zenon
by COPA-DATA

Manuale di zenon

Controlli

v.8.20

© 2020 Ing. Punzenberger COPA-DATA Srl

Tutti i diritti riservati.

La distribuzione e la copia di questo documento - indifferentemente dal metodo usato - è consentita solo con il permesso scritto della società COPA-DATA. I dati tecnici servono solo per la descrizione del prodotto e non sono caratteristiche garantite in senso legale. Ci riserviamo il diritto di apportare modifiche - anche per quanto riguarda gli aspetti tecnici.

Indice

1 Benvenuti nell'help di COPA-DATA 5

2 Controlli 5

3 Generale 6

3.1 Accesso all´API di zenon 6

3.2 Metodi 8

3.2.1 CanUseVariables 8

3.2.2 MaxVariables 9

3.2.3 VariableTypes 9

3.2.4 zenonExit 9

3.2.5 zenonExitEd 10

3.2.6 zenonInit 10

3.2.7 zenonInitEd 10

4 ActiveX 10

4.1 Sviluppare elementi Active X 11

4.1.1 Metodi 11

4.2 Esempio LatchedSwitch (C++) 13

4.2.1 Interfaccia 14

4.2.2 Control 14

4.2.3 Metodi 17

4.2.4 Gestione e visualizzazione 20

4.2.5 zenon Interface 22

4.3 Esempio CD_SliderCtrl (C++) 22

4.3.1 Interfaccia 23

4.3.2 Control 23

4.3.3 Metodi 26

4.3.4 Gestione e visualizzazione 28

4.3.5 zenon Interface 30

4.4 Esempio .NET Control come ActiveX (C#) 30

4.4.1 Creare Windows Form Control 30

4.4.2 Convertire .NET User Control in Dual Control 33

4.4.3 Lavorare nell´Editor con ActiveX via VBA 37

4.4.4 Collegare variabili di zenon al .NET User Control 38

5 .NET User Controls 42

5.1 Differenze nell´uso di .NET Control in Control Container e ActiveX 43
5.2 Esempio di .NET Control Container 43
5.2.1 Generale 43
5.2.2 Creare .Net User Control 46
5.2.3 Aggiungere CD_DotNetControlContainer e .NET User Control 54
5.2.4 Accesso al controllo utente tramite VSTA o VBA 59
5.3 Esempio .NET Control come ActiveX (C#) 62
5.3.1 Creare Windows Form Control 62
5.3.2 Convertire .NET User Control in Dual Control 65
5.3.3 Lavorare nell´Editor con ActiveX via VBA 69
5.3.4 Collegare variabili di zenon al .NET User Control 70

6 WPF 74

1 Benvenuti nell'help di COPA-DATA

ZENON VIDEO-TUTORIAL

Esempi pratici di progettazione con zenon si trovano nel nostro canale YouTube (https://www.copadata.com/tutorial_menu). I tutorial sono raggruppati per tema e forniscono una panoramica di come si lavora con i diversi moduli di zenon. Tutti i tutorial sono disponibili in inglese.

GUIDA GENERALE

Se mancano informazioni in questo capitolo dell'help o se avete richieste di aggiunte, contattate documentation@copadata.com via E-Mail.

SUPPORTO TECNICO ALLA PROGETTAZIONE

Per domande relative a progetti concreti, si prega di contattare il Customer Service via E-Mail all'indirizzo support@copadata.com.

LICENZE E MODULI

Se avete bisogno di moduli aggiuntivi o licenze, il nostro staff di sales@copadata.com sarà lieto di assistervi.

2 Controlli

In zenon possono essere implementati i propri controlli. Per questo sono disponibili:

- .NET User Controls (A pagina: 42) (sul tema implementazione in zenon, vedi anche .NET Controls nel manuale Immagini.)
- ActiveX (A pagina: 10) (sul tema implementazione in zenon vedi anche ActiveX nel manuale Immagini.)
- WPF

> **Info**
>
> Le informazioni sull'utilizzo delle interfacce di programmazione zenon (PCE, VBA, VSTA) si trovano nel manuale Interfacce di programmazione.

> **Attenzione**
>
> Tenere presente che errori in applicazioni come ActiveX, PCE, VBA, VSTA, WPF e applicazioni esterne che accedono a zenon via le API, possono condizionare anche la stabilità del Runtime.

3 Generale

Controlli per zenon possono essere realizzati tramite ActiveX, .NET e WPF. Si può accedere all´API di zenon tramite VBA/VSTA.

3.1 Accesso all´API di zenon

In zenon, un ActiveX Control può essere esteso integrandolo con funzioni speciali per accedere all´API di zenon.

ACCEDERE ALL´API DI ZENON

- Nelle **Projekt References**, selezionare tramite **Add References...** la **libreria di oggetti zenon Runtime**
- Inserire le funzioni estese nel codice di classe del controllo.

