here.
// Use extreme caution when editing this section.
//{{AFX_ODL_METHOD(CLatchedSwitchCtrl)
//}}AFX_ODL_METHOD
[id(6)] short CanUseVariables();
[id(7)] short VariableTypes();
[id(8)] short MaxVariables();
[id(9)] boolean zenonInit(IDispatch* dispElement);
[id(10)] boolean zenonExit();
[id(DISPID_ABOUTBOX)] void AboutBox();
};


4.2.2 Control

L’implementazione del Control avviene tramite la classe CLatchedSwitchCtrl. Questa classe possiede come Member gli oggetti CPictureHolder per l’accettazione dei Bitmaps. Inoltre sono compresi anche tre DispatchDriver per l’elemento dinamico e le variabili:

class CLatchedSwitchCtrl : public COleControl
{

DECLARE_DYNCREATE(CLatchedSwitchCtrl)

// Constructor
public:
CLatchedSwitchCtrl();

// Overrides

// ClassWizard generated virtual function overrides
//}}AFX_VIRTUAL(CLatchedSwitchCtrl)
public:
virtual void OnDraw(CDC* pdc, const CRect& rcBounds, const CRect& rcInvalid);
virtual void DoPropExchange(CPropExchange* pPX);
virtual void OnResetState();
virtual DWORD GetControlFlags();
//}}AFX_VIRTUAL

// Implementation
protected:
~CLatchedSwitchCtrl();

DECLARE_OLECREATE_EX(CLatchedSwitchCtrl) // Class factory and guid
DECLARE_OLETYPELIB(CLatchedSwitchCtrl) // GetTypeInfo
DECLARE_PROPPAGEIDS(CLatchedSwitchCtrl) // Property page IDs
DECLARE_OLECTLTYPE(CLatchedSwitchCtrl) // Type name and misc status

// Message maps

//}}AFX_MSG(CLatchedSwitchCtrl)
afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
//}}AFX_MSG
DECLARE_MESSAGE_MAP()

// Dispatch maps

//}}AFX_DISPATCH(CLatchedSwitchCtrl)
BOOL m_sollwertDirekt;
afx_msg void OnSollwertDirektChanged();
afx_msg LPPICTUREDISP GetSwitchOn();
afx_msg void SetSwitchOn(LPPICTUREDISP newValue);
afx_msg LPPICTUREDISP GetSwitchOff();
afx_msg void SetSwitchOff(LPPICTUREDISP newValue);
afx_msg LPPICTUREDISP GetLatchedOn();
afx_msg void SetLatchedOn(LPPICTUREDISP newValue);
afx_msg LPPICTUREDISP GetLatchedOff();
afx_msg void SetLatchedOff(LPPICTUREDISP newValue);
afx_msg short CanUseVariables();
afx_msg short VariableTypes();
afx_msg short MaxVariables();
afx_msg BOOL zenonInit(LPDISPATCH dispElement);
afx_msg BOOL zenonExit();
//}}AFX_DISPATCH
CPictureHolder m_SwitchOn;
CPictureHolder m_SwitchOff;
CPictureHolder m_LatchedOn;
CPictureHolder m_LatchedOff;

DECLARE_DISPATCH_MAP()

afx_msg void AboutBox();

// Event maps

//{{AFX_EVENT(CLatchedSwitchCtrl)
//}}AFX_EVENT
DECLARE_EVENT_MAP()

double VariantToDouble(const VARIANT FAR *v);
void VariantToCString(CString *c, const VARIANT FAR *v);
BOOL IsVariantString(const VARIANT FAR *v);
BOOL IsVariantValue(const VARIANT FAR *v);

// Dispatch and event IDs
public:

CString szVariable[2];
IElement m_dElement;
IVariable m_dLatchVar, m_dSwitchVar;

enum {
//}}AFX_DISP_ID(CLatchedSwitchCtrl)
dispidSollwertDirekt = 1L,
dispidSwitchOn = 2L,
dispidSwitchOff = 3L,
dispidLatchedOn = 4L,
4.2.3 Metodi

Vengono usati i seguenti metodi:

- CanUseVariables (A pagina: 19)
- VariableTypes (A pagina: 19)
- MaxVariables (A pagina: 20)
- zenonInit (A pagina: 20)
- zenonExit (A pagina: 21)

**CanUseVariables**

Questo metodo fornisce 1 come risultato; possono essere dunque utilizzate delle variabili di zenon.

```cpp
short CLatchSwitchCtrl::CanUseVariables()
{
    return 1;
}
```

**VariableTypes**

Il Control può lavorare solamente con variabili Bit, perciò viene restituito 0x0004.

```cpp
short CLatchSwitchCtrl::VariableTypes()
```
MaxVariables

Possono essere usate due variabili, perciò risulta 2.

short CLatchedSwitchCtrl::MaxVariables()
{
    return 2; // 2 Variablen
}

zenonInit

In questo metodo il sistema si va a prendere mediante il Dispatchpointer dell’elemento dinamico i Dispatchdriver delle variabili. Mediante questo Pointer il sistema va a prendere e setta i valori delle variabili con un clic e disegnando il control.

BOOL CLatchedSwitchCtrl::zenonInit(LPDISPATCH dispElement)
{
    m_dElement = IElement(dispElement);

    if (m_dElement.GetCountVariable() >= 2)
    {
        short iIndex = 0;
        m_dSwitchVar = IVariable(m_dElement.ItemVariable(COleVariant(iIndex)));
        m_dLatchVar = IVariable(m_dElement.ItemVariable(COleVariant(++iIndex)));
    }
    return TRUE;
zenonExit

Questo metodo rende di nuovo disponibili i Dispatchdriver.

BOOL CLatchedSwitchCtrl::zenonExit()
{
    m_dElement.ReleaseDispatch();
    m_dSwitchVar.ReleaseDispatch();
    m_dLatchVar.ReleaseDispatch();
    return TRUE;
}

4.2.4 Gestione e visualizzazione

Imposta valore

Un valore impostabile viene settato cliccando una volta sul Control con il tasto sinistro del mouse.

Nel caso in cui m_iSollwertDirekt sia pari a 0, si apre un dialogo per la selezione del valore impostabile; altrimenti viene invertito il valore attuale della variabile di conversione.

Se il valore della variabile di blocco è 1, viene eseguito solamente un MessageBeep; non può essere settato nessun valore impostabile tramite il Control.

void CLatchedSwitchCtrl::OnLButtonDown(UINT nFlags, CPoint point)
{
    CRect rcBounds;
    GetClientRect(&rcBounds);

    COleVariant coeValue((BYTE)TRUE);
    BOOL bLatch = (BOOL)VariantToDouble((LPVARIANT)&m_dLatchVar.GetValue());
    BOOL bSwitch = (BOOL)VariantToDouble((LPVARIANT)&m_dSwitchVar.GetValue());

    if (bLatch) // Bloccato!!
MessageBeep(MB_ICONEXCLAMATION);
else
{

if (m_sollwertDirekt)
{

bSwitch = !bSwitch;
}
else
{

CSollwertDlg dlg;
dlg.m_iSollwert = bSwitch ? 1 : 0;
if (dlg.DoModal() == IDOK)
{

if (dlg.m_iSollwert == 2) // Toggle

bSwitch = !bSwitch;
else

bSwitch = (BOOL)dlg.m_iSollwert;
}
}
coleValue = (double)bSwitch;
m_dSwitchVar.SetValue(coleValue);
}
COleControl::OnLButtonDown(nFlags, point);
}

Disegno

Al momento del disegno del Control il sistema va a prendersi i valori della variabili tramite il Dispatchdriver di queste ultime; in corrispondenza di ciò viene disegnato una delle 4 grafiche indicate. Quando cambia il valore di una variabile, il Control viene actualizzato facendo uso della routine OnDraw.

void CLatchedSwitchCtrl::OnDraw(CDC* pdc, const CRect& rcBounds, const CRect& rcInvalid)
{

CRect rcBitmap = rcBounds;
rcBitmap.NormalizeRect();

if (!m_dElement)
{
    m_SwitchOn.Render(pdc, &rcBounds, &rcBounds);
    return;
}

BOOL bVal1 = 0, bVal2 = 0;
VARIANT vRes;
if (m_dSwitchVar) // Variabili disponibili?
{
    vRes = m_dSwitchVar.GetValue();
bVal1 = (BOOL)VariantToDouble(&vRes);
}
if (m_dSwitchVar) // Variabile disponibile?
{
    vRes = m_dLatchVar.GetValue();
bVal2 = (BOOL)VariantToDouble(&vRes);
}

if (bVal1 && bVal2)
    m_SwitchOn.Render(pdc, rcBitmap, rcBitmap);
else if (!bVal1 && bVal2)
    m_SwitchOff.Render(pdc, rcBitmap, rcBitmap);
else if (bVal1 && !bVal2)
    m_LatchedOn.Render(pdc, rcBitmap, rcBitmap);
else
    m_LatchedOff.Render(pdc, rcBitmap, rcBitmap);
4.2.5 L’interfaccia di zenon

Per poter utilizzare il Dispatch Interface di zenon al fine di settare i valori impostabili, bisogna anzitutto impostare per le variabili e l’elemento delle classi che sono derivate da COleDispatchDriver. Il modo più semplice per impostare queste classi è farlo tramite il Class Wizard dell’ambiente di sviluppo (pulsante Add Class, selezione di From a type library, scegliere zenrt32.tlb).

Nel caso del nostro Control, esse sono le classi IElement e IVariable. Esse sono definite in zenrt32.h e zenrt32.cpp.

4.3 Esempio CD_SliderCtrl (C++)

Il seguente esempio descrive un ActiveX Control che è uguale al Windows SliderCtrl. Questa componente può essere collegata ad una variabile di zenon. L’utente può modificare il valore impostabile di una variabile tramite lo slider. Nel caso in cui la variabile sia modificata tramite un altro elemento dinamico, viene actualizzato lo slider.

4.3.1 Interfaccia

Il Control CD_SliderCtrl possiede la seguente Dispatch Interface:

```c++
[ uuid(5CD1B01D-015E-11D4-A1DF-080009FD837F),
  helpstring(Dispatch interface for CD_SliderCtrl Control), hidden
]
dispinterface _DCD_SliderCtrl
{
properties: //*** Proprietà del Control

[id(1)] short TickRaster;
[id(2)] boolean ShowVertical;
[id(3)] short LineSize;

methods: //*** Metodi del Control (per zenon-ActiveX)
```
4.3.2 Control

L’implementazione del Control avviene tramite la classe CD_SliderCtrlCtrl. Questa classe possiede come member un CSliderCtrl comune per Windows, con cui il Control viene editato in una sottoclasse. Le interfacce IVariable e IElement contengono interfacce di zenon che si è dovuto integrare. Esse sono derivate da COleDispatchDriver.

class CCD_SliderCtrlCtrl : public COleControl
{

DECLARE_DYNCREATE(CCD_SliderCtrlCtrl)

private: //*** Member-Variables

    BOOL m_bInitialized;
    BOOL m_bShowVertical;
    BOOL m_bTicksBoth;
    long m_nRangeStart;
    long m_nRangeEnd;
    long m_nTickOrientation;
    IVariable m_interfaceVariable;
    IElement m_interfaceElement;
    CSliderCtrl m_wndSliderCtrl;

public:

    CCD_SliderCtrlCtrl();

    //{{AFX_VIRTUAL(CCD_SliderCtrlCtrl)

    public:

    

};
virtual void OnDraw (CDC* pdc, const CRect& rcBounds, const CRect& rcInvalid);
virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
virtual void DoPropExchange (CPropExchange* pPX);
virtual void OnResetState ();
//}}AFX_VIRTUAL

protected:

~CCD_SliderCtrlCtrl();
//*** Metodo per la conversione di Variant
double VariantToDouble(const VARIANT FAR *vValue);

DECLARE_OLECREATE_EX(CCD_SliderCtrlCtrl) // Class factory and guid
DECLARE_OLETYPELIB (CCD_SliderCtrlCtrl) // GetTypeInfo
DECLARE_PROPPAGEIDS (CCD_SliderCtrlCtrl) // Property page IDs
DECLARE_OLECTLTYPE (CCD_SliderCtrlCtrl) // Type name and misc status

//*** Metodi per la funzionalità di SliderCtrl
BOOL IsSubclassedControl();
LRESULT OnOcmCommand (WPARAM wParam, LPARAM lParam);

//}}AFX_MSG(CCD_SliderCtrlCtrl)
afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
afx_msg void HScroll(UINT nSBCode, UINT nPos);
afx_msg void VScroll(UINT nSBCode, UINT nPos);
afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
//}}AFX_MSG
DECLARE_MESSAGE_MAP()

//}}AFX_DISPATCH(CCD_SliderCtrlCtrl)
afx_msg BOOL GetTickOnBothSides();
afx_msg void SetTickOnBothSides (short nNewValue);
afx_msg BOOL GetShowVertical();
afx_msg void SetShowVertical(BOOL bNewValue);
4.3.3 Metodi

Vengono usati i seguenti metodi:

- CanUseVariables (A pagina: 28)
- VariableTypes (A pagina: 28)
- MaxVariables (A pagina: 28)
- zenonInit (A pagina: 29)
- zenonExit (A pagina: 30)

**CanUseVariables**

Questo metodo fornisce 1 come risultato; possono essere dunque utilizzate delle variabili di zenon.

```cpp
short CCD_SliderCtrlCtrl::CanUseVariables()
{
    return 1;
}
```

**VariableTypes**

Il Control può lavorare con parola, byte, parola doppia e variabili float. Una lista dei possibili tipi di dato la trovate nella descrizione (A pagina: 9) generale di questo metodo.

```cpp
short CCD_SliderCtrlCtrl::VariableTypes()
{
    return 0x0001 | // Wort
    0x0002 | // Byte
    0x0008 | // D-Wort
    0x0010 | // Float
    0x0020; // D-Float
}
```

**MaxVariables**

Con questo Control può essere collegata solo una variabile.

```cpp
short CCD_SliderCtrlCtrl::MaxVariables()
{
```
Il parametro dispElement contiene l’interfaccia per l’elemento dinamico. Tramite questo elemento viene determinata la variabile collegata di zenon. Se questa è valida, viene settato il settore di SliderCtrls. Inoltre vengono settate le impostazioni per la visualizzazione (numero dei ticks, ...). Se non è collegata nessuna variabile, viene fissato un settore di funzionamento da 0 a 0. In tal modo, SliderCtrl non può essere modificato. Tramite la variabile m_bInitialized si stabilisce che i valori impostabili da adesso possono essere settati.

```cpp
BOOL CCD_SliderCtrlCtrl::zenonInit(LPDISPATCH dispElement)
{
    //*** Stabilisci la variabile tramite l’elemento zenon

    m_interfaceElement = IElement(pElementInterface);
    if (m_interfaceElement.GetCountVariable() > 0) {

        short nIndex = 0;
        m_interfaceVariable = IVariable
            (m_interfaceElement.ItemVariable(COleVariant(nIndex)));
    }

    //*** Inizializza il settore dello Slider-Ctrl
    if (m_interfaceVariable) {

        //*** Stabilisci settore di funzionamento
        m_nRangeStart = (long) VariantToDouble(&m_interfaceVariable.GetRangeMin());
        m_nRangeEnd = (long) VariantToDouble(&m_interfaceVariable.GetRangeMax());
        m_wndSliderCtrl.SetRange(m_nRangeStart, m_nRangeEnd, TRUE);

        //*** Stabilisci tick secondari
        m_wndSliderCtrl.SetTicFreq(m_nTickCount);
        m_wndSliderCtrl.SetPageSize(m_nTickCount);
        m_wndSliderCtrl.SetLineSize(m_nLineSize);
    } else {

        m_wndSliderCtrl.SetRange(0,0,TRUE);
        return FALSE;
    }
}

return 1; // 1 Variablen
m_bInitialized = TRUE;
return TRUE;
}

zenonExit

In questo metodo vengono di nuovo rese disponibili le interfacce di zenon.

BOOL CCD_SliderCtrlCtrl::zenonExit()
{

m_interfaceElement.ReleaseDispatch();
m_interfaceVariable.ReleaseDispatch();
return TRUE;
}

4.3.4 Gestione e visualizzazione

Disegno

Tramite DoSuperclassPaint viene disegnato lo SliderCtrl (visto che si tratta di a subclassed Control). Se al momento del disegno lo slider viene spostato, la variabile m_bInitialized assume il valore FALSE. In tal modo ci si assicura che il valore possa essere modificato. Generalmente il valore della variabile viene determinato ed esso viene rappresentato mediante il metodo SetPos dello SliderCtrl.

void CCD_SliderCtrlCtrl::OnDraw(CDC* pdc, const CRect& rcBounds, const CRect& rcInvalid) {

    // *** Attualizza la visualizzazione
    DoSuperclassPaint(pdc, rcBounds);
    if (m_interfaceVariable && m_bInitialized) {

        COleVariant cValue(m_interfaceVariable.GetValue());
        int nValue = (int) VariantToDouble(&cValue.Detach());
        m_wndSliderCtrl.SetPos(nValue);
    }
}
Imposta valore

Nel metodo `LButtonDown` la variabile `m_bInitialized` viene settata a `FALSE` e nel `LButtonUp` viene riportata a `TRUE`. In tal modo si assicura che il valore possa essere modificato. Altrimenti si dovrebbe aprire la `OnDraw` e si dovrebbe visualizzare il valore originario.

```cpp
void CCD_SliderCtrlCtrl::OnLButtonDown(UINT nFlags, CPoint point)
{
    m_bInitialized = FALSE;
    COleControl::OnLButtonDown(nFlags, point);
}

void CCD_SliderCtrlCtrl::OnLButtonUp(UINT nFlags, CPoint point)
{
    m_bInitialized = TRUE;
    COleControl::OnLButtonUp(nFlags, point);
}
```

Un valore impostabile viene provocato da una modifica dello slider. Nei metodi `HScroll` oppure `VScroll` viene settato il valore impostabile (in dipendenza del fatto che si tratti di uno slider orizzontale oppure verticale).