FUNZIONI ZENON ACTIVEX ESTESE

```
// Viene chiamata quando il controllo è inizializzato in zenon Runtime.
public bool zenonInit(zenon.Element dispElement)…
// Viene chiamata al momento della distruzione del Control in zenon Runtime.
public bool zenonExit()…
// Supporta i collegamenti variabili del controllo.
public short CanUseVariables()…
// Tipi di dati supportati da Com Control.
public short VariableTypes()…
```

```
// Numero massimo di variabili che possono essere collegate al controllo.
```

public short MaxVariables()...

ESEMPIO

L'oggetto COM di una variabile zenon viene memorizzato temporaneamente in un Member, in modo che sia possibile accedervi successivamente nel Paint Event del controllo.

```
zenon.Variabile 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.ReleaseComObject(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 dati sono supportati
}

public short MaxVariables()
{
return 1; //  Al controllo deve essere collegata al massimo una variabile.
}

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

I controlli ActiveX e .NET che utilizzano variabili zenon richiedono determinati metodi.

3.2.1 CanUseVariables

Prototipo: `short CanUseVariables();`

Questo metodo restituisce *1* o *0*

Valore	Descrizione
1:	Il controllo può utilizzare variabili zenon. Possono essere specificate per l'elemento dinamico (tramite il pulsante **Variabile**) solo variabili zenon dei tipi indicati con il metodo `VariableTypes` (A pagina: 9) e nel numero indicato con il metodo `MaxVariables` (A pagina: 9).
0:	Il controllo non può utilizzare variabili zenon, o non ha questo metodo. Possono essere specificate variabili di tutti i tipi, senza limitazione di numero. Tuttavia, queste possono essere utilizzate a Runtime solo tramite VBA.

3.2.2 MaxVariables

Prototipo: `short MaxVariables();`

Qui si definisce il numero di variabili che possono essere selezionate dall'elenco delle variabili.

Se il risultato è *1*, la selezione multipla viene disattivata nella lista delle variabili. Viene visualizzato un avviso se sono state comunque selezionate più variabili.

3.2.3 VariableTypes

Prototipo: `short VariableTypes();`

Il valore restituito da questo metodo viene utilizzato come maschera per i tipi di variabili utilizzabili nella lista delle variabili. Il valore è un'operazione **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
DWORT	0x0008	Posizione 3
FLOAT	0x0010	Posizione 4
DFLOAT	0x0020	Posizione 5
STRING	0x0040	Posizione 6
EIN_AUSGABE	0x8000	Posizione 15

3.2.4 zenonExit

Prototipo: `boolean zenonExit();`

Questo metodo è richiamato dal Runtime di zenon alla chiusura del controllo ActiveX.

Tutti i Dispatch Pointer per le variabili dovrebbero essere rilasciati.

3.2.5 zenonExitEd

Corrisponde a zenOnExit (A pagina: 9) e viene eseguito alla chiusura dell'ActiveX nell'Editor.

In questo modo l'Editor può reagire alle modifiche nell´ActiveX, ad esempio ai cambiamenti di valore.

Info: attualmente disponibile solo per ActiveX.

3.2.6 zenonInit

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

Questo metodo viene utilizzato a Runtime per passare al controllo ActiveX un puntatore per la Dispatch Interface dell'elemento dinamico. Questo puntatore può quindi essere utilizzato per indirizzare le variabili zenon collegate all´elemento dinamico.

Nella configurazione dell'elemento ActiveX si definisce l'ordine delle variabili trasferite mediante i pulsanti **Giù** o **Su**.
La finestra di dialogo **Introduzione elemento** si apre quando si fa doppio clic sull'elemento ActiveX, o si seleziona la proprietà **ActiveX Einstellungen** nel nodo **Darstellung** delle proprietà dell´elemento.

3.2.7 zenonInitEd

Corrisponde a zenonInit (A pagina: 10) e viene eseguito all'apertura dell'ActiveX (doppio clic sull'ActiveX) nell'Editor.

Info: attualmente disponibile solo per ActiveX.

4 ActiveX

La funzionalità del Runtime e dell´Editor di zenon può essere estesa con ActiveX .

Argomenti trattati in questo manuale:

- Sviluppare elementi Active X (A pagina: 11)
- Esempio LatchedSwitch (C++) (A pagina: 13)
- Esempio CD_SliderCtrl (C++) (A pagina: 22)
- Esempio .NET Control come ActiveX (C#) (A pagina: 30)

Le informazioni sull'elemento dinamico ActiveX si trovano nel capitolo ActiveX del manuale Immagini.

ACTIVEX PER WINDOWS CE

Se un ActiveX Control deve essere eseguito sotto Windows CE, il modello **apartment** deve essere impostato su *Threading*. Se è impostato su *Free*, il controllo non verrà eseguito nel Runtime di zenon.

4.1 Sviluppare elementi Active X

L´elemento dinamico ActiveX di zenon può trasferire variabili all´ActiveX Control senza che sia necessario passare per VBA per gestire il Control.

Il controllo può definire autonomamente quante variabili zenon può utilizzare e di che tipo possono essere. Le proprietà del Control possono essere definite utilizzando l'elemento dinamico.

A tal fine, l'interfaccia (Dispatch-Interface) del Control deve supportare una serie di metodi (A pagina: 11) specifici.

4.1.1 Metodi

Ogni controllo ActiveX che può utilizzare variabili zenon, deve contenere i seguenti metodi.

- CanUseVariables (A pagina: 8)
- MaxVariables (A pagina: 9)
- VariableTypes (A pagina: 9)
- zenonExit (A pagina: 9)
- zenonExitEd (A pagina: 10)
- zenonInit (A pagina: 10)
- zenonInitEd (A pagina: 10)

Non importa quale Dispatch ID hanno i metodi nell´interfaccia. Quando si richiamano i metodi, in zenon viene acquisito l´ID corretto dall´interfaccia.

4.1.1.1 CanUseVariables

Prototipo: `short CanUseVariables();`

Questo metodo restituisce *1* o *0*

Valore	Descrizione
1:	Il controllo può utilizzare variabili zenon. Possono essere specificate per l'elemento dinamico (tramite il pulsante **Variabile**) solo variabili zenon dei tipi indicati con il metodo `VariableTypes` (A pagina: 9) e nel numero

Valore	Descrizione
	indicato con il metodo `MaxVariables` (A pagina: 9).
0:	Il controllo non può utilizzare variabili zenon, o non ha questo metodo. Possono essere specificate variabili di tutti i tipi, senza limitazione di numero. Tuttavia, queste possono essere utilizzate a Runtime solo tramite VBA.

4.1.1.2 MaxVariables

Prototipo: `short MaxVariables();`

Qui si definisce il numero di variabili che possono essere selezionate dall'elenco delle variabili.

Se il risultato è *1*, la selezione multipla viene disattivata nella lista delle variabili. Viene visualizzato un avviso se sono state comunque selezionate più variabili.

4.1.1.3 VariableTypes

Prototipo: `short VariableTypes();`

Il valore restituito da questo metodo viene utilizzato come maschera per i tipi di variabili utilizzabili nella lista delle variabili. Il valore è un'operazione **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
DWORT	0x0008	Posizione 3
FLOAT	0x0010	Posizione 4
DFLOAT	0x0020	Posizione 5
STRING	0x0040	Posizione 6
EIN_AUSGABE	0x8000	Posizione 15

4.1.1.4 zenonExit

Prototipo: `boolean zenonExit();`

Questo metodo è richiamato dal Runtime di zenon alla chiusura del controllo ActiveX.

Tutti i Dispatch Pointer per le variabili dovrebbero essere rilasciati.

4.1.1.5 zenonExitEd

Corrisponde a zenOnExit (A pagina: 9) e viene eseguito alla chiusura dell'ActiveX nell'Editor.

In questo modo l'Editor può reagire alle modifiche nell´ActiveX, ad esempio ai cambiamenti di valore.

Info: attualmente disponibile solo per ActiveX.

4.1.1.6 zenonInit

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

Questo metodo viene utilizzato a Runtime per passare al controllo ActiveX un puntatore per la Dispatch Interface dell'elemento dinamico. Questo puntatore può quindi essere utilizzato per indirizzare le variabili zenon collegate all´elemento dinamico.

Nella configurazione dell'elemento ActiveX si definisce l'ordine delle variabili trasferite mediante i pulsanti **Giù** o **Su**.
La finestra di dialogo **Introduzione elemento** si apre quando si fa doppio clic sull'elemento ActiveX, o si seleziona la proprietà **ActiveX Einstellungen** nel nodo **Darstellung** delle proprietà dell´elemento.

4.1.1.7 zenonInitEd

Corrisponde a zenonInit (A pagina: 10) e viene eseguito all'apertura dell'ActiveX (doppio clic sull'ActiveX) nell'Editor.

Info: attualmente disponibile solo per ActiveX.

4.2 Esempio LatchedSwitch (C++)

L'esempio seguente descrive un controllo ActiveX che implementa 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 il controllo ActiveX solo se la variabile di blocco ha il valore 0.

Lo stato dell´elemento viene visualizzato da quattro bitmap, che possono essere selezionate nell´Editor di zenon tramite la finestra di dialogo delle proprietà del controllo.

4.2.1 Interfaccia

Il Control LatchedSwitch ha la seguente Dispatch Interface:

```
[ 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;
        [id(5)] IPictureDisp* LatchedOff;
        //}}AFX_ODL_PROP


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();

};
```

Le Properties da **SwitchOn** a **LatchedOff** contengono le bitmap per la visualizzazione dei quattro stati del controllo. Le stesse bitmap sono memorizzate in oggetti della classe CPictureHolder. La proprietà **SollwertDirekt** consente di specificare se l'immissione dei valori impostabili deve essere possibile tramite una finestra di dialogo, o direttamente cliccando sul controllo.

4.2.2 Control

La classe **CLatchedSwitchCtrl** viene utilizzata per implementare il controllo. Questa classe ha come Member gli oggetti **CPictureHolder** per contenere le bitmap. Inoltre, vengono trasportati 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,
dispidLatchedOff = 5L,
dispidCanUseVariables = 6L,
dispidVariableTypes = 7L,
dispidMaxVariables = 8L,
dispidZenOnInit = 9L,
dispidZenOnExit = 10L,
//}}AFX_DISP_ID
};
};
```

4.2.3 Metodi

Vengono usati i seguenti metodi:

- CanUseVariables (A pagina: 17)
- VariableTypes (A pagina: 18)
- MaxVariables (A pagina: 18)
- zenonInit (A pagina: 18)
- zenonExit (A pagina: 19)

4.2.3.1 CanUseVariables

Questo metodo restituisce 1, quindi si possono usare le variabili zenon.

```
short CLatchedSwitchCtrl::CanUseVariables()
{

return 1;
}
```

4.2.3.2 VariableTypes

Il Control può funzionare solo con variabili bit, quindi viene restituito *0x0004*.

```
short CLatchedSwitchCtrl::VariableTypes()
{

return 0x0004; // Nur Bit-Variablen
}
```

4.2.3.3 MaxVariables

Possono essere utilizzate due variabili, quindi viene restituito il valore *2*.

```
short CLatchedSwitchCtrl::MaxVariables()
{

return 2; // 2 Variablen
}
```

4.2.3.4 zenonInit

In questo metodo, i *Dispatchdriver* delle variabili sono recuperati tramite il *Dispatchpointer* dell'elemento dinamico. Questi *Pointer* sono utilizzati per ottenere e impostare i valori delle variabili quando si clicca e si disegna il controllo.

```
BOOL CLatchedSwitchCtrl::zenonInit(LPDISPATCH dispElement)

{

m_dElement = IElement(dispElement);

Element.m_lpDispatch->AddRef();

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;

}
```

Info

Element.m_lpDispatch->AddRef();

Gli oggetti inutilizzati non vengono automaticamente cancellati dalla memoria. Si ottiene questo risultato tramite la programmazione. Il programmatore determina se un oggetto può essere rimosso sulla base di un contatore di riferimento.

COM utilizza i metodi *IUnknow*, *AddRef* e *Release* per gestire il numero di riferimenti di interfacce su un oggetto.

Le regole generali per la chiamata di questi metodi sono:

- Ogni volta che il client riceve un Interface Pointer, *AddRef* deve essere richiamato sull'interfaccia.
- Ogni volta che il client smette di usare l´Interface Pointer, un *Release* deve essere richiamato.

In una semplice implementazione, una variabile contatore nell´oggetto viene incrementata ad ogni chiamata del metodo *AddRef* . Ogni chiamata di un *Release* riduce il contatore dell´oggetto. Quando questo contatore torna a NULL, l'Interface può essere rimossa dalla memoria.

È inoltre possibile implementare un contatore di riferimento in modo che ogni riferimento all'oggetto (e non a una singola Interface) venga conteggiato.

In questo caso, ogni *AddRef* e ogni *Release* chiamerà un sostituto di un'implementazione centrale sull'oggetto. Un *Release*, di conseguenza, rilascia l'intero oggetto quando il contatore di riferimento raggiunge lo zero.

4.2.3.5 zenonExit

Questo metodo rilascia nuovamente i Dispatchdriver.

```
BOOL CLatchedSwitchCtrl::zenonExit()
{

m_dElement.ReleaseDispatch();
m_dSwitchVar.ReleaseDispatch();
m_dLatchVar.ReleaseDispatch();
```

```
return TRUE;
}
```

4.2.4 Gestione e visualizzazione

4.2.4.1 Imposta valore

Un valore di setpoint viene impostato sul controllo cliccando con il tasto sinistro del mouse.

Se **m_iSollwertDirekt** è *0*, viene visualizzata una finestra di dialogo per la selezione del valore impostabile, altrimenti il valore corrente della variabile di commutazione viene invertito.

Se la variabile di blocco ha il valore *1*, viene eseguito solo un *MessageBeep* e non è possibile impostare alcun valore di setpoint tramite il controllo

```
void CLatchedSwitchCtrl::OnLButtonDown(UINT nFlags, CPoint point)
{

CRect rcBounds;
GetWindowRect(&rcBounds);

COleVariant coleValue((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);
}
```

4.2.4.2 Disegno

Quando si disegna il controllo, i valori delle variabili vengono recuperati tramite i Dispatchdriver di queste ultime, e uno dei quattro grafici predefiniti viene disegnato di conseguenza. Se il valore di una variabile cambia, il controllo viene aggiornato utilizzando la 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 zenon Interface

Per poter utilizzare la Dispatch Interface di zenon per settare valori impostabili, devono essere create classi derivate da COleDispatchDriver per le variabili e l'elemento. Il modo più semplice per creare queste classi è quello di utilizzare il Class Wizard dell'ambiente di sviluppo (pulsante **Add Class**, selezionare **From a type library**, scegliere *zenrt32.tlb*).