```cpp
void CCD_SliderCtrlCtrl::HScroll(UINT nSBCode, UINT nPos)
{
    switch (nSBCode) {

    case TB_LINEUP:
    case TB_PAGEUP:
    case TB_LINEDOWN:
    case TB_PAGEDOWN:
    case TB_THUMBTRACK:
    case TB_THUMBPOSITION:\

        //*** settare valore impostabile senza dialogo ?
        int nValue = m_wndSliderCtrl.GetPos();
        COleVariant cValue((short) nValue,VT_I2);
        m_interfaceVariable.SetValue(cValue);  
```
4.3.5 L’interfaccia di zenon

Per poter utilizzare il Dispatch Interface di zenon al fine di settare i valori impostabili, bisogna anzitutto impostare per le variabili e l’elemento delle classi che sono derivate da COleDispatchDriver. Il modo più semplice per impostare queste classi è farlo tramite il Class Wizard dell’ambiente di sviluppo (pulsante Add Class, selezione di From a type library, scegliere zenrt32.tlb).

Nel caso del nostro Control, esse sono le classi IElement e IVariable. Esse sono definite in zenrt32.h e zenrt32.cpp.

4.4 Esempio .NET Control come ActiveX (C#)

Il seguente esempio descrive un .NET Control che viene eseguito come ActiveX Control in zenon.

La creazione e integrazione avviene in quattro passi:

1. creare Windows Form Control (A pagina: 32)
2. trasformare .NET User Control in Dual Control (A pagina: 36)
3. lavorare nell’Editor tramite VBA con ActiveX (A pagina: 41)
4. Collegare variabili di zenon con il .NET User Control (A pagina: 42)

4.4.1 creare Windows Form Control

Per creare un Windows Form Control:
1. Avviare il Visual Studio 2008 e creare un nuovo progetto Windows Form Control Library:

![Visual Studio New Project dialog]

2. Rinominare il Control di default dando il nome desiderato.
   Nel nostro esempio: **SamplesControl.cs**.

![Visual Studio Solution Explorer]

---

**Note:**

- Assicurarsi di utilizzare il formato corretto per il file di controllo.
- Verificare che il progetto sia configurato correttamente per l'uso di ActiveX.
- Testare il controllo nel proprio ambiente per verificare la correttezza delle funzionalità.

---

**Riferimenti:**

- [Guida di Visual Studio](https://docs.microsoft.com/it-it/visualstudio)
- [API di ActiveX](https://msdn.microsoft.com/it-it/library/)
- [Esercizi Practice](https://www.example.com)
3. Aprire il Control Designer e aggiungete i Control desiderati, nel nostro caso una casella di testo (Text-box):
   Nel nostro esempio: La property visibile esternamente, "UserText" con **get** e accesso **set** che contiene il testo della casella testo:

![Code Designer](image)

5. compilate il progetto.

Il Windows Forms Control può essere adesso usato in altri progetti Windows Forms.
4.4.2 trasformare .NET User Control in Dual Control

Per trasformare .NET in un Dual Control, si deve anzitutto procedere ad attivare l’interfaccia COM per ActiveX.
1. Aprire il progetto e attivate alla voce Build-impostazioni la proprietà Register for COM interop:

2. Aprire il file AssemblyInfo.cs e

   - settate l’attributo ComVisible su true
   - aggiungete l’attributo ClassInterface

   ![AssemblyInfo.cs](image-url)
[assembly: ClassInterface(ClassInterfaceType.AutoDual)]
3. Apri il Codedesigner tramite View Code e aggiungete gli attributi ActiveX e gli inserimenti using necessari. Create un nuovo GUID per l’attributo GUID servendovi del menu Tools/Create GUID:

![Image of Visual Studio UI with code]

4. Per fare in modo che il Control sia selezionabile anche come Control superfici ActiveX, si devono ancora aggiungere nella classe control le seguenti funzioni:
   - RegisterClass
• UnregisterClass

Poi il controllo può essere registrato nella Registry.

5. Ricompilate il progetto.

Il Windows Form Control è adesso utilizzabile anche con ActiveX ed è stato registrato automaticamente con il Rebuild. Un ulteriore file Typelib zenOnDotNetControl.tlb è stato creato nel percorso output.

6. Per usare il control su un altro calcolatore:
   a) copiate il file DLL e il file TBL sul calcolatore di destinazione
   b) registrate il file tramite la riga comandi con:
      
      %windir%\Microsoft.NET\Framework\v2.0.50727\regasm.exe zenOnDotNetControl.dll /tlb:zenOnDotNetControl.tlb
7. Aggiungete l’esteso Windows Form Control come ActiveX Control nell’Editor in zenon:

4.4.3 lavorare nell’Editor tramite VBA con ActiveX

Per poter accedere nell’Editor zenon alle proprietà del Control:

1. create nell’Editor di zenon, per la precisione nel nodo *Interfacce di programmazione/Macro-VBA* una nuova macro *Init* con la denominazione *Init_ActiveX*.

Con questa macro è poi possibile accedere a tutte le properties esterne mediante *obElem.AktiveX...*
2. Assegnate questa macro all’ActiveX Control facendo uso della proprietà Macro VBA/Init dell’elemento ActiveX.

**ESEMPIO MACRO INIT**

```vbnet
Public Sub Init_ActiveX(obElem As Element)
    obElem.AktiveX.Usertext = "Setta la stringa sul Control"
End Sub
```

4.4.4 Collegare variabili di zenon con il .NET User Control

Sotto zenon avete la possibilità di estendere un ActiveX Control integrandolo con delle funzioni specifiche, per poter accedere a API zenon.

**METODI NECESSARI**

- public bool zenOnInit (A pagina: 44) (Viene aperto al momento dell’inizializzazione del Control nel Runtime di zenon.)
- public bool zenOnInitED (A pagina: 45) (Viene usato nell’Editor.)
- public bool zenOnExit() (A pagina: 45) (Viene aperto al momento della distruzione del Control nel Runtime di zenon.)
- public bool zenOnExitED() (A pagina: 45) (Viene usato nell’Editor.)
- public short CanUseVariables() (A pagina: 45) (Supporta il collegamento di variabili.)
- public short VariableTypes() (A pagina: 45) (Tipi di dato supportati da Control)
- public MaxVariables() (A pagina: 46) (Numero massimo delle variabili che possono essere collegate con il Control.)

AGGIUNGI REFERENZE

1. In Microsoft Visual Studio selezionate alla voce Add References la biblioteca oggetti-Runtime di zenon, per poter accedere nel Control a zenon API.

   ![Add Reference](image)

2. Aggiungete le funzioni estese nel class-code del control per accedere all’intera zenon API.
Nel nostro esempio, il COM Object di una variabile di zenon viene salvato temporaneamente in Member un per potervi poi accedere nel Paint Event del Control:

```csharp
public bool zenOnInit(zenOn.Element dispElement)
```

Con l’aiuto di questo metodo, viene consegnato (nel runtime) all’ActiveX Control una lancetta sulla Dispatch Interface dell’elemento dinamico. Grazie a questa lancetta si può passare alle variabili di zenon collegate all’elemento dinamico.

La successione delle variabili può essere impostata nel dialogo "Immissione elemento" con il pulsante in basso o in alto. Il dialogo Introduzione elemento si apre se:

- cliccate due volte sull’elemento ActiveX
- oppure se selezionate Proprietà nel menu contestuale, oppure
- selezionate nella finestra delle proprietà nel nodo Visualizzazione la proprietà Impostazioni ActiveX
public bool zenOnInitED(zenOn.Element dispElement)

Corrisponde a public bool zenOnInit (A pagina: 44) e viene eseguito al momento dell’apertura dell’ActiveX nell’Editor (doppio clic su ActiveX)

public bool zenOnExit()

Questo metodo viene chiamato in causa dal runtime di zenon al momento della chiusura dell’ActiveX Control. In questo caso dovrebbero essere resi disponibili tutti i Dispatch Pointer su variabili.

public bool zenOnExitED()

Corrisponde a public bool zenOnExit() (A pagina: 45) e viene eseguito nell’editor al momento della chiusura dell’ActiveX. In tal modo si può reagire anche nell’Editor a modifiche, come, ad esempio, cambio di valori.

public short CanUseVariables()

Questo metodo fornisce come risultato 1 se il Control può usare delle variabili di zenon e 0 in caso contrario.

- 1: È possibile indicare per l’elemento dinamico (tramite il pulsante Variabili) solamente delle variabili di zenon dei tipi indicati tramite il metodo VariableTypes e nella quantità indicata nel metodo MaxVariables.
- 0: Nel caso in cui CanUseVariables dia 0, oppure nel caso in cui il Control non possieda tale metodo, possono essere indicate variabili di ogni tipo e senza limitazioni di numero. Esse possono essere usate nel runtime, ma solo tramite VBA.

public short VariableTypes()

Il valore restituito da questo metodo viene usato come maschera per i tipi di dato utilizzabili nella lista delle variabili. Il valore è una connessione AND dei seguenti valori (definiti in zenon32/dy_type.h):
### Public MaxVariables()

In questa sede si stabilisce il numero delle variabili che possono essere selezionate dalla lista di variabili.

1: nella lista variabili viene disattivata la possibilità di multiselect. Il sistema emette un avvertimento nel caso in cui si tenti di selezionare più variabili contemporaneamente.

### 5. .NET User Controls

Con i .NET Control si può estendere la funzionalità del Runtime e dell’Editor di zenon.

In questa guida trovate quanto segue:

- Differenze fra Control Container e ActiveX (A pagina: 47)
- Esempio di .NET Control Container (A pagina: 47)
- Esempio .NET Control come ActiveX (C#) (A pagina: 32)

Le informazioni concernenti l’uso di .NET Controls in ActiveX le trovate nella guida Immagini nel capitolo .NET Controls.

<table>
<thead>
<tr>
<th>Parametri</th>
<th>Valore</th>
<th>Descrizione</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD</td>
<td>0x0001</td>
<td>corrisponde a posizione 0</td>
</tr>
<tr>
<td>BYTE</td>
<td>0x0002</td>
<td>corrisponde a posizione 1</td>
</tr>
<tr>
<td>BIT</td>
<td>0x0004</td>
<td>corrisponde a posizione 2</td>
</tr>
<tr>
<td>DWORD</td>
<td>0x0008</td>
<td>corrisponde a posizione 3</td>
</tr>
<tr>
<td>FLOAT</td>
<td>0x0010</td>
<td>corrisponde a posizione 4</td>
</tr>
<tr>
<td>DFLOAT</td>
<td>0x0020</td>
<td>corrisponde a posizione 5</td>
</tr>
<tr>
<td>STRING</td>
<td>0x0040</td>
<td>corrisponde a posizione 6</td>
</tr>
<tr>
<td>IN_OUTPUT</td>
<td>0x8000</td>
<td>corrisponde a posizione 15</td>
</tr>
</tbody>
</table>
5.1 Differenze nell’uso di .NET Control in Control Container e ActiveX

Un .Net User Control può:

- essere implementato tramite CD_DotNetControlContainer Control direttamente nell’elemento ActiveX di zenon
- essere usato come ActiveX Control ed essere implementato direttamente nell’elemento ActiveX di zenon

Le differenze fra Container Control e ActiveX Control sono soprattutto:

<table>
<thead>
<tr>
<th>CD_DotNetControlContainer Control</th>
<th>ActiveX Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non deve essere registrato sul calcolatore.</td>
<td>Deve essere registrato come ActiveX sul calcolatore (regsvr32).</td>
</tr>
<tr>
<td>Se ci sono delle modifiche nel Control, il DLL deve essere cambiato.</td>
<td>Se ci sono delle modifiche nel Control, il TLB deve essere nuovamente registrato.</td>
</tr>
<tr>
<td>Accesso tramite VBA e VSTA possibile solamente tramite metodi CD_DotNetControlContainer.</td>
<td>Accesso tramite VBA e VSTA molto semplice.</td>
</tr>
</tbody>
</table>

5.2 Esempio di .NET Control Container

In questo tutorial si descrive come creare in Visual Studio 2010 un semplice .NET User Control (linguaggio di programmazione c#) e come implementarlo poi con l’aiuto di zenon CD_DotNetControlContainer Control come ActiveX in un elemento Active X di zenon.

5.2.1 Generale

Il CD_DotNetControlContainer serve da wrapper fra un User Control e l’elemento ActiveX di zenon. Tutti i metodi usati nel seguente esempio, nonché tutti i Public Methods e Properties vengono trasmessi tramite il CD_DotNetControlContainer dall’User Control all’ActiveX e possono essere usati da zenon, anche in VBA e VSTA.
Se nell’User Control si referenzia alla Programming Interface di zenon, si può accedere direttamente agli oggetti di zenon.

![Diagram](image)

Nel seguente esempio faremo quanto segue:

- **imposteremo un .NET User Control (A pagina: 50)**
- **aggiungeremo CD_DotNetControlContainer e .NET User Control (A pagina: 59)**
- **consentiremo l’accesso all’User Control tramite VSTA (VBA) (A pagina: 64)**

```csharp
public bool zenOnInit(zenOn.Element dispElement)
```

Con l’aiuto di questo metodo, viene consegnato (nel runtime) all’ActiveX Control una lancetta sulla Dispatch Interface dell’elemento dinamico. Grazie a questa lancetta si può passare alle variabili di zenon collegate all’elemento dinamico.

La successione delle variabili può essere impostata nel dialogo "Immissione elemento" con il pulsante **in basso 0 in alto**. Il dialogo Introduzione elemento si apre se:

- **cliccate due volte sull’elemento ActiveX**
- **oppure se selezionate Proprietà nel menu contestuale, oppure**
- **selezionate nella finestra delle proprietà nel nodo Visualizzazione la proprietà Impostazioni ActiveX**
public bool zenOnExit()

Questo metodo viene chiamato in causa dal runtime di zenon al momento della chiusura dell’ActiveX Control. In questo caso dovrebbero essere resi disponibili tutti i Dispatch Pointer su variabili.

public short CanUseVariables()

Questo metodo fornisce come risultato 1 se il Control può usare delle variabili di zenon e 0 0 in caso contrario

- 1: È possibile indicare per l’elemento dinamico (tramite il pulsante Variabili) solamente delle varibili di zenon dei tipi indicati tramite il metodo VariableTypes e nella quantità indicata nel metodo MaxVariables.
- 0: Nel caso in cui CanUseVariables dia 0, oppure nel caso in cui il Control non possieda tale metodo, possono essere indicate variabili di ogni tipo e senza limitazioni di numero. Esse possoo essere usate nel runtime, ma solo tramite VBA.

public short VariableTypes()

Il valore restituito da questo metodo viene usato come maschera per i tipi di dato utilizzabili nella lista delle variabili. Il valore è una connessione AND dei seguenti valori (definiti in zenon32/dy_type.h):

<table>
<thead>
<tr>
<th>Parametri</th>
<th>Valore</th>
<th>Descrizione</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD</td>
<td>0x0001</td>
<td>corrisponde a posizione 0</td>
</tr>
<tr>
<td>BYTE</td>
<td>0x0002</td>
<td>corrisponde a posizione 1</td>
</tr>
<tr>
<td>BIT</td>
<td>0x0004</td>
<td>corrisponde a posizione 2</td>
</tr>
<tr>
<td>DWORD</td>
<td>0x0008</td>
<td>corrisponde a posizione 3</td>
</tr>
<tr>
<td>FLOAT</td>
<td>0x0010</td>
<td>corrisponde a posizione 4</td>
</tr>
<tr>
<td>DFLOAT</td>
<td>0x0020</td>
<td>corrisponde a posizione 5</td>
</tr>
<tr>
<td>STRING</td>
<td>0x0040</td>
<td>corrisponde a posizione 6</td>
</tr>
<tr>
<td>IN_OUTPUT</td>
<td>0x8000</td>
<td>corrisponde a posizione 15</td>
</tr>
</tbody>
</table>
public MaxVariables()

In questa sede si stabilisce il numero delle variabili che possono essere selezionate dalla lista di variabili.

1: nella lista variabili viene disattivata la possibilità di multiselect. Il sistema emette un avvertimento nel caso in cui si tenti di selezionare più variabili contemporaneamente.

5.2.2 Creare .Net User Control

L’User Control è un semplice control che consente di settare un nuovo valore tramite un campo di inserimento (Textbox). Dopo che si è premuto il pulsante, il valore viene scritto sulla variabile desiderata di zenon.

Un’altra funzione ha il compito di riconoscere se si è verificato un cambio nel valore della variabile in zenon e di visualizzare il nuovo valore nel Control.

FASI DELLA PROCEDURA

1. In primo luogo procedete alla creazione di un nuovo progetto in VS; a tal scopo usate il tipo di progetto „Windows Forms Control Library“

**Importante:** impostate Framework a 3.5!

3. Prossimo passo: create l’User Control. A tal scopo avete bisogno di due textbox per inserimento ed emissione, nonché di un pulsante per settare un nuovo valore sulla variabile di zenon. Denominate:
   - la prima textbox „txtGetZenonVariable“
   - la seconda textbox „txtSetZenonVariable“
   - il pulsante „btnSetZenonVariable“

4. Per poter accedere agli oggetti di zenon, avete bisogno di una referenza alla Programming Interface di zenon. A tale scopo:
   - cliccate sul nodo „References“ in Solution Explorer
   - aprite il menu contestuale
   - selezionate Add References...
   - andate alla voce com
   - selezionate la zenon programming interface library
A questo punto dovrebbe apparire la referenza "zenOn" nella lista delle referenze.

5. Il prossimo passo consiste nell’impostare nel programmcode di `zenon_CD_DotNetControlContainer.cs` una variabile globale del tipo `zenon.Variable`:

6. Questa variabile viene inizializzata tramite il public methode `zenOnInit`:

   e viene resa di nuovo disponibile attraverso il public methode `zenOnExit`: 
Nei seguenti metodi si stabilisce se le variabili e i tipi di dato di zenon possono essere usati e quante variabili possono essere consegnate:

```csharp
    /// <summary>
    /// This public method is needed to link zenon variables.
    /// It is the control.
    /// </summary>
    /// <returns>
    /// public short setZenonVariable()
    /// { return 1; // Only 1 variable is supported };
    /// </returns>
    /// <summary>
    /// This public method returns the type of.
    /// supported zenon Variables.
    /// </summary>
    /// <returns>
    /// short type
    /// public short returnVariableType()
    /// { return short.MinValue; // All Data Types supported }
    /// </returns>
    /// <summary>
    /// This public method returns the number of.
    /// supported zenon Variables.
    /// </summary>
    /// <returns>
    /// public short returnNumberOfVariables()
    /// { return 1; // Only 1 variable should list to the Control };
    /// </returns>
```

7. Il prossimo passo consiste nello stabilire nel Click-Event del pulsante `btnSetZenonVariable` che, cliccando sul pulsante stesso, il valore viene scritto nella textbox `txtSetZenonVariable` sul valore della variabile zenon e poi il contenuto della textbox viene cancellato.

```csharp
    /// <summary>
    /// This will be triggered by clicking the button. The new value will be set to the zenon Variable.
    /// </summary>
    /// <param name="sender">param</param>
    /// <param name="e">e</param>
    private void btnSetZenonVariable_Click(object sender, EventArgs e)
    {
        // Set value from textbox to the zenon variable
        _e.SetValue(_e.GetType().GetField("txtSetZenonVariable.text", BindingFlags.Public | BindingFlags.NonPublic | BindingFlags.Instance),
                    this.txtSetZenonVariable.Text + "string.empty")
    }
```

8. Per poter reagire alla modifica del valore della variabile, avete bisogno del Paint-Event del Control. Viene eseguito anche un trigger per il Paint-Event se cambia il valore della variabile zenon inizializzata e perciò non può essere usato per l’aggiornamento dei valori. Visto che le variabili che sono referenziate nell’elemento ActiveX zenon, sono automaticamente advised, si può in generale rinunciare al container `zenon.OnlineVariable` nel Control.

```
    /// <summary>
    /// This will be triggered by painting the user control or the value of the variable changed.
    /// After the value of the Variable changed the Control will be repainted and the new value.
    /// will be set to the Textbox.
    /// </summary>
    /// <param name="sender">param</param>
    /// <param name="e">e</param>
    private void zenonVariablePaint(object sender, PaintEventArgs e)
    {
        if (e.Cursor != null)
        {
            return;
        }
        else
        {
            zenonVariable.Text = _variable.value;
            return;
        }
```

**UNA VISIONE INSIEME DEL CODICE**

Qui ancora una volta l’intero codice in una visione d’insieme:
using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using zenOn;

namespace zenon_CD_DotNetControlContainer
{

    public partial class zenon_CD_DotNetControlContainer : UserControl
    {

        //This will be needed to get the zenon Variable Container
        zenOn.Variable m_cVal = null;

        public zenon_CD_DotNetControlContainer()
        {

            InitializeComponent();
        }

        /// <summary>
        /// This public Method will be called by the initialization of the control during the zenon Runtime.
        /// </summary>

        /// <param name="dispElement"/></param>
/// <returns></returns>

public bool zenOnInit(zenOn.Element dispElement)
{
    // Check if zenon Variables are added to the Control

    if (dispElement.CountVariable > 0)
    {
        try
        {
            // Take the first zenon Variable and added to the global Variable
            m_cVal = dispElement.ItemVariable(0);
        }
        catch
        {
        }
    }
    return true;
}

/// <summary>
/// This public Method will be calles by the release of the control during the zenon Runtime.
/// </summary>
/// <returns></returns>

public bool zenOnExit()
{
    try
    {
{  
    if (m_cVal != null)
    {
        //Release the zenon Variable (Com-Object)
        System.Runtime.InteropServices.Marshal.FinalReleaseComObject(m_cVal);
        m_cVal = null;
    }
}

catch {}  
return true;

/// <summary>
/// This public Method is needed to link zenon Variables  
/// to the control.  
/// </summary>
/// <returns></returns>
public short CanUseVariables()
{
    return 1;  // Only tis Variable is supported
}

/// <summary>
/// This public Method returns the Type of
/// supported zenon Variables
/// </summary>
/// <returns></returns
public short VariableTypes()
{
    return short.MaxValue; // all Data Types supported
}

/// <summary>
/// This public Method returns the number of
/// supported zenon Variables
/// </summary>
/// <returns></returns>
public short MaxVariables()
{
    return 1; // Only 1 Variable should linked to the Control
}

/// <summary>
/// This will be trigged by clicking the Button. The new Value will be set to the zenon Variable
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnSetZenonVariable_Click(object sender, EventArgs e)
{
    //Set Value from TextBox to the zenon Variable
    m_cVal.set_Value(0,txtSetZenonVariable.Text.ToString());
    this.txtSetZenonVariable.Text = string.Empty;
}
/// <summary>
/// This will be triggered by painting the User Control or the Value of the Variable changed.
/// After the value of the Variable changed the Control will be new painted and the new Value
/// will be set to the Textbox.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>

private void zenon_CD_DotNetControlContainer_Paint(object sender, PaintEventArgs e)
{
    if (m_cVal != null)
    {
        this.txtGetZenonVariable.Text = m_cVal.get_Value(0).ToString();
        return;
    }
    else
    {
        this.txtGetZenonVariable.Text = "Variable Value";
        return;
    }
}
CREA RELEASE

Alla fine create un release per poter implementare il DLL completo in zenon o in CD_DotNetControlContainer.

A tal scopo si deve assolutamente passare nei settings da Debug a Release.

5.2.3 aggiungeremo CD_DotNetControlContainer e .NET User Control

Per preparare il progetto zenon e aggiungere il CD_DotNetControlContainer e il .NET User Control, realizzate i passi seguenti:

1. Impostate una variabile interne del tipo Stringa e settatela sulla lunghezza stringa di 30.

3. Selezionate nel progetto l’elemento ActiveX e trascintelo nell’immagine zenon.
   - Si apre il dialogo Configurazione.
   - Selezionate il CD_DotNetControlContainer.Container Control.

4. Per inserire in modo scollegato il .NET User Control nel CD_DotNetControlContainer Control:
   - Cliccate sul pulsante Proprietà...
- viene aperto un nuovo dialogo

- cliccando sul pulsante Load selezionate il percorso della cartella di progetto, per esempio:
  C:\ProgramData\COPA-DATA\SQL\9888419d-251e-4595-b396-9be42367997c\FILES\zenon\custom\additional\zenon_CD_DotNetControlContainer.dll

Aggiungendo il DLL nella cartella additional, il Control viene spedito automaticamente ad un altro calcolatore al momento di copiare o caricare i file Runtime. Per questa ragione, il collegamento non va perduto.

Adesso dovrebbe essere visualizzato il .NET User Control.
Confermate il dialogo cliccando su **ok**.

5. L’ultimo passaggio consiste nel collegare una variabile con il Control facendo uso del pulsante **Variabili**.

La variabile selezionata viene collegata automaticamente mediante il metodo **public** `zenonInit` con la variabile definita globalmente (.NET UserControl). Il collegamento con il Control diviene però effettivo solamente dopo l’avvio del Runtime.
Poi dovete procedere a collegare la variabile interna con un elemento di testo.

6. Dopo l’avvio del Runtime, il Control è, in un primo momento, ancora vuoto.

Se viene scritto un valore nella seconda box di testo e poi confermato mediante il pulsante Set zenon Variable, il valore viene scritto sulla variabile di zenon. (L’evento btnSetZenonVariable_Click viene eseguito.)

Esso viene visualizzato anche nell’elemento di testo di zenon.
Se il valore viene modificato direttamente nell´elemento di testo di zenon,

![Image of a dialog box with options to modify a variable and an input field.]

esso verrà scritto tramite il **Paint** Event del .NET Control nella prima box di testo.

![Image of a .NET Control with a label and a text field.]

### 5.2.4 Accesso all´User Control tramite VSTA oppure VBA

In questo esempio si mostra l´accesso via VSTA. La procedura è la stessa con VBA.

1. Estendete il Control con un Label (**label**) e denominatelo **lblZenonInfo**. In questo Label deve essere visualizzato il valore di un´altra variabile di zenon. Il nuovo valore deve essere settato tramite una macro VSTA.

![Image of a .NET Control with a label and a text field.]

2. Estendete il Code della Property (Information) e aggiungete le proprietà `get` e `set` alla Property. Esse permettono di leggere e scrivere il testo del Label.

```csharp
public partial class zenon_CD_DotNetContainerControl : UserControl
{
    //this will be needed to get the zenon Variable container
    zenbozen.Variable my_var = null;

    public zenon_CD_DotNetContainerControl()
    { InitializationComponent();
    }

    public string Information
    {
        get { this.lblMainInfo.Text = value; }
        set { return this.lblMainInfo.Text; }
    }
}
```

3. Create un nuovo release per il nostro User Control e copiatelo nella cartella `additional` del progetto di zenon.

   **Non dimenticate:** chiudete prima l’Editor di zenon!

   Cancellate il vecchio DLL e riavviate l’Editor di zenon. Se il DLL si dovesse ancora trovare nella cartella, cancellatelo una seconda volta. Adesso potete importare il DLL modificato. Il `CD_DotNetContainerControl` e l’ActiveX vengono aggiornati automaticamente.

4. Cliccate nell’Editor di zenon su ActiveX e aprire la finestra delle proprietà.
Potete vedere adesso la nuova proprietà **Information** nella finestra di selezione del Control e potete anche settare un valore.

![Image of Control Selection Window]

Questo valore viene settato anche nel Control ("myInformation")

![Image of Control Window with Variable Value]

5. Per poter lavorare con il **CD_DotNetControlContainer** in VSTA oppure VBA, avete bisogno anzitutto della referenza al Control. Dopo che si è aperto VSTA per il progetto (**ProjectAddin**), la referenza di **CD_DotNetControlContainer** deve essere aggiunta.

![Image of Reference Add Window]

6. Con il seguente code può essere settato nuovamente il valore per la proprietà del Control `Information`:

```csharp
public void Macro_Test()
{
    try
    {
        zenon.Elements = this.getElements();
        // Create a Variable of Type CD_DotNetControlContainer-Container and get the zenon ActiveX Element
        // with a cast
        CD_DotNetControlContainer container = (CD_DotNetControlContainer)(ContainerElements[0].ActiveElement);
        // Use the getActiveXProperties() and the name of the Properties "Information" we can set
        // to an "info" variable("New Information") or the Property;
        // if (container.getActiveXProperty("Information") == "MyInformation")
        //    info = container.getActiveXProperty("Information", "New Information");
        if (container.getActiveXProperty("Information") == "MyInformation")
            info = container.getActiveXProperty("Information", "New Information");
    }
    catch (Exception ex)
    {
    }
}
```

7. Alla fine:
   - create una funzione di zenon Esegui macro VSTA
   - collegate la funzione ad un bottone

Durante il Runtime, si cambia il Label cliccando sul pulsante `myInformation` su `New Information`. 
Se si clicca di nuovo sullo stesso pulsante si torna alla situazione precedente.

5.3 **Esempio .NET Control come ActiveX (C#)**

Il seguente esempio descrive un .NET Control che viene eseguito come ActiveX Control in zenon.

La creazione e integrazione avviene in quattro passi:

1. creare Windows Form Control (A pagina: 32)
2. trasformare .NET User Control in Dual Control (A pagina: 36)
3. lavorare nell’Editor tramite VBA con ActiveX (A pagina: 41)
4. Collegare variabili di zenon con il .NET User Control (A pagina: 42)

5.3.1 **creare Windows Form Control**

Per creare un Windows Form Control:
1. Avviare il Visual Studio 2008 e creare un nuovo progetto Windows Form Control Library:

![Image of New Project dialog in Visual Studio]

2. Rinominare il Control di default dando gli il nome desiderato.
   
   Nel nostro esempio: SampesControl.cs.

![Image of Solution Explorer in Visual Studio]
3. Aprite il Control Designer e aggiungete i Control desiderati, nel nostro caso una casella di testo (Text-box):

Nel nostro esempio: La property visibile esternamente, "UserText" con get e accesso set che contiene il testo della casella testo:

![Image of Code Editor and Windows Forms Control](image.png)

```
namespace myNamespace
{
    public partial class SampleControl : UserControl
    {
        public SampleControl()
        {
            InitializeComponent();
        }

        public string UserText
        {
            get { return this.textBox1.Text; }
            set { this.textBox1.Text = value; }
        }
    }
}
```

5. compilare il progetto.

Il Windows Forms Control può essere adesso usato in altri progetti Windows Forms.
**Importante:** il Control deve essere inserito manualmente nel Control Toolbox facendo uso di Choose Items.

5.3.2 trasformare .NET User Control in Dual Control

Per trasformare .NET in un Dual Control, si deve anzitutto procedere ad attivare l’interfaccia COM per ActiveX.
1. Aprire il progetto e attivate alla voce **Build**-**impostazioni** la proprietà **Register for COM interop**:

![Register for COM interop](image)

2. Aprire il file **AssemblyInfo.cs** e
   - **settate l´attributo ComVisible su true**
   - **aggiungete l´attributo ClassInterface**

   ```csharp
   [assembly: ComVisible(true)]
   ```
[assembly: ClassInterface(ClassInterfaceType.AutoDual)]
3. Aprire il Codedesigner tramite **View Code** e aggiungete gli attributi ActiveX e gli inserimenti **using** necessari. Creare un nuovo GUID per l’attributo GUID servendovi del menu **Tools/Create GUID**:

![Visual Studio Code](image)

4. Per fare in modo che il Control sia selezionabile anche come Control superfici ActiveX, si devono ancora aggiungere nella classe control le seguenti funzioni:
   - **RegisterClass**
- `UnregisterClass`

Poi il controllo può essere registrato nella Registry.

5. **Ricompilare il progetto.**

Il Windows Form Control è adesso utilizzabile anche con ActiveX ed è stato registrato automaticamente con il Rebuild. Un ulteriore file Typelib `zenOnDotNetControl.tlb` è stato creato nel percorso output.

6. **Per usare il controllo su un altro calcolatore:**
   
   a) copiate il file DLL e il file TBL sul calcolatore di destinazione

   b) registra il file tramite la riga comandi con:

   ```
   %windir%\Microsoft.NET\Framework\v2.0.50727\regasm.exe zenOnDotNetControl.dll /tlb:zenOnDotNetControl.tlb
   ```
7. Aggiungete l’esteso Windows Form Control come ActiveX Control nell’Editor in zenon:

5.3.3 lavorare nell’Editor tramite VBA con ActiveX

Per poter accedere nell’Editor zenon alle proprietà del Control:

1. create nell’Editor di zenon, per la precisione nel nodo Interfacce di programmazione/Macro-VBA una nuova macro Init con la denominazione Init_ActiveX.

Con questa macro è poi possibile accedere a tutte le properties esterne mediante obElem.AktiveX...
2. Assegnate questa macro all’ActiveX Control facendo uso della proprietà Macro VBA/Init dell’elemento ActiveX.

**ESEMPIO MACRO INIT**

```vba
Public Sub Init_ActiveX(obElem As Element)
    obElem.AktiveX.Usertext = "Setta la stringa sul Control"
End Sub
```

5.3.4 Collegare variabili di zenon con il .NET User Control

Sotto zenon avete la possibilità di estendere un ActiveX Control integrandolo con delle funzioni specifiche, per poter accedere a API zenon.

**METODI NECESSARI**

- public bool zenOnInit (A pagina: 44) (Viene aperto al momento dell’inizializzazione del Control nel Runtime di zenon.)
- public bool zenOnInitED (A pagina: 45) (Viene usato nell’Editor.)
- public bool zenOnExit() (A pagina: 45) (Viene aperto al momento della distruzione del Control nel Runtime di zenon.)
- public bool zenOnExitED() (A pagina: 45) (Viene usato nell’Editor.)
- public short CanUseVariables() (A pagina: 45) (Supporta il collegamento di variabili.)
- public short VariableTypes() (A pagina: 45) (Tipi di dato supportati da Control)
- public MaxVariables() (A pagina: 46) (Numero massimo delle variabili che possono essere collegate con il Control..)

AGGIUNGI REFERENZE

1. In Microsoft Visual Studio selezionate alla voce Add References la biblioteca oggetti-Runtime di zenon, per poter accedere nel Control a zenon API.

2. Aggiungete le funzioni estese nel class-code del control per accedere all’intera zenon API.
Nel nostro esempio, il COM Object di una variabile di zenon viene salvato temporaneamente in Member un per poter poi accedere nel Paint Event del Control:

```csharp
public bool zenOnInit(zenOn.Element dispElement)

Con l’aiuto di questo metodo, viene consegnato (nel runtime) all’ActiveX Control una lancetta sulla
Dispatch Interface dell’elemento dinamico. Grazie a questa lancetta si può passare alle variabili di zenon
collegate all’elemento dinamico.

La successione delle variabili può essere impostata nel dialogo "Immissione elemento" con il pulsante in
di basso o in alto. Il dialogo Introduzione elemento si apre se:

- cliccate due volte sull’elemento ActiveX
- oppure se selezionate Proprietà nel menu contestuale, oppure
- selezionate nella finestra delle proprietà nel nodo Visualizzazione la proprietà
  Impostazioni ActiveX
public bool zenOnInitED(zenOn.Element dispElement)

Corrisponde a public bool zenOnInit (A pagina: 44) e viene eseguito al momento dell’apertura dell’ActiveX nell’Editor (doppio clic su ActiveX)

public bool zenOnExit()

Questo metodo viene chiamato in causa dal runtime di zenon al momento della chiusura dell’ActiveX Control. In questo caso dovrebbero essere resi disponibili tutti i Dispatch Pointer su variabili.

public bool zenOnExitED()

Corrisponde a public bool zenOnExit() (A pagina: 45) e viene eseguito nell’editor al momento della chiusura dell’ActiveX. In tal modo si può reagire anche nell’Editor a modifiche, come, ad esempio, cambio di valori.

public short CanUseVariables()

Questo metodo fornisce come risultato 1 se il Control può usare delle variabili di zenon e 0 in caso contrario

- 1: È possibile indicare per l’elemento dinamico (tramite il pulsante Variabili) solamente delle variabili di zenon dei tipi indicati tramite il metodo VariableTypes e nella quantità indicata nel metodo MaxVariables.