Nel nostro controllo queste sono le classi **IElement** e **IVariable**. Sono definite in *zenrt32.h* e *zenrt32.cpp*.

4.3 Esempio CD_SliderCtrl (C++)

L'esempio seguente descrive un controllo ActiveX uguale a Windows **SliderCtrl**. Questo componente può essere collegato ad una variabile di zenon. L'utente può utilizzare uno slider universale per modificare il setpoint di una variabile. Se la variabile è modificata tramite un altro elemento dinamico, lo slider viene aggiornato.

4.3.1 Interfaccia

Il controllo **CD_SliderCtrl** ha la seguente Dispatch Interface:

```
[ 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)

[id(4)] boolean zenonInit(IDispatch* pElementInterface);
[id(5)] boolean zenonExit();
[id(6)] short VariableTypes();
[id(7)] short CanUseVariables();
[id(8)] short MaxVariables();

[id(DISPID_ABOUTBOX)] void AboutBox();
};
```

4.3.2 Control

Il controllo è implementato utilizzando la classe CD_SliderCtrlCtrl. Questa classe ha come member un **CSliderCtrl** standard in Windows, con cui il controllo viene editato come sottoclasse. Le interfacce **IVariable** e **IElementcontengono interfacce**zenon che dovevano essere integrate. Sono derivate da **COleDispatchDriver**.

```
class CCD_SliderCtrlCtrl : public COleControl
{

DECLARE_DYNCREATE(CCD_SliderCtrlCtrl)
private: //*** Member-Variablen

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();
//*** Metodi di 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);
afx_msg short GetTickOrientation();
afx_msg void SetTickOrientation (short nNewValue);
afx_msg BOOL zenonInit(LPDISPATCH pElementInterface);
afx_msg BOOL zenonExit();
afx_msg short VariableTypes();
afx_msg short CanUseVariables();
afx_msg short MaxVariables();
//}}AFX_DISPATCH
DECLARE_DISPATCH_MAP()

afx_msg void AboutBox();

//{{AFX_EVENT(CCD_SliderCtrlCtrl)
//}}AFX_EVENT
DECLARE_EVENT_MAP()

public:

enum {
//{{AFX_DISP_ID(CCD_SliderCtrlCtrl)
dispidShowVertical = 1L,
dispidTicksOnBothSides = 2L,
dispidTickOrientation = 3L,
dispidZenOnInit = 4L,
dispidZenOnExit = 5L,
dispidVariableTypes = 6L,
```

```
dispidCanUseVariables = 7L,
dispidMaxVariables = 8L,
//}}AFX_DISP_ID
};
};
```

4.3.3 Metodi

Vengono usati i seguenti metodi:

- CanUseVariables (A pagina: 26)
- VariableTypes (A pagina: 26)
- MaxVariables (A pagina: 27)
- zenonInit (A pagina: 27)
- zenonExit (A pagina: 28)

4.3.3.1 CanUseVariables

Questo metodo restituisce *1*, quindi è possibile utilizzare le variabili zenon.

```
short CCD_SliderCtrlCtrl::CanUseVariables()
{

return 1;
}
```

4.3.3.2 VariableTypes

Il controllo può funzionare con le variabili word, byte, double word e float. Nella descrizione (A pagina: 9) generale di questo metodo è possibile trovare un elenco dei possibili tipi di dati.

```
short CCD_SliderCtrlCtrl::VariableTypes()
{

return 0x0001 | // Wort

0x0002 | // Byte
0x0008 | // D-Wort
0x0010 | // Float
```

```
0x0020; // D-Float
}
```

4.3.3.3 MaxVariables

A questo controllo può essere collegata una sola variabile.

```
short CCD_SliderCtrlCtrl::MaxVariables()
{

return 1; // 1 Variablen
}
```

4.3.3.4 zenonInit

Il parametro **dispElement** contiene l'interfaccia per l'elemento dinamico. Questo elemento viene utilizzato per determinare la variabile zenon collegata. Se questa è valida, è impostato l'intervallo dello **SliderCtrls**. Inoltre, vengono settate le impostazioni per la visualizzazione (numero di ticks, ...). Se non è collegata nessuna variabile, viene definito un intervallo di funzionamento da *0* a *0*. Così SliderCtrl non può essere modificato. La variabile **m_bInitialized** viene utilizzata per specificare che i valori impostabili possono ora essere settati.

```
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;
}

m_bInitialized = TRUE;
return TRUE;
}
```