- 0: Nel caso in cui CanUseVariables dia 0, oppure nel caso in cui il Control non possieda tale metodo, possono essere indicate variabili di ogni tipo e senza limitazioni di numero. Esse possono essere usate nel runtime, ma solo tramite VBA.

public short VariableTypes()

Il valore restituito da questo metodo viene usato come maschera per i tipi di dato utilizzabili nella lista delle variabili. Il valore è una connessione AND dei seguenti valori (definiti in zenon32/dy_type.h):
<table>
<thead>
<tr>
<th>Parametri</th>
<th>Valore</th>
<th>Descrizione</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD</td>
<td>0x0001</td>
<td>corrisponde a posizione 0</td>
</tr>
<tr>
<td>BYTE</td>
<td>0x0002</td>
<td>corrisponde a posizione 1</td>
</tr>
<tr>
<td>BIT</td>
<td>0x0004</td>
<td>corrisponde a posizione 2</td>
</tr>
<tr>
<td>DWORD</td>
<td>0x0008</td>
<td>corrisponde a posizione 3</td>
</tr>
<tr>
<td>FLOAT</td>
<td>0x0010</td>
<td>corrisponde a posizione 4</td>
</tr>
<tr>
<td>DFLOAT</td>
<td>0x0020</td>
<td>corrisponde a posizione 5</td>
</tr>
<tr>
<td>STRING</td>
<td>0x0040</td>
<td>corrisponde a posizione 6</td>
</tr>
<tr>
<td>IN_OUTPUT</td>
<td>0x8000</td>
<td>corrisponde a posizione 15</td>
</tr>
</tbody>
</table>

**public MaxVariables()**

In questa sede si stabilisce il numero delle variabili che possono essere selezionate dalla lista di variabili.

1: nella lista variabili viene disattivata la possibilità di multiselect. Il sistema emette un avvertimento nel caso in cui si tenti di selezionare più variabili contemporaneamente.

### 6. Elemento WPF

Con l’elemento dinamico WPF possono essere implementati e visualizzati file WPF/XAML validi in zeron.
6.1  Basi:

XAML


WPF


- WPF rappresenta l’ambiente di programmazione.
- XAML descrive, basandosi su XML, l’interfaccia grafica con un linguaggio markup.
  A seconda della struttura del file XAML si ha la possibilità di collegare proprietà, eventi e trasformazioni degli elementi WPF con variabili e funzioni di zenon.
- Il framework riunisce diverse aree di una presentazione come l’interfaccia utente, disegni, grafica, audio, video, documenti e tipografia.


6.1.1  WPF nella visualizzazione di processo

Per zenon, XAML apre delle possibilità grafiche individuali. Elementi di visualizzazione e elementi dinamici possono essere adattati graficamente indipendentemente dalla progettazione. Per esempio, delle illustrazioni complesse vengono create da designer e poi importate in forma di file XAML in zenon e collegate con le logiche desiderate. Ci sono le seguenti possibilità di applicazione:
ELEMENTI DINAMICI IN ANALOG-LOOK

Le grafiche non devono più essere disegnate in zenon, ma possono essere importate direttamente sotto forma di file XAML. Ciò permette l’uso di elementi illustrati in modo complesso nella visualizzazione di processo. Riflessi, ombre, effetti 3D ecc, ad esempio, vengono supportati come grafica. Gli elementi adattati alla branca industriale in questione permettono una gestione intuitiva, analoga agli strumenti sul macchinario.

ILLUSTRAZIONI COMPLESSE PER GESTIONE INTUITIVA

L’implementazione di elementi di visualizzazione basati su XAML conferisce una veste grafica più pregevole ai progetti e permette una visualizzazione di processo più semplice e chiara. Gli elementi ottimizzati in fatto di usabilità facilitano la gestione. La visualizzazione chiara di dati semplicifica la comprensione di contenuti complessi. La possibilità di adattamento flessibile dei singoli elementi aumenta il comfort della gestione. In tal modo è possibile per il progettista determinare indipendentemente valori, scale e unità.

PRESENTAZIONE DATI CHIARA E RIASSUNTI

Elementi di visualizzazione raggruppati permettono una rappresentazione chiara dei dati di processo importanti, in modo tale che l’operatore che gestisce l’impianto sia sempre informato su quanto sta accadendo nel processo. Valutazioni grafiche, valori e slider possono essere raggruppati in un elemento e permettono una gestione veloce e non semplice.

VISUALIZZAZIONE SPECIFICA PER UN SETTORE INDUSTRIALE
Elementi come termometro, scale o bargraf fanno parte degli elementi base della visualizzazione di processo. Attraverso XAML è possibile adattare l’aspetto grafico di questi elementi alle diverse esigenze delle differenti branche industriali. In tal modo, gli operatori che lavorano nell’impianto trovano gli elementi cui sono abituati, che conoscono già a causa del loro lavoro con i macchinari, nella visualizzazione del processo sul terminale.

**ADATTAMENTO A CORPORATE DESIGN**

Le illustrazioni possono essere anche adattate alle indicazioni dei diversi committenti permettendo di raggiungere un’immagine omogenea del processo produttivo. Come base, possono essere utilizzati, ad esempio, gli elementi di gestione standard di zenon, che verranno poi adattati allo stile del Corporate Design.

### 6.1.2 Trasmissione valori di zenon a WPF

zenon lavora internamente sempre con il tipo di dato `Double` oppure `String`. Essi vengono inviati all’elemento WPF. L’elemento WPF è connesso in maniera scollegata in un .NET Container. Affinché il tipo di dato possa essere usato, deve essere stato per lo più convertito. Questa conversione viene eseguita automaticamente da .NET.

Il settaggio del valore avviene secondo le seguenti regole:

- se il tipo .NET (`System.Object`) non è visibile perzenon, il valore viene mandato così com’è a .NET. .NET deve occuparsi della rappresentazione o della conversione.

- Se il tipo .NET è del tipo boleano (`System.Boolean`), zenon scrive, come previsto dalla convenzione.NET, 0 oppure –1.

- se il tipo.NET è conosciuto, il sistema controlla se .NET può convertire il valore. A questo scopo viene usato il convertitore di .NET.
  - Si: Il valore viene inviato.
  - No: il valore viene inviato comunque. Se .NET reagisce con una segnalazione di errore, il valore viene trasformato da zenon in una string e viene inviato di nuovo.
6.1.3 Oggetti referenziati

In WPF non possono essere usati solamente oggetti standard, come rettangoli, pulsanti, campi testo ecc, ma anche WPF User Controls, che vengono referenziati come Assembly.

WPF User Controls sono oggetti creati individualmente. Per esempio, questo elemento può avere l’aspetto di un tachimetro e offrire delle proprietà speciali e degli effetti ottici, come, ad esempio, la proprietà „Value”, che, al momento di essere settata, fa sì che la lancetta del tachimetro si muova e indichi il valore.

Il workflow per raggiungere questo obiettivo:

- L’aspetto di un User Controls viene disegnato con oggetti standard che vengono offerti da WPF.
- Le proprietà e le interazioni vengono programmate.
- Il pacchetto completo viene compilato e è disponibile come .NET Assembly.

Questo Assembly può essere usato per progetti WPF. Per far ciò, deve essere referenziata (collegata) nell’Editor WPF (per es: Microsoft Expression Blend). Selezionate a questo scopo nel dialogo di selezione file in zenon l’Assembly:

Da questo momento i WPF User Controls dell’Assembly possono essere selezionati nel Toolbox alla voce Custom User Controls e possono essere usati nel progetto WPF.

USA ASSEMBLYS REFERENZIATI IN ZENON

Per usare un Assembly in zenon, deve essere reso disponibile in forma di file. File collection del formato .cdwpf gestiscono ciò autonomamente, il che significa che non c’è bisogno di altri passi di configurazione. Per i file del formato .xaml, invece, devono essere aggiunte le assemblies alla cartella File:

- cliccate nella struttura ad albero su File
- selezionate Altro
- selezionate nel menu contestuale Aggiungi file
- Si apre un dialogo per la selezione file
- aggiungete l’Assembly desiderato
Nella visualizzazione di un file WPF nell’elemento WPF (editor e runtime), gli Assembly vengono caricati da questa cartella. In tal modo si assicura che, quando vengono trasmessi i file runtime per mezzo di Trasporto Remoto, tutti gli assemblys referenziati siano esistenti sul computer di destinazione.

Un file collection (.cdwpf) e un file XAML con lo stesso nome possono esistere in parallelo. Tutte le assemblies (*.dll) dei file collection e della cartella Altri vengono copiate nella cartella di lavoro. Nel caso esistano diverse assemblies dello stesso nome, viene usata quella che ha il numero di versione file più alto.

⚠️ **Attenzione**

Gli Assembly vengono eliminati dopo il caricamento solo quando l’applicazione viene chiusa. Il che significa:

se un file WPF viene visualizzato con un assembly referenziato in zenon, questo assembly viene caricato e si trova nella memoria fino alla chiusura di zenon, anche se l’immagine è stata chiusa. Se volete eliminare un assembly dalla cartella File/Altri, prima di tutto bisogna riavviare l’editor in modo tale che l’assembly venga eliminato.

**GESTIONE MULTI-PROGETTO**

Nella gestione multi-progetto, in tutti i progetti deve essere usato lo stesso Assembly. Se in un progetto un Assembly viene sostituito da un’altra versione, essa deve essere sostituita anche in tutti gli altri progetti che sono caricati nell’Editor o nel Runtime.

6.1.4 **Assegnazione di oggetto zenon a WPF Content**

L’assegnazione di oggetti zenon a WPF Content avviene mediante il Nome dell’oggetto WPF. Bisogna tener presente:

Oggetti visuali non possiedono nessuna proprietà RuntimeNamePropertyAttribute. Per questo motivo nel momento in cui viene caricato WPF Content, la informazione ulteriore Name non è disponibile.

In tal modo, non è più possibile un’assegnazione univoca di oggetti zenon a oggetti WPF. Per questo motivo, nel dialogo di configurazione di zenon vengono elencati solamente oggetti logici. Nel MSDN o nella Microsoft Website è visibile quali oggetti WPF dispongono della proprietà RuntimeNamePropertyAttribute.
WORKAROUND

Per dinamizzare oggetti visuali, è possibile adottare il seguente workaround:

nel caso di elementi visuali, la proprietà dinamicizzante viene collegata mediante una connessione data con la Text-Property di un Textbox invisibile.

Visto che la Textbox supporta come oggetto logico la proprietà Name, questa viene visualizzata zenon.

La Textbox-Property può essere dunque dinamicizzata in zenon.

In tal modo viene dinamicizzato indirettamente anche l’oggetto visuale.

6.1.5 Workflows

La tecnologia WPF/XAML consente nuovi workflows nella visualizzazione di processo. La separazione fra design e funzionalità è garanzia di una chiara divisione dei compiti fra progettista e designer; i compiti di design sono facili da eseguire grazie alla possibilità di utilizzare design già esistenti che non devono essere più modificati dal progettista.

Al workflow per la creazione di elementi WPF inzenon partecipano le seguenti persone:

- Designer
  - illustra elementi
  - si occupa dell’aspetto grafico per MS Expression Design

- operatore MS Expression Blend
  - anima gli elementi
  - crea delle variabili per l’animazione degli elementi WPF inzenon, ai quali può accedere il progettista.

- il progettista
  - implementa elementi in zenon
  - imposta logica e funzionalità

Distinguiamo:

- workflow con Microsoft Expression Blend (A pagina: 89)
- workflow con Adobe Illustrator (A pagina: 89)
**workflow con Microsoft Expression Blend**

Se si usa Microsoft Expression Blend, la creazione di un elemento WPF avviene in quattro passaggi:

1. illustrazione dell’elemento in **MS Expression Blend** (A pagina: 90)
2. aprire l’elemento in **MS Expression Design** e esportare come WPF
3. Animazione in **MS Expression Blend** (A pagina: 90)
4. Integrazione in zenon (A pagina: 137)

Un esempio di creazione di un elemento WPF con Microsoft Expression Blend lo trovate nel capitolo Crea pulsante come file XAML usando Microsoft Expression Blend erstellen (A pagina: 90).

**workflow con Adobe Illustrator**

Con **Adobe Illustrator** abbiamo il seguente Workflow:

1. Illustration degli elementi in **Adobe Illustrator** (A pagina: 95)
2. Import di file .ai e modifica in **MS Expression Design** (A pagina: 97)
3. WPF-Export da **MS Expression Design** (A pagina: 97)
4. Animazione in **MS Expression Blend** (A pagina: 99)
5. Integrazione in zenon (A pagina: 144)

Un esempio di questa procedura lo trovate nel capitolo Workflow con Adobe Illustrator (A pagina: 94).

### 6.2 Indicazioni per i designer

Questo capitolo fornisce delle informazioni concernenti il modo corretto di creare file WPF in Microsoft Expression Blend e Adobe Illustrator. I tutorial per la Creazione di un elemento pulsante (A pagina: 90) e di un elemento bargraf (A pagina: 94) mostrano il modo in cui, con poche azioni, vengono creati dei file WPF completamente funzionanti per zenon partendo da grafiche già esistenti.

A tal scopo, si usano i seguenti strumenti:

- Adobe Illustrator CS3 (AI)
Informazioni su

Se volete usare in WPF degli oggetti referenziati (Assemblys), tenete presente le indicazioni contenute nel capitolo Oggetti referenziati (A pagina: 86).

6.2.1 workflow con Microsoft Expression Blend

Con Microsoft Expression Blend, un elemento WPF viene:

- illustrato
- in MS Expression Design convertito in formato WPF
- animato

L’esempio seguente mostra l’illustrazione e la trasformazione di un elemento pulsante in un file XAML.

Nota: Una versione test di "Microsoft Expression Blend" può essere scaricata dalla Website di Microsoft.

Creare un pulsante come file XAML con Microsoft Expression Blend

CREA UN PULSANTE

1. Avviare Expression Blend
2. Selezionate l’opzione **New Projekt**

3. Selezionate come tipo di progetto **WPF**

4. Assegnate un percorso a Vostra scelta e un nome (per es. MyBlendProject)

Potete ignorare le impostazioni **Language** e **Version** visto che non deve essere programmata nessuna funzionalità.
5. Dopo aver chiuso il dialogo cliccando su **OK**, Microsoft Blend crea un nuovo progetto con le impostazioni selezionate. Expression Blend aggiunge un file XAML vuoto, che contiene già una referenza di classe.

6. Cancellate il file relativo a quello XAML facendo uso del menu contestuale.

7. Cambiate il nome del file XAML da `MainControl.xaml` a `MyButton.xaml`.

8. La grandezza development del file, viene settata di default a 640 volte 480 pixel e dovrà essere modificata:
   a) Passate alla visuale **XAML**
   b) Correggete la grandezza portandola a 100 volte 100 pixel.
   c) Cancellate la referenza di classe `x:Class="MyBlendProject.MyButton"`

9. Passate alla visuale **Design**

10. Aggiungete tramite la barra dei simboli un pulsante.

11. definite le proprietà.
   - **Width**: 50
In tal modo, il pulsante si trova adesso al centro del control.

12. Salavate le modifiche e aprite il file a fini di controllo in Internet Explorer. Vedrete che il pulsante adesso viene visualizzato nella dimensioni di 50 x 50 pixel.

**RENDERE IL PULSANTE SCALABILE**

Se implementate questo file in questo stato in zenon, il pulsante avrà sempre esattamente la dimensione pari a 50 x 50 pixel. Visto che il pulsante deve essere usato per scalabilità, passate di nuovo a Expression Blend:

1. selezionate il pulsante nella visione ad albero.

2. Selezionate nel menu contestuale del bottone la voce *Group Into -> Viewbox*

3. Il pulsante viene aggiunto in una *Viewbox*.

4. definite le proprietà di Viewbox
   - **Width**: Auto
   - **Height**: Auto
5. Salvate il file.

6. Se adesso avete aperto il file in Internet Explorer, il pulsante viene automaticamente scalato in caso di cambio delle dimensioni nella finestra IE. Questo file si adatterà automaticamente anche in zenon ad un eventuale cambio delle dimensioni dell’elemento WPF.

MODIFICA NOME

Prima di poter implementare il file in zenon, dovete dare un nome all’elemento WPF. Di default, gli elementi WPF non hanno un nome in Expression Blend e sono distinti da parentesi quadrate e dal loro tipo. L’attribuzione di content zenon a WPF-Content avviene tramite il nome degli elementi WPF:

- modificati nella struttura ad albero il nome
- del pulsante chiamandolo MyButton
- Cambiate il nome di ViewBox a MyViewBox

Questo pulsante può essere integrato adesso come file XAML in zenon (A pagina: 137).

6.2.2 workflow con Adobe Illustrator

Se si usa Adobe Illustrator, un elemento WPF viene:

- illustrato in Adobe Illustrator
- in MS Expression Design convertito in un WPF
- animato in MS Expression Blend
L’esempio seguente mostra l’illustrazione e la trasformazione di un elemento bargraf in un file XAML.

**Illustration Bargraf**

Un bargraf viene creato in Adobe Illustrator.

1. **AI: Elemento di partenza Bargraf**

   ![Illustrato in Adobe Illustrator CS3.](image)

2. **AI: visuale percorso Bargraf in Adobe Illustrator**

   - tutti gli effetti devono essere trasformati (oggetto -> trasforma aspetto)
   - tutte le linee vengono trasformate in percorsi (oggetto -> Percorso -> linea di contorno)
   - non usare nessun filtro come ombra ecc

**INDICAZIONI CONCERNENTI LA COMPATIBILITÀ**

Le illustrazioni che vengono relizzate facendo uso di Adobe Illustrator, sono adatte in linea di principio ad un export WPF. Tuttavia, non tutti gli effetti illustrator possono essere convertiti in modo adeguato in Expression Design/Blend. Tenete presente:
<table>
<thead>
<tr>
<th>Effetto</th>
<th>Descrizione</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maschere di ritaglio</td>
<td>Le maschere di ritaglio create in Adobe Illustrator non vengono interpretate correttamente da Expression Design. Per lo più, esse vengono visualizzate in Blend come delle superfici di color nero. Vi consigliamo di impostare le illustrazioni senza maschere di ritaglio.</td>
</tr>
<tr>
<td>Filtri e effetti</td>
<td>Non ogni filtro di Adobe Illustrator viene ripreso in Expression Design in modo esattamente corrispondente. I filtri di sfocatura gaussiana, effetti per il design degli angoli di Illustrator, per esempio, non funzionano in Expression Design.</td>
</tr>
<tr>
<td>Campi di testo</td>
<td>per poter collegare campi di testo con codice, essi devono essere impostati separatamente in Expression Blend. <em>Labels</em> sono necessari per testi dinamici; per le informazioni statiche, invece, sono sufficienti semplici <em>Campi testo</em>. MS Design non prevede la possibilità di creare label di testo. Dovete impostarli direttamente in MS Blend.</td>
</tr>
<tr>
<td>Trasparenze e trasparenze</td>
<td>Si possono avere delle difficoltà in Adobe Illustrator per la corretta interpretazione di impostazioni di trasparenza, in particolare di quelle concernenti le trasparenze di gruppo. MS Expression Blend e MS Expression Design offrono la possibilità di creare di nuovo delle impostazioni di trasparenza.</td>
</tr>
<tr>
<td>di gruppo</td>
<td></td>
</tr>
<tr>
<td>Moltiplica livelli</td>
<td>Queste impostazioni di livello in Adobe Illustrator non vengono riprodotte sempre correttamente in MS Expression Blend. C’è però la possibilità di impostare direttamente in Expression Design <em>Moltiplica livelli</em>.</td>
</tr>
<tr>
<td>Strumenti analogici e</td>
<td>Per preparare le grafiche nel modo migliore all’animazione, lancetta e indicatore devono essere sempre nella posizione di partenza, per lo più ore 0 oppure 12:00 . In questo modo, i parametri di posizione per rotazioni ecc, vengono indicati in Blend correttamente e l’animazione può avvenire senza la necessità di una conversione delle indicazioni concernenti la posizione.</td>
</tr>
<tr>
<td>posizioni standard</td>
<td></td>
</tr>
</tbody>
</table>
Export WPF

Per realizzare l’animazione in Microsoft Expression Blend c’è bisogno di file WPF. Consigliamo l’uso di Microsoft Expression Design per realizzare questo export, dato che esso fornisce buoni risultati e che la maggior parte degli effetti Illustrator vengono interpretati correttamente.

Nota: in Internt viene offerto un Plug-in gratuito per l’export diretto di file WPF da Adobe Illustrator. Questo Plug-in offre la possibilità di effettuare un export veloce e non complicato da Illustrator; per l’applicazione attuale, però, non è molto adatto visto che comporta delle perdite di pregevolezza grafica. Sono possibili anche delle divergenze di colore rispetto al documento originario.

File nel formato .ai possono essere importati regolarmente in Expression Design. I percorsi sono mantenuti.

Attenzione Alcuni effetti Illustrator comuni, però, non possono essere rappresentati correttamente da Expression Design (vedi capitolo Illustration (A pagina: 95)).

In 5 passi esportiamo l’elemento bargraf già impostato:

1. **ED: Importazione**
   - importate il File Illustrator (A pagina: 95) preparato in Microsoft Expression Design 3 facendo uso di *file -> Importare*
2. **ED: Ottimizzazione**

- se il file originario non viene correttamente visualizzato in MS Expression Design, esso può essere modificato e ottimizzato

3. **ED: Seleziona**

- evidenziate con la freccia **Seleziona diretta** in MS Expression Design l’elemento per l’export WPF, in questo caso l’intero orologio.

4. **ED: Avvia Export**

- avviare l’export mediante **File -> Esporta**
- si apre il dialogo per la configurazione delle impostazioni di export
5. **ED: Impostazioni di Export**

- Effettuate le seguenti impostazioni di export:
  a) **Formato:** XAML Silverlight 4 / WPF Canvas  
     Nominare sempre gli oggetti: **attivate con simbolo di spuntatura**  
     Piazzate oggetti raggruppati in un XAML Layoutcontainer: **attivate con simbolo di spuntatura**
  b) **Testo:** Blocco testo modificabile  
  c) **Effetti linea:** tutti griglia

Il file esportato ha la desinenza `.xaml`. Nel successivo passaggio viene preparato e animato (A pagina: 99) in MS Expression Blend.

**Animazione in blend**

Con MS Expression Blend:

- vengono animati i file XAML del MS Expression Design.
- le variabili per la gestione degli effetti che possono essere utilizzati da zenon, vengono create.

In 13 passi giungiamo dallo statico XAML all’elemento animato che può essere integrato in zenon:

1. **crea progetto EB:**

   a) apriente Microsoft Expression Blend  
   b) impostate un nuovo progetto  
   c) selezionate come **Tipo di progetto WPF-> WPF- Control Library**  
   d) asseginate un nome (nel nostro tutorial: **Mein_Projekt**)  
   e) selezionate un luogo dove salvare il progetto
f) selezionate una lingua (nel nostro tutorial: C#)
g) selezionate la Framework Version 3.5

2. **EB: cancella MainControl.xaml.cs**

   a) passate a `MainControl.xaml.cs`
   b) cancellate questo file facendo uso del comando **Cancella** nel menu contestuale

3. **EB: apri file XAML esportati**

   a) apriete il menu contestuale per **Mein_Projekt** (con il tasto destro del mouse)
   b) selezionate **Aggiungi elemento esistente**
   c) selezionate il file XAML esportato da Microsoft Expression Design per aprirlo in Microsoft Expression Blend.

4. **EB: apri MainControl.xaml**

   a) aprite il `MainControl.xaml` generato automaticamente
   b) navigate nel settore oggetti e assi tempo all’inserimento **UserControl**

5. **EB: adatta XAML-Code**

   a) cliccate con il tasto destro del mouse **su UserControl**
   b) selezionate nel menu contestuale **visualizza XAML**
   c) cancellate nel XAML-Code le righe 7 e 9:
x:Class="Mein_Projekt.MainControl"
d:DesignWidth="640" d:DesignHeight="480"

6. **EB: controlla XAML-Code**

- Adesso il XAML Code dovrebbe avere il seguente aspetto:

```xml
<UserControl
    xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation
    xmlns:x=http://schemas.microsoft.com/winfx/2006/xaml
    mc:Ignorable="d"
    x:Name="UserControl">

    <Grid x:Name="LayoutRoot"/>

</UserControl>
```

7. **EB: copia elementi**

   a) apri il file XAML importato da Expression Design

   b) evidenzia tutti gli elementi

   c) seleziona nel menu contestuale copia

   d) torna al file XAML generato automaticamente

8. **EB: inserisci elementi**

   a) clicca con il tasto destro del mouse su **Layout Root**
b) selezionate Aggiungi

9. **EB: adatta tipo layout**

   a) cliccate con il tasto destro del mouse su *LayoutRoot* -> *modifica tipo di layout* -> *Viewbox*
   
   b) adesso la struttura dovrebbe avere il seguente aspetto: *UserControl* -> *LayoutRoot* -> *Grid* -> *Elementi*
   
   c) assegnate un nomer per *LayoutRoot* e *Grid* cliccando due volte sul nome

10. **EB: testi e formati**

    • testi dinamici e statici vengono segnati con campi di testo
    • Valori (numeri) vengono emessi con *Labels*

11. **EB: inserisci label**

    • *Labels* sostituiscono i numeri che devono essere collegati in seguito per mezzo di variabili *Int* (deve essere eseguito per tutti gli elementi-numeri)

12. **EB: setta proprietà**

    • per visualizzare il 100%, settate la proprietà *MaxHeight* dell’elemento visualizzazione bargraf sul valore 341 (l’altezza massima dell’elemento indicatore corrisponde a 340)
Elemento WPF

13. **EB: prepara per uso in zenon**

- a) cancella tutte le denominazioni nome (i nomi possono essere assegnati solamente per gli elementi che devono essere utilizzati tramite zenon)
- b) salvate il file XAML con un nome qualsiasi.
- c) integrate il file XAML in zenon (A pagina: 144)

**Un consiglio per il controllo:** se il file XAML viene visualizzato in Microsoft Internet Explorer senza problemi e si adatta alla grandezza della finestra di Internet Explorers, viene usato correttamente anche in zenon.

### 6.3 Engineering in zenon

Per poter utilizzare i WPF con zenon deve essere installato Microsoft Framework 3.5 sia sul pc con l’Editor che sul pc con il Runtime.

**CONDIZIONI PER VIsUALIZZAZIONE WPF IN ZENON**

La dinamicizzazione è disponibile al momento per variabili semplici; array e strutture non possono dinamicizzate. Per questo motivo possono essere utilizzate le seguenti funzioni WPF in zenon:

- Proprietà dell’elemento, che corrispondono a tipi di dato semplici come `String`, `Int`, `Bool` ecc.
- Proprietà dell’elemento del tipo "Object", che possono essere settate con semplici tipi di dato.
- Eventi dell’elemento possono essere usati con funzioni; i parametri degli eventi, però, non sono disponibili/analizzabili in zenon.
- Elementtransformation per cui è disponibile un `RenderTransform` nell’elemento nel file XAML.

**Attenzione:** Se in trasformazione il content si trova al di là dell’area dell’elemento WPF, questa parte del contents va perduto, ppure non disegnato.

**Indicazioni sull’ombra:** Per gli elementi WPF non è possibile visualizzare un’ombra in zenon.
Attenzione

Se vengono generati i file runtime per un progetto per una versione precedente a quella 6.50, gli esistenti Elementi WPF non vengono implementati nelle immagini runtime.

REQUISITI SOTTO WINDOWS VISTA E WINDOWS 7

Se un’immagine del tipo WPF contiene uno slider e vengono usati nello stesso tempo effetti "aero" di Windows Vista o Windows 7, si possono verificare dei problemi di refresh nell’editor di zenon.

6.3.1 Crea elemento WPF

Per creare un elemento WPF:

1. selezionate nella barra dei simboli degli elementi il simbolo Elemento WPF, oppure scegliete l’inserimento corrispondente nel menu Elementi
2. selezionate nella finestra principale il punto start
3. Trascinate l’elemento con il mouse
4. selezionate fra le proprietà nel gruppo Visualizzazione quella File XAML
5. Si apre il dialogo di selezione file.
6. selezionate il file desiderato da
   Sono validi i file del formato
   - *.xaml: Extensible Application Markup Language
   - *.cdwpf: WPF file collection, mostra anche anteprima
   (Il file deve essere già disponibile nel manager di progetto alla voce File/GraficheOppure essere impostato nuovamente nel dialogo.)
7. Configurate il collegamento (A pagina: 105)
6.3.2 CDWPF-file collection (file collection)

Regole per l’uso di file collection:

- I file possono trovarsi direttamente nel file ZIP, oppure in una cartella comune.
- Il nome del file XAML dovrebbe corrispondere a quello del file collection.
- Può essere contenuto solamente un file XAML.
- La grafica di anteprima dovrebbe essere piccola e non essere più alta di 64 pixel. Nome del file anteprima: preview.png oppure il nome del file XAML con la desinenza .png.
- Si possono usare quante assemblies si voglia. La differenziazione avviene sulla base della versione del file in forma numerica.
- File collection non devono necessariamente contenere Assembly.
- Il sistema passa a setaccio tutte le cartelle e tiene in considerazione solamente i file del tipo *.dll, *.xaml oppure *.png.
- Se un file collettivo (.cdwpf) viene sostituito da un file con una differente versione, devono essere adattati tutti i file CDWPF corrispondenti in tutti i simboli e immagini in tutti i progetti.

6.3.3 Configurazione del collegamento

Per configurare un elemento WPF

1. selezionate fra le proprietà nel gruppo Visualizzazione quella Configurazione
2. Un dialogo con tre tab si apre con un’anteprima del file XAML, nonché con gli elementi presenti nel file.
<table>
<thead>
<tr>
<th>Parametri</th>
<th>Descrizione</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Elementi esistenti</strong></td>
<td>Mostra in una struttura ad albero i detti elementi del file. L’elemento selezionato può essere collegato con dati di processo. L’attribuzione di WPF a dati di processo avviene sulla base del nome elemento. Per questo motivo, gli elementi vengono visualizzati solamente se essi e i relativi elementi sovraridinti possiedono un nome. Le attribuzioni vengono configurate e mostrate nei tab Proprietà, Eventi e Trasformazioni.</td>
</tr>
<tr>
<td><strong>Anteprima</strong></td>
<td>Nell’anteprima, l’elemento selezionato lampeggia.</td>
</tr>
<tr>
<td><strong>Proprietà (A pagina: 108)</strong></td>
<td>Configurazione e visualizzazione delle proprietà (variabili, autorizzazioni, interblocchi, valori collegati).</td>
</tr>
<tr>
<td><strong>Eventi (A pagina: 115)</strong></td>
<td>Configurazione e rappresentazione degli eventi (funzioni).</td>
</tr>
<tr>
<td><strong>Trasformazione (A pagina: 117)</strong></td>
<td>Configurazione e rappresentazione delle trasformazioni.</td>
</tr>
<tr>
<td><strong>Nome</strong></td>
<td>Nome della proprietà.</td>
</tr>
<tr>
<td><strong>Collegamento</strong></td>
<td>Sezione del collegamento.</td>
</tr>
<tr>
<td><strong>Tipo di collegamento</strong></td>
<td>Tipo di collegamento (variabile, autorizzazione, funzione)</td>
</tr>
<tr>
<td><strong>Info WPF</strong></td>
<td>Mostra l’attuale valore nel WPF-Content per le proprietà. Per l’utente è direttamente visibile di quale tipo di proprietà si tratta (boleana, string, ecc).</td>
</tr>
<tr>
<td><strong>Collegato</strong></td>
<td>Indica se una proprietà viene utilizzata al momento. Per Default non contenuta nella visualizzazione, può essere selezionata tramite il Menu contestuale-&gt; Selezione colonne.</td>
</tr>
</tbody>
</table>

**Informazioni su**

Nel dialogo di configurazione possono essere visualizzati solamente oggetti logici. Gli oggetti visuali non vengono visualizzati. Ulteriori informazioni sugli sfondi e come si fa a dinamicizzare gli oggetti visuali le potete leggere nella sezione Attribuzione zenon Oggetto per WPF Content (A pagina: 87).
Proprietà

Le proprietà permettono il collegamento di:

- Variabili (A pagina: 110)
- Valori (A pagina: 111)
- Autorizzazioni e interbloccchi (A pagina: 113)
### Parametri

<table>
<thead>
<tr>
<th>Nome</th>
<th>Descrizione</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nome</td>
<td>Nome della proprietà.</td>
</tr>
</tbody>
</table>
| Collegamento | Variabile collegata, autorizzazione o valore collegato.  
Cliccando sulla colonna, si apre il dialogo di selezione corrispondente a quanto indicato nella colonna **Tipo di collegamento**. |
| Tipo di collegamento | Selezione di un collegamento. |
| Info WPF  | Mostra l’attuale valore nel WPF-Content per le proprietà.  
Per l’utente è direttamente visibile di quale tipo di proprietà si tratta (boleana, string, ecc). |
| Collegato | Indica se una proprietà viene utilizzata al momento.  
Per Default non contenuta nella visualizzazione, può essere selezionata tramite il **Menu contestuale-> Selezionare colonne**. |

### CREA COLLEGAMENTO

Per creare un collegamento:

1. evidenziate anzitutto la riga con la proprietà che deve essere collegata
2. cliccate nella cella **Tipo di collegamento**
3. selezionate il collegamento desiderato fra quelli proposti dalla lista a cascata.

Sono a disposizione:

- **<non collegato>** (elimina un collegamento esistente)
- Autorizzazione/interblocco
- Variabile
- Collegamento valore

4. cliccate nella cella **Collegamento**
5. si apre il dialogo per la configurazione del collegamento desiderato
Informazioni su

Proprietà di WPF e di zenon possono divergere. Se viene collegata, ad esempio, la proprietà Visibilità, sono a disposizione in .NET tre valori:

0 - visibile
1 - invisibile
2 - einklappt

Questi valori devono essere rappresentati mediante la variabile di zenon collegata.

Collegamento variabile

Per collegare una variabile con una proprietà WPF:

1. evidenziate anzitutto la riga con la proprietà che deve essere collegata
2. cliccate nella cella Tipo di collegamento
3. selezionate dalla lista a cascata Variabile
4. cliccate nella cella Collegamento
5. si apre il dialogo per la configurazione della variabile

Questo dialogo vale anche per la selezione delle variabili per Trasformazioni (A pagina: 117). La configurazione permette anche la conversione di unità zenon in unità WPF.
<table>
<thead>
<tr>
<th>Parametri</th>
<th>Descrizione</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variabile collegata</td>
<td>Selezionare la variabile da collegare. Un clic sul comando ... apre il dialogo che serve alla selezione.</td>
</tr>
<tr>
<td>Intervallo valore dell’elemento WPF</td>
<td>Indicazioni per la conversione dei valori variabili in valori WPF.</td>
</tr>
<tr>
<td>Conversione intervallo valore</td>
<td><strong>Attivo:</strong> La conversione unità WPF è attiva.</td>
</tr>
<tr>
<td></td>
<td><strong>Effetto durante il Runtime</strong> Il valore di zenon (inclusa unità zenon) viene convertito per mezzo valore normato minimo e massimo nell’area WPF.</td>
</tr>
<tr>
<td></td>
<td><strong>Ad esempio:</strong> Il valore di una variabile varia da 100 a 200. Nelle variabili, l’area normata è impostata fra 100 e 200. L’obiettivo è quello di rappresentare delle modifiche mediante una manopola WPF. Per far ciò:</td>
</tr>
<tr>
<td></td>
<td>‣ alla voce Transformation viene collegata la proprietà RotateTransform.Angle con la variabile</td>
</tr>
<tr>
<td></td>
<td>‣ viene attivato adatta valore</td>
</tr>
<tr>
<td></td>
<td>‣ viene configurata un’area valori WPF da 0 a 360</td>
</tr>
<tr>
<td></td>
<td>Adesso la manopola sarà girata in corrispondenza di un valore di, per esempio, 150, di 180 gradi.</td>
</tr>
<tr>
<td>Minimo</td>
<td>definisce il valore WPF minimo.</td>
</tr>
<tr>
<td>Massimo</td>
<td>definisce il valore WPF massimo.</td>
</tr>
<tr>
<td>OK</td>
<td>Riprende le impostazioni e chiude il dialogo.</td>
</tr>
<tr>
<td>Annulla</td>
<td>Rifiuta le impostazioni e chiude il dialogo.</td>
</tr>
<tr>
<td>Guida</td>
<td>Attiva la guida online</td>
</tr>
</tbody>
</table>

**Collega valori**

Valori collegati possono essere o una string, oppure un valore numerico del tipo double. Quando si apre l’immagine, viene settato il valore scelto nel WPF content, dopo che quest’ultimo è stato caricato.
Attenzione

Il tipo di dato della proprietà WPF non deve essere necessariamente Double oppure String. Ma zenon setta solamente dei valori del tipo string oppure double. Essi devono essere convertiti in pagina WPF da .NET. Dettagli su questo argomento li trovate nel capitolo Trasmissione valore da zenon a WPF (A pagina: 85).