4.3.3.5 zenonExit

In questo metodo vengono rilasciate nuovamente le interfacce zenon.

```
BOOL CCD_SliderCtrlCtrl::zenonExit()
{

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

4.3.4 Gestione e visualizzazione

4.3.4.1 Disegno

Lo SliderCtrl viene disegnato tramite **DoSuperclassPaint** (dato che si tratta di un controllo di tipo gesubclassed). Se lo slider viene spostato al momento del disegno, la variabile **m_bInitialized** assume il valore *FALSE*. Questo garantisce che il valore possa essere modificato. Normalmente, il valore della variabile viene determinato e visualizzato utilizzando il metodo **SetPos** di SliderCtrl.

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

//***  Aggiorna la vista
DoSuperclassPaint(pdc, rcBounds);
if (m_interfaceVariable && m_bInitialized) {
```

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

4.3.4.2 Imposta valore

Nel metodo **LButtonDown** la variabile **m_bInitialized** viene impostata su *FALSE*, e nell'evento **LButtonUp** viene riportata a *TRUE*. Questo garantisce che il valore possa essere modificato. Altrimenti verrebbe richiamata la routine **OnDraw** e visualizzato il valore originale.

```
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´impostazione valore viene causato da una modifica dello slider. Nei metodi **HScroll** o **VScroll** viene settato il valore impostabile (a seconda che lo slider sia orizzontale o verticale).

```
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 zenon Interface

Per poter utilizzare la Dispatch Interface di zenon per settare valori impostabili, devono essere create classi derivate da COleDispatchDriver per le variabili e l'elemento. Il modo più semplice per creare queste classi è quello di utilizzare il Class Wizard dell'ambiente di sviluppo (pulsante **Add Class**, selezionare **From a type library**, scegliere *zenrt32.tlb*).

Nel nostro controllo queste sono le classi **IElement** e **IVariable**. Sono definite in *zenrt32.h* e *zenrt32.cpp*.

4.4 Esempio .NET Control come ActiveX (C#)

L'esempio seguente descrive un controllo .NET che viene eseguito come ActiveX Control in zenon.

La creazione e l'integrazione avviene in quattro fasi:

1. Creare Windows Form Control (A pagina: 30)
2. Convertire .NET User Control in Dual Control (A pagina: 33)
3. Lavorare nell´Editor con ActiveX via VBA (A pagina: 37)
4. Collegare variabili di zenon al .NET User Control (A pagina: 38)

Attenzione

Quando si utilizzano gli oggetti COM di zenon nei controlli utente o nelle applicazioni esterne di creazione propria, questi devono essere rilasciati utilizzando il metodo **Marshal.ReleaseComObject**. Il rilascio tramite il metodo **Marshal.FinalReleaseComObject** non deve essere utilizzato, perché Add-In di zenon smettono di funzionare quando si usa questo metodo.

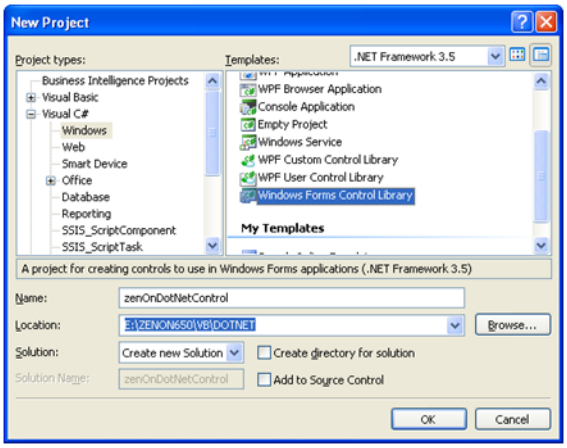
Info

Gli screenshot per questo tema sono disponibili solo in inglese.

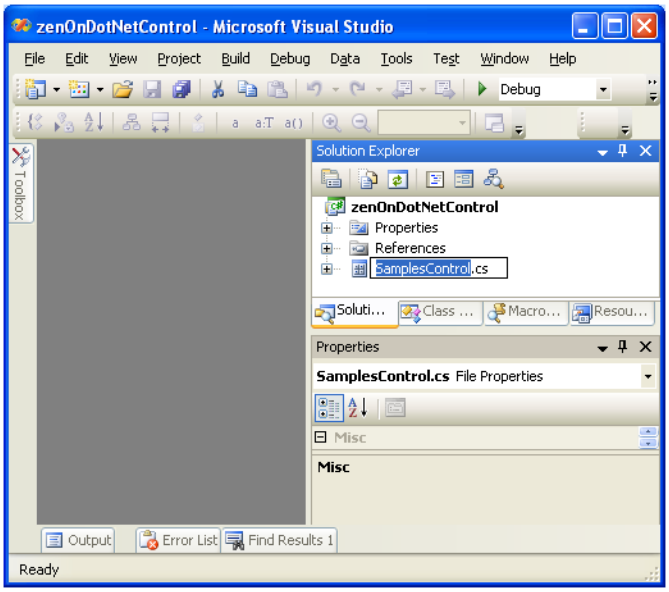
4.4.1 Creare Windows Form Control

Per creare un controllo Windows Form:

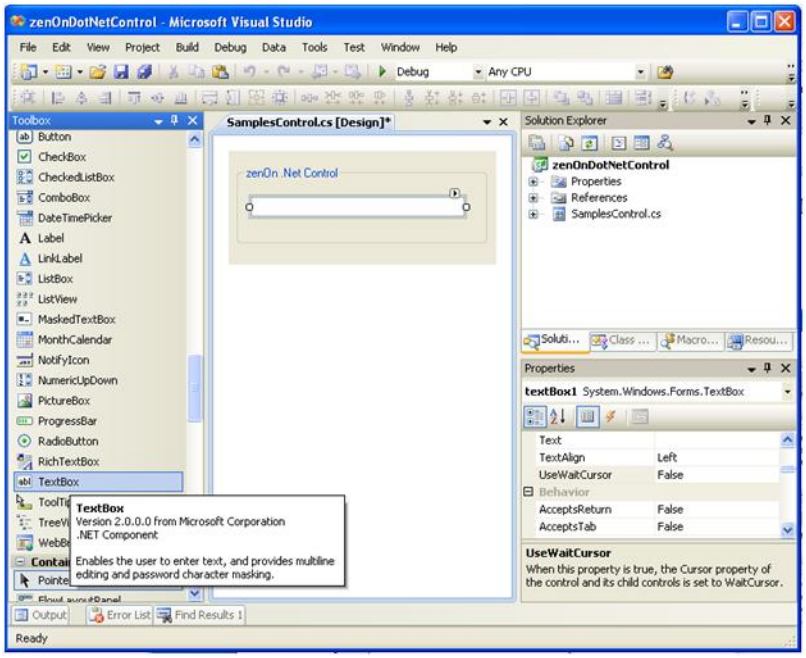
1. Avviare Visual Studio 2008 e e creare un nuovo progetto Windows **Form Control Library**:



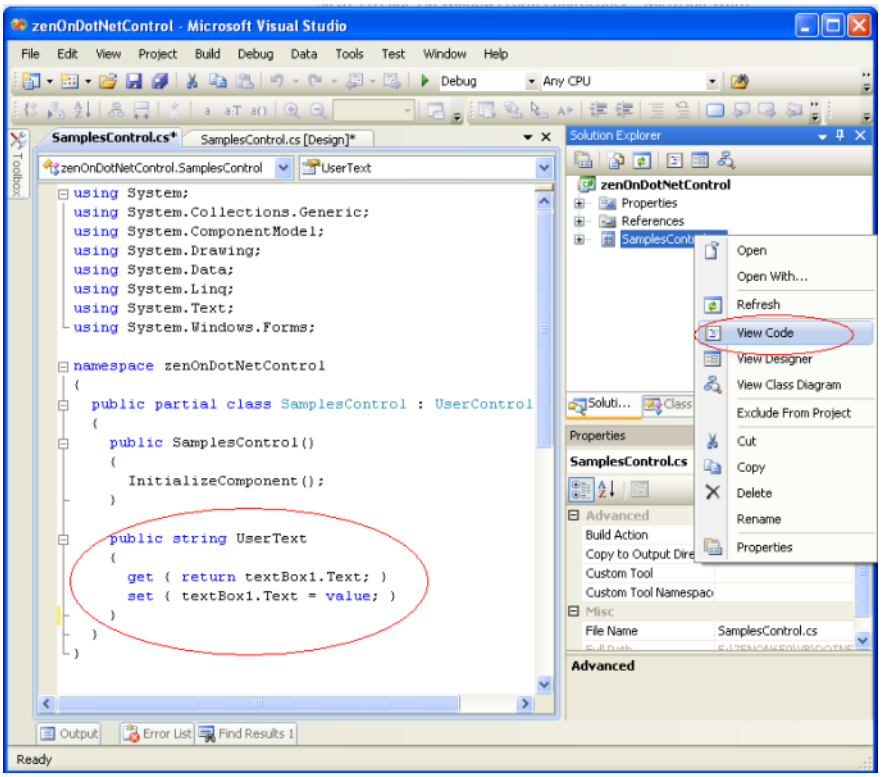
2. Rinominare il controllo predefinito assegnandogli il nome desiderato.
   In questo esempio: **SampesControl.cs**.



3. Aprire il Control Designer e inserire i controlli desiderati, nel nostro caso una casella di testo:



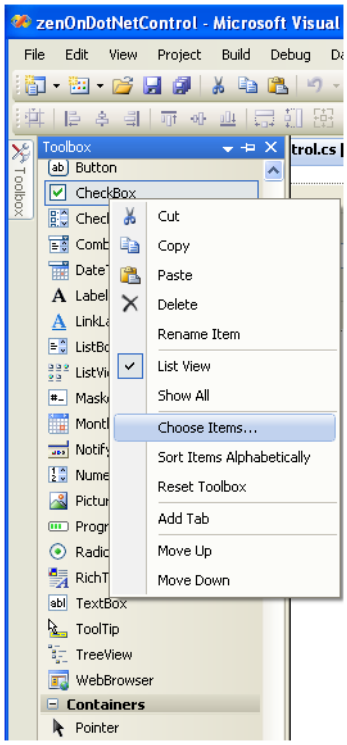
4. I controlli di solito hanno delle proprietà. Aprire Code Designer tramite **View Code** e aggiungere le proprietà disponibili esternamente.
   In questo esempio: Proprietà visibile esternamente "**UserText**" con accesso **get** e **set**, che contiene il testo della casella di testo:



5. Compilare il progetto.

Il controllo Windows Form può ora essere utilizzato in altri progetti Windows Forms.