Per collegare una valore con una proprietà WPF:

1. evidenziate anzitutto la riga con la proprietà che deve essere collegata
2. cliccate nella cella Tipo di collegamento
3. selezionate nel Combobox l’opzione Collegamenti valori
4. cliccate nella cella Collegamento
5. si apre il dialogo per la configurazione del collegamento valori
### Parametri

<table>
<thead>
<tr>
<th>Parametri</th>
<th>Descrizione</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valore collegato:</td>
<td>Inserimento di un valore numerico o di un valore stringa.</td>
</tr>
<tr>
<td>Utilizza Stringa</td>
<td><strong>Attivo:</strong> un valore string viene usato al posto di un valore numerico. I valori stringa sono convertibili per quello che concerne la lingua. Durante il runtime, al momento dell’apertura dell’immagine il testo viene tradotto e salvato in WPF content. Se si effettua un cambio lingua mentre l’immagine è aperta, il valore stringa viene tradotto di nuovo e salvato.</td>
</tr>
<tr>
<td>Valore stringa/numerico</td>
<td>A second che sia scelta la proprietà usa string o meno, viene inserito in questo campo un valore numerico o string. Nel caso di valori numerici, può essere inoltre selezionata un’unità di misura.</td>
</tr>
<tr>
<td>Unità:</td>
<td>Selezione di un’unità di misura della lista a cascata. Essa deve essere stata definita nel Cambio unità.</td>
</tr>
<tr>
<td></td>
<td>L’unità di misura viene assegnata al valore numerico. Se durante il runtime viene eseguito un cambio unità, il valore viene convertito nella nuova unità di misura e salvato nel content WPF.</td>
</tr>
<tr>
<td>OK</td>
<td>Riprende le impostazioni e chiude il dialogo.</td>
</tr>
<tr>
<td>Annulla</td>
<td>Rifiuta le impostazioni e chiude il dialogo.</td>
</tr>
<tr>
<td>Guida</td>
<td>Attiva la guida online</td>
</tr>
</tbody>
</table>

### Collega autorizzazione o interblocco

Le autorizzazioni possono essere assegnate per tutto l’elemento WPF. All’elemento viene assegnato un livello utente. Le autorizzazioni vengono concesse nell’ambito del livello utente per singoli control. Se un’autorizzazione è attiva, viene scritto il valore 1 sulla proprietà nell’elemento.

Per collegare un’autorizzazione o un interblocco con una proprietà WPF:

1. evidenziate anzitutto la riga con la proprietà che deve essere collegata
2. cliccate nella cella **Tipo di collegamento**
3. selezionate nella lista a cascatà **Autorizzazione/Interblocco**
4. cliccate nella cella **Collegamento**

5. si apre il dialogo per la configurazione della autorizzazione

**Parametri** | **Descrizione**
--- | ---
Collega autorizzazione/interblocco | Determinazione dell’autorizzazione
Stato collegato | Selezione di un’autorizzazione che viene collegata ad un control WPF fra quelle della lista cascata. Ad esempio: la visibilità e la gestione di un pulsante WPF può dipendere dallo stato dell’utente.

**Autorizzazione** | **Descrizione**
--- | ---
Livelli di autorizzazione disponibili | Se l’utente possiede dei diritti sufficienti per usare l’elemento WPF, il valore 1 viene scritto sulla proprietà.
Nessuna autorizzazione | Se l’utente possiede dei diritti sufficienti per usare l’elemento WPF, il valore 1 viene scritto sulla proprietà.
Non bloccato | Se l’elemento non viene bloccato mediante un blocco, il valore 1 viene scritto sulla proprietà.
Bloccato | Se l’elemento viene bloccato mediante un blocco, il valore 1 viene scritto sulla proprietà.
Accessibile | Se esiste l’autorizzazione e l’elemento non è bloccato, il valore 1 viene scritto sulla proprietà.
Non operabili | Se manca l’autorizzazione o se l’elemento è bloccato, il valore 1 viene scritto sulla proprietà.
Eventi

Gli eventi permettono di collegare delle funzioni di zenon con un elemento WPF.
<table>
<thead>
<tr>
<th>Parametri</th>
<th>Descrizione</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nome</td>
<td>Nome della proprietà.</td>
</tr>
<tr>
<td>Collegamento</td>
<td>Funzione collegata. Cliccando nella cella, apriete il dialogo di configurazione.</td>
</tr>
<tr>
<td>Tipo di collegamento</td>
<td>Selezione di un collegamento. Cliccando nella cella, apriete il dialogo di selezione.</td>
</tr>
<tr>
<td>Info WPF</td>
<td>Mostra l’attuale valore nel WPF-Content per le proprietà. Per l’utente è direttamente visibile di quale tipo di proprietà si tratta (boleana, string, ecc).</td>
</tr>
<tr>
<td>Collegato</td>
<td>Indica se una proprietà viene utilizzata al momento. Per Default non contenuta nella visualizzazione, può essere selezionata tramite il Menu contestuale-&gt; Selezionare colonne.</td>
</tr>
</tbody>
</table>

**COLLEGA FUNZIONI**

Per creare un collegamento:

1. evidenziate anzitutto la riga con la proprietà che deve essere collegata
2. cliccate nella cella Tipo di collegamento
3. selezionate dalla lista a cascata Funzione
4. cliccate nella cella Collegamento
5. si apre il dialogo per la configurazione della funzione
<table>
<thead>
<tr>
<th>Parametri</th>
<th>Descrizione</th>
</tr>
</thead>
<tbody>
<tr>
<td>Funzione collegata</td>
<td>Selezionare la funzione da collegare. Cliccare sul pulsante ... e si apre il dialogo per la selezione della funzione.</td>
</tr>
<tr>
<td>OK</td>
<td>Riprende e accetta la selezione e chiude il dialogo.</td>
</tr>
<tr>
<td>Annulla</td>
<td>Rifiuta le modifiche e chiude il dialogo.</td>
</tr>
<tr>
<td>Guida</td>
<td>Apre la guida online</td>
</tr>
</tbody>
</table>

**Trasformazione**

L'elemento WPF non supporta alcuna funzionalità di rotazione. Se, per esempio, l'elemento WPF si trova in un simbolo, e quest'ultimo viene ruotato, l'elemento WPF non partecipa alla rotazione. Proprio per questo motivo, per WPF è stato introdotto un meccanismo proprio, detto trasformazione per consentire la rotazione oppure una trasformazione degli elementi. Queste trasformazioni vengono configurate nella scheda Trasformazione.
**Attenzione:** se `Content` si trova al di là dell’area dell’elemento WPF, questa parte del `content` va persa, oppure non viene disegnata.
## Parametri

<table>
<thead>
<tr>
<th><strong>Nome</strong></th>
<th><strong>Descrizione</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Nome</td>
<td>Nome della proprietà.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Collegamento</strong></th>
<th><strong>Descrizione</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Selezione delle variabili correlate.</td>
<td>Le trasformazioni vengono visualizzate in XAML come oggetti di trasformazione con proprietà proprie. Se un elemento supporta una trasformazione, le possibili proprietà degli oggetti di trasformazione vengono rappresentate nella visuale a lista. (Ulteriori informazioni in merito: integra pulsante come WPF-XAML in zenon. (A pagina: 137) Se la variabile collegata viene settata, ad esempio, sul valore 10, questo valore viene scritto sulla destinazione WPF e l’elemento viene ruotato di 10°.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Tipo di collegamento</strong></th>
<th><strong>Descrizione</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Selezione del tipo di collegamento trasformazione.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Info WPF</strong></th>
<th><strong>Descrizione</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mostra l’attuale valore nel WPF-Content per le proprietà. Per l’utente è direttamente visibile di quale tipo di proprietà si tratta (boleana, string, ecc).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Collegato</strong></th>
<th><strong>Descrizione</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Indica se una proprietà viene utilizzata al momento.</td>
<td>Per Default non contenuta nella visualizzazione, può essere selezionata tramite il Menu contestuale-&gt; Seleziona colonne.</td>
</tr>
</tbody>
</table>

## COLLEGA TRASFORMAZIONE

Per collegare una trasformazione utente con una proprietà WPF:

1. evidenziate anzitutto la riga con la proprietà che deve essere collegata
2. cliccate nella cella **Tipo di collegamento**
3. selezionate dalla lista a cascata **trasformazione**
4. cliccate nella cella **Collegamento**
5. si apre il dialogo per la configurazione della variabile
La configurazione permette anche la conversione di unità zenon in unità WPF.
### Parametri

<table>
<thead>
<tr>
<th>Parametro</th>
<th>Descrizione</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variabile collegata</td>
<td>Selezione della variabile da collegare. Un clic sul comando ... apre il dialogo che serve alla selezione.</td>
</tr>
<tr>
<td>Intervallo valore dell'elemento WPF</td>
<td>Indicazioni per la conversione dei valori variabili in valori WPF</td>
</tr>
<tr>
<td>Conversione intervallo valore</td>
<td><strong>Attivo:</strong> La conversione unità WPF è attiva. <strong>Effetto durante il Runtime</strong> Il valore di zenon (inclusa unità zenon) viene convertito per mezzo valore normato minimo e massimo nell’area WPF.</td>
</tr>
<tr>
<td></td>
<td><strong>Ad esempio:</strong> Il valore di una variabile varia da 100 a 200. Nelle variabili, l’area normata è impostata fra 100 e 200. L’obiettivo é quello di rappresentare delle modifiche mediante una manopola WPF. Per far ciò:</td>
</tr>
<tr>
<td></td>
<td>▶ alla voce Transformation viene collegata la proprietà RotateTransform.Angle con la variabile</td>
</tr>
<tr>
<td></td>
<td>▶ viene attivato adatta valore</td>
</tr>
<tr>
<td></td>
<td>▶ viene configurata un’area valori WPF da 0 a 360</td>
</tr>
<tr>
<td></td>
<td>Adesso la manopola sarà girata in corrispondenza di un valore di, per esempio, 150, di 180 gradi.</td>
</tr>
<tr>
<td>Minimo</td>
<td>definisce il valore WPF minimo.</td>
</tr>
<tr>
<td>Massimo</td>
<td>definisce il valore WPF massimo.</td>
</tr>
<tr>
<td><strong>OK</strong></td>
<td>Riprende le impostazioni e chiude il dialogo.</td>
</tr>
<tr>
<td><strong>Annulla</strong></td>
<td>Rifiuta le impostazioni e chiude il dialogo.</td>
</tr>
<tr>
<td><strong>Guida</strong></td>
<td>Attiva la guida online</td>
</tr>
</tbody>
</table>

### 6.3.4 Validità di file XAML

La validità di file XAML è legata ad alcuni proesupposti:
- Namespace corretto
- nessuna referenza class
- Scalabilità

NAMESPACE CORRETTO
L’elemento WPF può visualizzare esclusivamente WPF-Content, il che significa:

Solamente i file XAML con un namespace WPF corretto, possono essere visualizzati nell’elemento WPF. I file che usano un Silverlight-Namespace, non possono essere caricati o visualizzati in modo corretto. Nella maggior parte dei casi è però sufficiente sostituire il Silverlight-Namespace con il WPF-Namespace.

WPF-Namespace:

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

NESSUN USO DI REFERENZE CLASS
Visto che i file XAML vengono caricati in modo dinamico, non è possibile usare i file XAML che contengono delle referenze di classe (chiave, „Class“ im Header). La funzionalità che è stata programmatà in un file C# proprio, non può essere usata.

SCALABILITÀ
Nel caso in cui il contenuto di un elemento WPF dovesse essere adattato alla grandezza dell’elemento WPF, i control dell’elemento WPF devono essere inscatolati in un control che offre questa funzionalità, ad esempio, una View Box. Inoltre bisogna far in modo che gli elementi Altezza e Largezza vengano configurati in automatico.

CONTROLLO DI CORRETEZZA DI UN FILE XAML.
Pe controllare se un file XAML ha il formato corretto:

- Apri un file XAML in Internet-Explorer
  - se si può aprire senza ulteriori plugins (Java o simili), questo file può essere caricato oppure visualizzato anche con alta sicurezza dazenon.
Nel caso in cui emergono dei problemi al momento di caricare il file, esso viene visualizzato in Internet-Explorer e emerge con chiarezza in quale riga si sono verificati i problemi.

Anche la scala può essere testata in questo modo: se il file non è strutturato in modo corretto, il contenuto si adatterà alla grandezza della finestra di Internet Explorer.

**MESSAGGIO DI ERRORE**

Se si usa in zenon un file non valido, al momento del caricamento del file stesso nell’elemento WPF viene visualizzata una segnalazione di errore nella finestra di emissione.

Ad esempio:

"Errore nel caricamento di xaml-Datei:C:\ProgramData\COPA-DATA\SQL\781b1352-59d0-437e-a173-08563c3142e9\FILES\zenon\custom\media\UserControl1.xaml

6.3.5 **Elementi predefiniti**

C’è una serie di elementi WPF che sono forniti con zenon, oppure possono essere scaricati dal Webshop.


Elementi:

- Visuale rotonda (A pagina: 125)
- Visuale stato di avanzamento (A pagina: 130)
- Bargraf verticale (A pagina: 131)
- Indicatore della temperatura (A pagina: 132)
- Orologio analogico (A pagina: 133)
- Slider universale (A pagina: 134)
SOSTITUISCI ASSEMBLY CON NUOVA VERSIONE

Può essere utilizzato per un elemento WPF sia nell’Editor che nel Runtime solo un Assembly a progetto. Se in un progetto esistono due versioni di un Assembly, verrà usato il file caricato per primo. Il sistema provvederà a chiedere all’utente quale versione deve essere utilizzata. Se si decide di continuare ad usare la versione utilizzata fino a quel momento, non sono necessarie ulteriori azioni. Se si sceglie una versione più recente, però, dovranno essere adattati tutti i file CDWPF corrispondenti in tutti i simboli e immagini in tutti i progetti.

Indicazioni concernenti la gestione multi-proietto: Se in un progetto un Assembly viene sostituito da una versione più recente, essa deve essere sostituita in tutti gli altri progetti che sono caricati nell’Editor o nel Runtime.
### Visualizzazione circolare - CircularGaugeControl

<table>
<thead>
<tr>
<th>Property</th>
<th>Funzione</th>
<th>Valore</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentValue</td>
<td>Valore attuale che deve essere visualizzato.</td>
<td>Double</td>
</tr>
<tr>
<td>IsReversed</td>
<td>Orientamento scale, in senso orario o antiorario.</td>
<td>Boolean</td>
</tr>
<tr>
<td>ElementFontFamily</td>
<td>Carattere dell’elemento.</td>
<td>Font</td>
</tr>
<tr>
<td>MinValue</td>
<td>Valore minimo della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>MaxValue</td>
<td>Valore massimo della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>ScaleRadius</td>
<td>Radio della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>ScaleStartAngle</td>
<td>Angolo in corrispondenza del quale inizia la scala.</td>
<td>Double</td>
</tr>
<tr>
<td>ScaleLabelRotationMode</td>
<td>Orientamento del titolo della scala.</td>
<td>Enum:</td>
</tr>
<tr>
<td></td>
<td>▶ None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ Autmatic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ Surround In</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ Surround Out</td>
<td></td>
</tr>
<tr>
<td>ScaleSweepAngle</td>
<td>Area angolo che definisce la dimensione della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>ScaleLabelFontSize</td>
<td>Dimensione carattere della denominazione della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>ScaleLabelColor</td>
<td>Colore del carattere della denominazione della scala.</td>
<td>Color</td>
</tr>
<tr>
<td>ScaleLabelRadius</td>
<td>Radio secondo il quale è orientata la denominazione della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>ScaleValuePrecision</td>
<td>Esattezza della denominazione della scala.</td>
<td>Integer</td>
</tr>
<tr>
<td>PointerStyle</td>
<td>Forma del pointer che indica il valore.</td>
<td>Enum:</td>
</tr>
<tr>
<td></td>
<td>▶ Arrow</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▶ Rectangle</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Triangle Cap</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pentagon</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Triangle</td>
<td></td>
</tr>
<tr>
<td>Elemento WPF</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>MajorTickColor</td>
<td>Colore delle suddivisioni principali della scala.</td>
<td>Color</td>
</tr>
<tr>
<td>MinorTickColor</td>
<td>Colore delle suddivisioni secondarie della scala.</td>
<td>Color</td>
</tr>
<tr>
<td>MajorTickSize</td>
<td>Dimensione delle suddivisioni principali della scala.</td>
<td>Size</td>
</tr>
<tr>
<td>MinorTickSize</td>
<td>Dimensione delle suddivisioni secondarie della scala.</td>
<td>Size</td>
</tr>
<tr>
<td>MajorTicksCount</td>
<td>Numero delle suddivisioni principali della scala.</td>
<td>Integer</td>
</tr>
<tr>
<td>MajorTicksShape</td>
<td>Forma/tipo delle suddivisioni principali della scala.</td>
<td>Enum: Rectangle, Trapezoid, Triangle</td>
</tr>
<tr>
<td>MinorTicksShape</td>
<td>Forma/tipo delle suddivisioni secondarie della scala.</td>
<td>Enum: Rectangle, Trapezoid, Triangle</td>
</tr>
<tr>
<td>MinorTicksCount</td>
<td>Numero delle suddivisioni secondarie della scala.</td>
<td>Integer</td>
</tr>
<tr>
<td>PointerSize</td>
<td>Dimensione del pointer.</td>
<td>Size</td>
</tr>
<tr>
<td>PointerCapRadius</td>
<td>Dimensione del punto di fissaggio del pointer.</td>
<td>Double</td>
</tr>
<tr>
<td>PointerBorderBrush</td>
<td>Colore della cornice del pointer</td>
<td>Brush</td>
</tr>
<tr>
<td>PointerCapStyle</td>
<td>Forma/tipo del punto di fissaggio del pointer.</td>
<td>Enum: BackCap, FrontCap, Screw</td>
</tr>
<tr>
<td>PointerCapBorderBrush</td>
<td>Colore del punto di fissaggio del pointer.</td>
<td>Brush</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>PointerBrush</td>
<td>Colore dell’indicatore.</td>
<td>Brush</td>
</tr>
<tr>
<td>GaugeBorderBrush</td>
<td>Colore della cornice dell’elemento.</td>
<td>Brush</td>
</tr>
<tr>
<td>GaugeBackgroundBrush</td>
<td>Colore dello sfondo dell’elemento.</td>
<td>Brush</td>
</tr>
<tr>
<td>PointerCapColorBrush</td>
<td>Colore del punto di fissaggio del pointer.</td>
<td>Brush</td>
</tr>
<tr>
<td>GaugeMiddlePlate</td>
<td>Raggio della placca intermedia dell’elemento di sfondo</td>
<td>Double</td>
</tr>
<tr>
<td>PointerOffset</td>
<td>Offset del pointer (spostamento).</td>
<td>Double</td>
</tr>
<tr>
<td>RangeRadius</td>
<td>Raggio dell’area totale di visualizzazione.</td>
<td>Double</td>
</tr>
<tr>
<td>RangeThickness</td>
<td>Spessore dell’area totale di visualizzazione.</td>
<td>Double</td>
</tr>
<tr>
<td>RangeStartValue</td>
<td>Valore iniziale dell’area totale di visualizzazione.</td>
<td>Double</td>
</tr>
<tr>
<td>Range1EndValue</td>
<td>Valore finale della 1° area e valore di start della 2° area.</td>
<td>Double</td>
</tr>
<tr>
<td>Range2EndValue</td>
<td>Valore finale della 2° area e valore di start della 3° area.</td>
<td>Double</td>
</tr>
<tr>
<td>Range3EndValue</td>
<td>Valore finale della 3° area e valore di start della 4° area.</td>
<td>Double</td>
</tr>
<tr>
<td>Range4EndValue</td>
<td>Valore finale della 4° area e valore di start della 5° area.</td>
<td>Double</td>
</tr>
<tr>
<td>Range5EndValue</td>
<td>Valore finale della 5° area e valore di start della 6° area.</td>
<td>Double</td>
</tr>
<tr>
<td>Range6EndValue</td>
<td>Valore finale della 6° area.</td>
<td>Double</td>
</tr>
<tr>
<td>Range1ColorBrush</td>
<td>Colore della 1° area.</td>
<td>Brush</td>
</tr>
<tr>
<td>Range2ColorBrush</td>
<td>Colore della 2° area.</td>
<td>Brush</td>
</tr>
<tr>
<td>Range3ColorBrush</td>
<td>Colore della 3° area.</td>
<td>Brush</td>
</tr>
<tr>
<td>Range4ColorBrush</td>
<td>Colore della 4° area.</td>
<td>Brush</td>
</tr>
<tr>
<td>Range5ColorBrush</td>
<td>Colore della 5° area.</td>
<td>Brush</td>
</tr>
<tr>
<td>Range6ColorBrush</td>
<td>Colore della 6° area.</td>
<td>Brush</td>
</tr>
<tr>
<td>ScaleOuterBorderBrush</td>
<td>Colore del bordo della scala.</td>
<td>Brush</td>
</tr>
<tr>
<td>ScaleBackgroundBrush</td>
<td>Colore dello sfondo della scala.</td>
<td>Brush</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>ValueTextStyle</td>
<td>Forma/tipo della visualizzazione valore.</td>
<td>Enum:</td>
</tr>
<tr>
<td></td>
<td>• LargeFrame</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• SmallFrame</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• None</td>
<td></td>
</tr>
<tr>
<td>ValueDialogContent</td>
<td>Contenuto della visualizzazione valore.</td>
<td>Enum:</td>
</tr>
<tr>
<td></td>
<td>• Testo</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TextValue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Value</td>
<td></td>
</tr>
<tr>
<td>ValueTextSize</td>
<td>Dimensione carattere della visualizzazione valore.</td>
<td>Double</td>
</tr>
<tr>
<td>ValueTextColor</td>
<td>Colore carattere della visualizzazione valore.</td>
<td>Color</td>
</tr>
<tr>
<td>IsGlassReflection</td>
<td>Attivazione dell’effetto glass reflection sull’elemento.</td>
<td>Boolean</td>
</tr>
<tr>
<td>GaugeOffsett</td>
<td>Abbassamento del punto di rotazione di tutto l’elemento.</td>
<td>Double</td>
</tr>
</tbody>
</table>
**Barra stato di avanzamento - ProgressBarControl**

<table>
<thead>
<tr>
<th>Property</th>
<th>Funzione</th>
<th>Valore</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentValue</td>
<td>Valore attuale che deve essere visualizzato.</td>
<td>Double</td>
</tr>
<tr>
<td>MinValue</td>
<td>Il valore minimo del campo valori.</td>
<td>Double</td>
</tr>
<tr>
<td>MaxValue</td>
<td>Il valore massimo del campo valori.</td>
<td>Double</td>
</tr>
<tr>
<td>ProgressBarDivisionCount</td>
<td>Numero delle suddivisioni della barra stato di avanzamento (Boxed).</td>
<td>Integer</td>
</tr>
<tr>
<td>VisibilityText</td>
<td>Visibilità della visualizzazione valore.</td>
<td>Boolean</td>
</tr>
<tr>
<td>TextSize</td>
<td>Dimensione carattere della visualizzazione valore.</td>
<td>Double</td>
</tr>
<tr>
<td>TextColor</td>
<td>Colore della visualizzazione valore.</td>
<td>Color</td>
</tr>
<tr>
<td>ProgressBarBoxedColor</td>
<td>Colore della cornice della barra stato di avanzamento (Boxed).</td>
<td>Color</td>
</tr>
<tr>
<td>ProgressBarMarginDistance</td>
<td>Distanza del Progressbar-Box dal bordo dell’elemento (sinistra, sopra destra, sotto).</td>
<td>Double</td>
</tr>
<tr>
<td>ProgressBarInactiveBrush</td>
<td>Colore indicatore non attivo.</td>
<td>Brush</td>
</tr>
<tr>
<td>ProgressBarActiveBrush</td>
<td>Colore indicatore attivo.</td>
<td>Brush</td>
</tr>
<tr>
<td>ProgressBarPadding</td>
<td>Distanza della barra di stato avanzamento dal Progressbar-Box (sinistra, sopra destra, sotto).</td>
<td>Double</td>
</tr>
<tr>
<td>ElementBorderBrush</td>
<td>Colore della cornice dell’elemento.</td>
<td>Brush</td>
</tr>
<tr>
<td>ElementBackgroundBrush</td>
<td>Colore dello sfondo dell’elemento.</td>
<td>Brush</td>
</tr>
<tr>
<td>Property</td>
<td>Funzione</td>
<td>Valore</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>CurrentValue</td>
<td>Valore attuale che deve essere visualizzato.</td>
<td>Double</td>
</tr>
<tr>
<td>MinValue</td>
<td>Valore minimo della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>MaxValue</td>
<td>Valore massimo della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>MajorTicksCount</td>
<td>Numero delle suddivisioni principali della scala.</td>
<td>Integer</td>
</tr>
<tr>
<td>MinorTicksCount</td>
<td>Numero delle suddivisioni secondarie della scala.</td>
<td>Integer</td>
</tr>
<tr>
<td>MajorTickColor</td>
<td>Colore delle suddivisioni principali della scala.</td>
<td>Color</td>
</tr>
<tr>
<td>MinorTickColor</td>
<td>Colore delle suddivisioni secondarie della scala.</td>
<td>Color</td>
</tr>
<tr>
<td>ElementBorderBrush</td>
<td>Colore della cornice dell’elemento.</td>
<td>Brush</td>
</tr>
<tr>
<td>ElementBackgroundBrush</td>
<td>Colore dello sfondo dell’elemento.</td>
<td>Brush</td>
</tr>
<tr>
<td>ElementGlasReflection</td>
<td>Attivazione dell’effetto glas reflection sull’elemento.</td>
<td>Visibilità</td>
</tr>
<tr>
<td>ElementFontFamily</td>
<td>Carattere dell’elemento.</td>
<td>Font</td>
</tr>
<tr>
<td>ScaleFontSize</td>
<td>Dimensione carattere della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>ScaleFontColor</td>
<td>Colore carattere della scala.</td>
<td>Color</td>
</tr>
<tr>
<td>IndicatorBrush</td>
<td>Colore di riempimento del bargraf.</td>
<td>Brush</td>
</tr>
<tr>
<td>BargraphSeparation</td>
<td>Numero dei trattini di separazione del bargraf.</td>
<td>Integer</td>
</tr>
<tr>
<td>BargraphSeparationColor</td>
<td>Colore delle suddivisione della scala.</td>
<td>Color</td>
</tr>
</tbody>
</table>
### Indicazione della temperatura - TemperatureIndicatorControl

<table>
<thead>
<tr>
<th>Property</th>
<th>Funzione</th>
<th>Valore</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentValue</td>
<td>Valore attuale che deve essere visualizzato.</td>
<td>Double</td>
</tr>
<tr>
<td>MinValue</td>
<td>Valore minimo della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>MaxValue</td>
<td>Valore massimo della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>MajorTicksCount</td>
<td>Numero delle suddivisioni principali della scala.</td>
<td>Integer</td>
</tr>
<tr>
<td>MinorTicksCount</td>
<td>Numero delle suddivisioni secondarie della scala.</td>
<td>Integer</td>
</tr>
<tr>
<td>TickNegativColor</td>
<td>Colore della suddivisione principale negativa (che passa gradualmente a TickPositivColor).</td>
<td>Color</td>
</tr>
<tr>
<td>TickPositivColor</td>
<td>Colore della suddivisione principale positiva (che passa gradualmente a TickNegativColor).</td>
<td>Color</td>
</tr>
<tr>
<td>MinorTickColor</td>
<td>Colore delle d secondarie.</td>
<td>Color</td>
</tr>
<tr>
<td>ElementBorderBrush</td>
<td>Colore della cornice dell’elemento.</td>
<td>Brush</td>
</tr>
<tr>
<td>ElementBackgroundBrush</td>
<td>Colore dello sfondo dell’elemento.</td>
<td>Brush</td>
</tr>
<tr>
<td>ElementGlasReflection</td>
<td>Attivazione dell’effetto glas reflection sull’elemento.</td>
<td>Visibility</td>
</tr>
<tr>
<td>ElementFontFamily</td>
<td>Carattere dell’elemento.</td>
<td>Font</td>
</tr>
<tr>
<td>IndicatorColor</td>
<td>Colore di riempimento dell’indicatore.</td>
<td>Color</td>
</tr>
<tr>
<td>IndicatorBorderColor</td>
<td>Colore della cornice dell’indicatore.</td>
<td>Color</td>
</tr>
<tr>
<td>MajorTickSize</td>
<td>Dimensione delle suddivisioni principali della scala.</td>
<td>Size</td>
</tr>
<tr>
<td>MinorTickSize</td>
<td>Dimensione delle suddivisioni secondarie della scala.</td>
<td>Size</td>
</tr>
<tr>
<td>ScaleLetteringDistance</td>
<td>Distanza della didascalia della scala (verticale); bisogna scrivere una didascalia per ogni x suddivisione principale.</td>
<td>Integer</td>
</tr>
<tr>
<td>IndicatorScaleDistance</td>
<td>Distanza fra indicatore e scala (orizzontale).</td>
<td>Double</td>
</tr>
<tr>
<td>ScaleFontSize</td>
<td>Dimensione carattere della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>ScaleFontColor</td>
<td>Colore carattere della scala.</td>
<td>Color</td>
</tr>
<tr>
<td>Unit</td>
<td>Unità.</td>
<td>String</td>
</tr>
</tbody>
</table>
**Elemento WPF**

<table>
<thead>
<tr>
<th>Property</th>
<th>Funzione</th>
<th>Valore</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ElementStyle</strong></td>
<td>Forma/tipo dell’elemento.</td>
<td>Enum:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>› SmallFrame</td>
</tr>
<tr>
<td></td>
<td></td>
<td>› Unit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>› None</td>
</tr>
</tbody>
</table>

**Orologio analogico - AnalogClockControl**

<table>
<thead>
<tr>
<th>Property</th>
<th>Funzione</th>
<th>Valore</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ElementStyle</strong></td>
<td>Forma/tipo dell’elemento.</td>
<td>Enum:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>› SmallNumbers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>› BigNumbers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>› No</td>
</tr>
<tr>
<td><strong>ElementBackgroundBrush</strong></td>
<td>Colore dello sfondo dell’elemento.</td>
<td>Brush</td>
</tr>
<tr>
<td><strong>ElementGlasReflection</strong></td>
<td>Attivazione dell’effetto glas reflection sull’elemento.</td>
<td>Visibility</td>
</tr>
<tr>
<td><strong>Offset</strong></td>
<td>Valore in ore (h) che rappresenta la differenza di tempo rispetto all’orologio di sistema.</td>
<td>Int16</td>
</tr>
<tr>
<td><strong>OriginText</strong></td>
<td>Testo visualizzato nell’orologio (per es. indicazione del luogo).</td>
<td>String</td>
</tr>
</tbody>
</table>
### Slider universale - UniversalReglerControl

<table>
<thead>
<tr>
<th>Property</th>
<th>Funzione</th>
<th>Valore</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentValue</td>
<td>Valore attuale che deve essere visualizzato.</td>
<td>Double</td>
</tr>
<tr>
<td>ElementFontFamily</td>
<td>Carattere dell’elemento.</td>
<td>Font</td>
</tr>
<tr>
<td>MinValue</td>
<td>Valore minimo della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>MaxValue</td>
<td>Valore massimo della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>Raggio</td>
<td></td>
<td>Double</td>
</tr>
<tr>
<td>ScaleRadius</td>
<td>Radio della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>ScaleStartAngle</td>
<td>Angolo in corrispondenza del quale inizia la scala.</td>
<td>Double</td>
</tr>
<tr>
<td>ScaleLabelRotationMode</td>
<td>Orientamento del titolo della scala.</td>
<td>Enum:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>None, Automatic, SurroundIn, SurroundOut</td>
</tr>
<tr>
<td>ScaleSweepAngle</td>
<td>Area angolo che definisce la dimensione della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>ScaleLabelFontSize</td>
<td>Dimensione carattere della denominazione della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>ScaleLabelColor</td>
<td>Colore del carattere della denominazione della scala.</td>
<td>Color</td>
</tr>
<tr>
<td>ScaleLabelRadius</td>
<td>Radio secondo il quale è orientata la denominazione della scala.</td>
<td>Double</td>
</tr>
<tr>
<td>ScaleValuePrecision</td>
<td>Esattezza della denominazione della scala.</td>
<td>Integer</td>
</tr>
<tr>
<td>ElementStyle</td>
<td>Tipo di visualizzazione dell’elemento:</td>
<td>Enum:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Knob, Plate, None</td>
</tr>
<tr>
<td>MajorTickColor</td>
<td>Colore delle suddivisioni principali della scala.</td>
<td>Color</td>
</tr>
<tr>
<td>MinorTickColor</td>
<td>Colore delle suddivisioni secondarie della scala.</td>
<td>Color</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>MajorTickSize</td>
<td>Dimensione delle suddivisioni principali della scala.</td>
<td>Size</td>
</tr>
<tr>
<td>MinorTickSize</td>
<td>Dimensione delle suddivisioni secondarie della scala.</td>
<td>Size</td>
</tr>
<tr>
<td>MajorTicksCount</td>
<td>Numero delle suddivisioni principali della scala.</td>
<td>Integer</td>
</tr>
<tr>
<td>MajorTicksShape</td>
<td>Forma/tipo delle suddivisioni principali della scala.</td>
<td>Enum:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Rectangle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Trapezoid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Triangle</td>
</tr>
<tr>
<td>MinorTicksShape</td>
<td>Forma/tipo delle suddivisioni secondarie della scala.</td>
<td>Enum:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Rectangle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Trapezoid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Triangle</td>
</tr>
<tr>
<td>MinorTicksCount</td>
<td>Numero delle suddivisioni secondarie della scala.</td>
<td>Integer</td>
</tr>
<tr>
<td>BackgroundBorderBrush</td>
<td>Colore della cornice dell’elemento.</td>
<td>Brush</td>
</tr>
<tr>
<td>Elemento WPF</td>
<td>Descrizione</td>
<td>Tipo</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>BackgroundBrush</td>
<td>Colore dello sfondo dell’elemento.</td>
<td>Brush</td>
</tr>
<tr>
<td>PointerCapColorBrush</td>
<td>Colore del punto di fissaggio del pointer.</td>
<td>Brush</td>
</tr>
<tr>
<td>GaugeMiddlePlate</td>
<td>Raggio della placca intermedia dell’elemento di sfondo</td>
<td>Double</td>
</tr>
<tr>
<td>ValueFontSize</td>
<td>Dimensione carattere della visualizzazione valore.</td>
<td>Double</td>
</tr>
<tr>
<td>ValueFontColor</td>
<td>Colore carattere della visualizzazione valore.</td>
<td>Color</td>
</tr>
<tr>
<td>IsGlasReflection</td>
<td>Attivazione dell’effetto glas reflection sull’elemento.</td>
<td>Boolean</td>
</tr>
<tr>
<td>KnobBrush</td>
<td>Colore del bottone.</td>
<td>Brush</td>
</tr>
<tr>
<td>IndicatorColorBrush</td>
<td>Colore dell’indicatore.</td>
<td>Brush</td>
</tr>
<tr>
<td>IndicatorBackgroundBrush</td>
<td>Colore di sfondo dell’indicatore non attivo.</td>
<td>Brush</td>
</tr>
<tr>
<td>KnobSize</td>
<td>Diametro del bottone.</td>
<td>Double</td>
</tr>
<tr>
<td>KnobIndicatorSize</td>
<td>Dimensione indicatore del bottone.</td>
<td>Size</td>
</tr>
<tr>
<td>ElementSize</td>
<td>Grandezza dell’elemento.</td>
<td>Size</td>
</tr>
<tr>
<td>VisibilityKnob</td>
<td>Attivazione del bottone.</td>
<td>Boolean</td>
</tr>
<tr>
<td>ValuePosition</td>
<td>Posizione della visualizzazione valore.</td>
<td>Double</td>
</tr>
<tr>
<td>ValueVisibility</td>
<td>Attivazione della visualizzazione valore.</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

### 6.3.6 Esempio: Integrazione di WPF in zenon

Come vengono creati i file XAML e integrati in zenon come elementi WPF, lo potete vedere nei seguenti esempi:

- Integra pulsante come WPF-XAML in zenon (A pagina: 137)
- Integra bargraf come WPF-XAML in zenon (A pagina: 144)
- Integrare DataGrid Control in zenon (A pagina: 149)
Integra pulsante come WPF-XAML in zenon

Struttura dell’esempio:

- Creazione di un pulsante (A pagina: 90) in Microsoft Expression Blend
- integra in zenon
- Collegamento di una variabile o con una funzione.
- Adattamento del pulsante alla grandezza dell’elemento.
- Crea un pulsante

Create in primo luogo un pulsante come descritto nel capitolo Crea pulsante come file XAML con Microsoft Expression Blend (A pagina: 90). Per usare il file XAML in zenon, aggiungetelo nella visione ad albero del progetto nella cartella File/grafiche.

INTEGRA PULSANTE

Nota: per la descrizione seguente viene usato un progetto di zenon con i seguenti contenuti:

- un’immagine vuota come immagine di partenza
- una variabile interna int del tipo Int
- una funzione Funktion_0 del tipo setta valore impostabile con:
  - Attiva opzione Direttamente su Hardware
  - Valore impostabile settato a 45

Per integrare il pulsante:

1. apriete anzitutto l’immagine vuota
2. piazzate un elemento WPF (A pagina: 104) nell’immagine
3. selezionate nella finestra delle proprietàFile XAML
4. selezionate il file XAML (per es.: MyButton.xaml) e chiudete il dialogo
5. selezionate la proprietà Configurazione
CONFIGURA PULSANTE

Il dialogo di configurazione mostra un’anteprima del file XAML selezionato. Nella struttura ad albero c’è una lista di tutti gli elementi nominati nel file XAML:

1. selezionate il pulsante WPF, che si trova in LayoutRoot->MyViewBox->MyButton
2. Cericate nella scheda Proprietà l’inserimento Content; esso contiene il testo del pulsante.
3. cliccate nella colonna Tipo di collegamento
4. Selezionate nella lista a cascata l’inserimento Variabile.
5. cliccate nella colonna Collegamento
6. Si apre il dialogo di selezione variabili.
7. selezionate la variabile int, per collegarla alla proprietà Content.

EVENTI

Per assegnare anche eventi:
1. Selzionate la scheda Eventi.

2. Cercate l’inserimento Click; questo evento viene avviato dall’elemento WPF non appena si clicca sul pulsante.

3. Cliccate nella colonna Tipo di collegamento

4. Selezionate nella lista a cascata l’inserimento Funzione.

5. Cliccate nella colonna Collegamento

6. Si apre il Dialogo di selezione funzioni.

7. Selzionate Funktion_0

8. Confermate la modifica effettuata con OK.

9. Aggiungete nell’immagine ancora un Elemento Valore numerico

10. Collegate anche questo Valore numerico alla variabile int.

11. Poi compilate i file runtime e avviate il runtime.
Nel runtime viene visualizzato l’elemento WPF; il testo del pulsante è 0. Non appena cliccate sul pulsante, viene avviato l’evento `Click` e la funzione `Setta valore impostabile` viene eseguita. Il valore 45 viene settato direttamente sull’hardware e sia il Valore numerico che il Pulsante indicano il valore 45.

Settate per mezzo dell’elemento *Valore numerico* un valore impostabile di 30; in tal modo questo valore verrà ripreso anche dall’elemento WPF.

**AUTORIZZAZIONE**

In modo analogo di quanto accade con un *Valore numerico*, un elemento WPF può essere bloccato (simbolo del lucchetto) oppure reso disponibile a seconda dell’autorizzazione di gestione. Settate a 1 il livello autorizzazione dell’elemento WPF e impostate un utente *Test* con livello di autorizzazione 1. Poi impostate le funzioni *Login* con dialogo e *Logout*. Collegate queste due funzioni con 2 nuovi pulsanti con testo nell’immagine.
Nel dialogo di configurazione dell’elemento WPF, selezionate il pulsante WPF MyButton e scegliete il tab Proprietà

1. Selezionate l’elemento IsEnabled
2. Cliccate nella colonna Tipo di collegamento
3. Selezionate nella lista a cascata l’inserimento Autorizzazione di gestione/Blocco.
4. Cliccate nella colonna Collegamento
5. Selezionate nella lista a cascata l’opzione Autorizzazione di gestione esistente.
6. Chiudete il dialogo cliccando su OK.
Compilate i file runtime e fate attenzione a selezionare anche le Autorizzazioni di gestione per trasmissione. Dopo l’avvio del runtime, il pulsante WPF dell’immagine viene visualizzato come non attivo ed è inutilizzabile. Se effettuate il Login con l’utente Test, il pulsante viene attivato e può essere utilizzato. Non appena effettuate il Logout, il pulsante viene bloccato.

TRASFORMAZIONE

Per poter usare delle trasformazioni, dovete adattare il file XAML.

1. Passate al programma Expression Blend
2. Selezionate MyButton, in modo tale che le proprietà del pulsante siano visibili nella finestra delle proprietà.
3. Selezionate alla voce Transform di RenderTransform l’opzione Apply relative transform

In tal modo, viene aggiunto un blocco al file XAML; esso salva durante il runtime le impostazioni per la trasformazione.

```
<Button RenderTransform=
  <TransformGroup>
    <ScaleTransform ScaleX="1" ScaleY="1"/>
    <SkewTransform AngleX="0" AngleY="0"/>
    <RotateTransform Angle="0"/>
    <TranslateTransform X="0" Y="0"/>
  </TransformGroup>
</Button RenderTransform>
```

4. Salvate il file e sostituite nel progetto zenon la vecchia versione con questo nuovo file.
5. **aprite di nuovo il dialogo di configurazione dell’elemento WPF:**

   a) **Selezionate il pulsante** MyButton.
   b) **Passate alla scheda** Trasformazioni
   c) **Selezionate l’elemento** RotateTransform.Angle
   d) **cliccate nella colonna** Tipo di collegamento
   e) **Selezionate nella lista a cascata** l’inserimento Trasformazioni.
   f) **cliccate nella colonna** Collegamento
   g) Si apre il dialogo di selezione variabili.
   h) **selezionate la variabile** int, per collegarla alla proprietà RotateTransform.Angle.
Poi compilate i file runtime e avviare il runtime. Effettuate il Login come utente Test e cliccate sul pulsante. Il pulsante riceve il valore 45 e l’elemento WPF ruota di 45°.

Integra bargraf come WPF-XAML in zenon

Struttura dell’esempio:

- Creazione di una bargraf (A pagina: 94) in Adobe Illustrator e trasformazione in WPF
- integra in zenon
- Collegamento con variabile
- adatta l’elemento bargraf-WPF

CREA BARGRAF

Create in primo luogo un bargraf come descritto nel capitolo Workflow con Adobe Illustrator (A pagina: 94). Per usare il file XAML in zenon, aggiungetelo nella visione ad albero del progetto nella cartella File/grafiche.

INTEGRA BARGRAF

Nota: per la descrizione seguente viene usato un progetto di zenon con i seguenti contenuti:

- un’immagine vuota come immagine di partenza
- 4 variabili del driver interno per
  - scala 0
  - scala media
- scala alta
- valore attuale
  - una variabile del driver matematico per la visualizzazione del valore attuale 255)

Per integrare il bargraf:

1. apri anzitutto l’immagine vuota
2. piazza un elemento WPF (A pagina: 104) nell’immagine
3. seleziona nella finestra delle proprietà File XAML
4. seleziona il file XAML desiderato (per esempio bargraph_vertical.xaml) e chiude il dialogo

**ADATTA BARGRAF**

Prima della configurazione, la scala del file XAML viene adattata, se necessario.

A tale scopo:

- impostate una nuova variabile matematica che calcola il nuovo valore in riferimento alla scala, per esempio:
- Variabile: 0-1000
- variabile matematica (valore creato in xaml)*Variabile/1000

Poi viene configurato il file XAML.

**CONFIGURA BARGRAF**

1. cliccate sull’elemento WPF e selezionate la proprietà Configurazione
2. Il dialogo di configurazione mostra un’anteprima del file XAML selezionato
3. selezionate il valore minimo, quello medio e quello massimo e collegate la variabile corrispondente nella proprietà `Content`
4. selezionate lo slider e collegate la proprietà `Value` con la variabile matematica (nel nostro esempio: `calculation`
5. Controllate la progettazione nel runtime:

Integrare DataGrid Control in zenon

Per creare un DataGrid Control, avete bisogno di:

- Visual Studio

Fate attenzione di creare sempre dei progetti che si basino su .NET Framework 3.5.

CREARE WPF USER CONTROL

Nel nostro esempio, esso riceve il nome MyWPFLibrary.

2. Aggiungete i WPF Toolkit Asemblies alle referenze. A tale scopo:
   a) cliccate con il tasto destro del mouse sul progetto.
   b) selezionate Add reference...
   c) selezionate nella Tab .NET

3. Create una nuova data connection nel server Explorer, A tale scopo:
   a) cliccate con il tasto destro del mouse su Data Connections
   b) selezionate Add connection...

Il nostro esempio usa la banca dati Northwind che vien messa a disposizione da Microsoft come esempio di database.
Dopo aver aggiunto una connessione, la finestra del Server Explorer avrà più o meno il seguente aspetto:

![Server Explorer](image)

Il prossimo passo consiste nella creazione di un data set.

**CREAZIONE DI UN DATA SET.**

1. Cliccate con il tasto destro del mouse sul progetto.
2. Selezionate nel menù contestuale **Add – New Item**...
3. Impostate un nuovo data set.
5. Trascinate le tabelle di cui avete bisogno nella finestra Design.

Il prossimo passo consiste nella configurazione del file XAML.

**CONFIGURAZIONE DEL FILE XAML**

1. Aggiungete i *namespaces* nel file XAML.
   
   Avete bisogno del *namespace* del WPF Toolkit e una referenza alla classe:

   ```xml
   <UserControl xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
   presentation" xmlns:x="http://schemas.microsoft.com/winfx/2006/xam1" 
   xmlns:my1="http://schemas.microsoft.com/microsoft/2008/wpf/modeling" 
   xmlns:my2="http://schemas.microsoft.com/microsoft/2008/wpf/modeling" 
   xmlns:my="http://schemas.microsoft.com/microsoft/2008/wpf/modeling"
   mc:Ignorable="d">
   <UserControl.Resources>
     <my1:MyDataSet x:Key="MyDataSet"/>
     <CollectionViewSource x:Key="customersViewSource" Source="{Binding Path=Customers,
     Source={StaticResource MyDataSet}}" />
   </UserControl.Resources>
   <Grid DataContext="{StaticResource customersViewSource}"
   Height="304" HorizontalAlignment="Left" Margin="6,7,0,0"
   Name="dataGrid1" VerticalAlignment="Top" Width="497"/>
DisplayMemberPath="CompanyName" ItemsSource="{Binding}"
SelectedValuePath="CustomerID" />
</Grid>

3. Aprire il file code-behind (xaml.cs) e aggiungere nel constructor le seguenti righe:

```csharp
public UserControl1()
{
    InitializeComponent();
    MyWPFLibrary.MyDataSet ds = (MyWPFLibrary.MyDataSet)(this.FindResource("MyDataSet"));
    MyWPFLibrary.MyDataSetTableAdapters.CustomersTableAdapter ta = new MyWPFLibrary.MyDataSetTableAdapters.CustomersTableAdapter();
    ta.Fill(ds.Customers);
    customersViewSource.View.MoveCurrentToFirst();
}
```

Ciò ha i seguenti esiti:
- si va a prendere il data set
- si crea nuovo TableAdapter
- si riempie il data set
- si mette a disposizione questa informazione al DataGrid Control

Adesso si può realizzare la soluzione.

**BUILD**

Create adesso la soluzione. Nella cartella Output vengono create durante il processo alcune DLL.

Adesso disponete delle DLL che offrono le funzionalità necessarie. Tuttavia, zenon può visualizzare solamente file XAML che non possono essere collegati al file code-behind file. Per questa ragione, è necessaria un’ulteriore DLL che referenzi la DLL che è stata appena creata. A tale scopo:

1. create un altro progetto, un’altra WPF User Control Library.
2. Nel nostro esempio, essa viene nominata DataGridControl.

3. Aggiungete in questi nuovo progetto una referenza al progetto appena creato.

4. Il file XAML ha il seguente aspetto:

```xml
<UserControl x:Class="test.UserControl1"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 mc:Ignorable="d"
 d:DesignHeight="300" d:DesignWidth="300">
 <Grid>
 </Grid>
</UserControl>
```

5. Visto che tutti i contenuti necessari sono presenti nella DLL e che non si ha bisogno di un code-behind, cancellate:

   x:Class="test.UserControl1"

6. Cancellate anche (per il posizionamento) le seguenti righe:

   mc:Ignorable="d"

   d:DesignHeight="300" d:DesignWidth="300"

7. Definite quello che deve essere visualizzato nel file XAML. Poi aggiungete quanto segue:

```xml
<UserControl xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mywpflib="clr-namespace:MyWPFLibrary;assembly=MyWPFLibrary">
<Grid x:Name="GridName">
  <mywpflib:UserControl1 HorizontalAlignment="Left" Name="userControl11" VerticalAlignment="Top"/>
</Grid>
</UserControl>

La riga `xmlns:mywpflib="clr-namespace:MyWPFLibrary;assembly=MyWPFLibrary"` definisce il namespace `mywpflib` e stabilisce che esso deve usare la Assembly precedentemente costruita.

8. Aggiungete nei TAG del Grid un nome già esistente.


10. Costruite questa soluzione:

    Ciò causa una segnalazione di errore:

    ![Immagini degli errori](image)

    11. per eliminare questo errore, cancellate semplicemente il file code-behind ed eseguite un rebuild.

Il prossimo passo consiste nell’aggiunta del file XAML in zenon.

**PASSI IN ZENON**

1. Apriete l’Editor di zenon

2. Passate al nodo File -> Grafiche
3. Selezionate nel menù contestuale **Aggiungi file...**

4. Scegliete il file XAML dal luogo di salvataggio e inseritelo.

5. Aggiungete le DLL con la funzionalità per il file XAML. A tale scopo:
   
   a) selezionate nel menù contestuale del nodo File -> Altri **Aggiungi file...**

   b) selezionate la **WPFToolkit.dll** e la DLL del primo progetto

6. impostate un´immagine.
7. Aggiungete un elemento WPF e selezionate il file XAML. A questo punto dovreste vedere questo segue:

![Image of WPF editor with XAML file]

Nota: se il file XAML viene cancellato o ne deve essere fatto un update all’interno di zenon, potrebbe essere che le DLL siano ancora aperte e non possano essere cancellate dalla cartella file. Per risolvere questo problema si deve riavviare l’Editor. Probabilmente sarà anche sufficiente disattivare il progetto per poi riattivarlo.
### 6.3.7 Trattamento errori

**INSERIMENTI NEI FILE LOG.**

<table>
<thead>
<tr>
<th>Inserimento</th>
<th>Level</th>
<th>Descrizione</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xaml file found in %s with different name, using default!</td>
<td>Avviso</td>
<td>Il nome del file collection e quello del file XAML in esso contenuto non corrispondono. Per evitare dei conflitti interni, nella cartella viene usato il file con il nome del file collection e la desinenza .xaml.</td>
</tr>
<tr>
<td>no preview image found in %s</td>
<td>Avviso</td>
<td>Il file collection non contiene una grafica anteprima valida <a href="#">preview.png</a> oppure [Nome del file XAML].png. Perciò non può essere visualizzata nessuna anteprima.</td>
</tr>
<tr>
<td>Xaml file in %s not found or not unique!</td>
<td>Errore</td>
<td>Il file collection non contiene nessun file XAML, oppure più file con la desinenza .xaml; non può essere usato.</td>
</tr>
<tr>
<td>Could not remove old assembly %s</td>
<td>Avviso</td>
<td>Nella cartella di lavoro si trova un Assembly che deve essere sostituito con una nuova versione, ma non può essere cancellato.</td>
</tr>
<tr>
<td>Could not copy new assembly %s</td>
<td>Errore</td>
<td>Per un Assembly nella cartella di lavoro è disponibile una nuova versione, che però non può essere copiata nella cartella stessa. Possibile causa: per es. il vecchio Assembly è ancora caricato. La vecchia versione continua ad essere usata, la nuova non può essere utilizzata.</td>
</tr>
<tr>
<td>file exception in %s</td>
<td>Errore</td>
<td>Si è verificato un errore file nel momento di accedere al file collection.</td>
</tr>
<tr>
<td>Generic exception in %s</td>
<td>Errore</td>
<td>Si è verificato un errore generale nel momento di accedere al file collection.</td>
</tr>
</tbody>
</table>