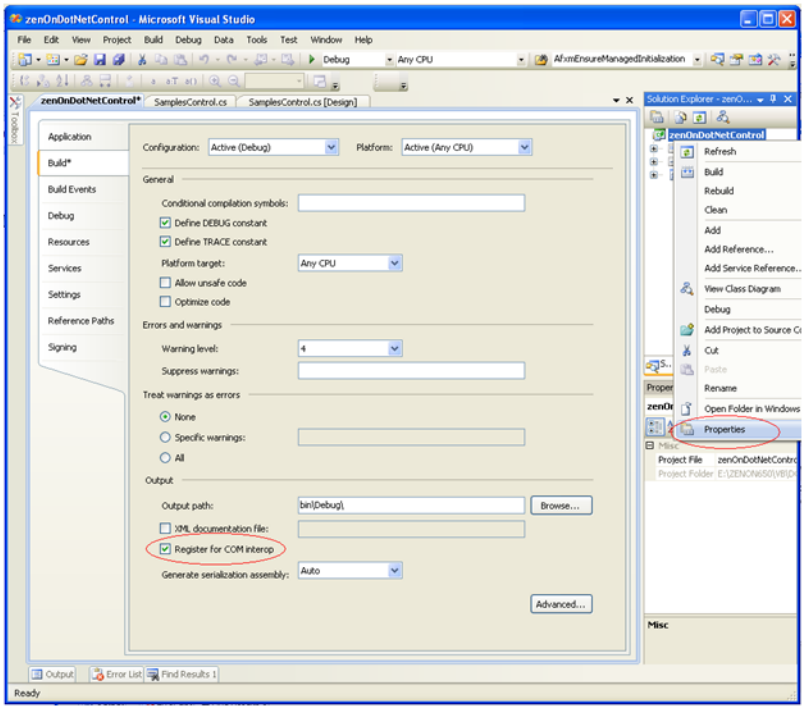
Importante: il controllo deve essere inserito manualmente nel Control Toolbox utilizzando **Choose Items**.



4.4.2 Convertire .NET User Control in Dual Control

Per convertire il .NET in un Dual Control, deve prima essere attivata l'interfaccia COM per l'ActiveX.

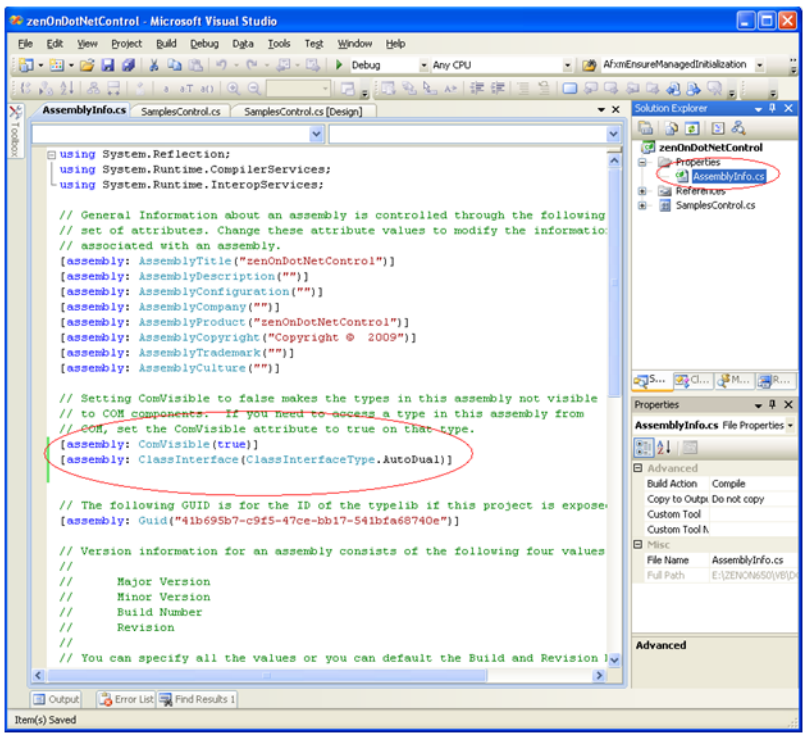
1. Aprire il progetto e attivare la proprietà **Register for COM interop** nelle impostazioni **Build**:



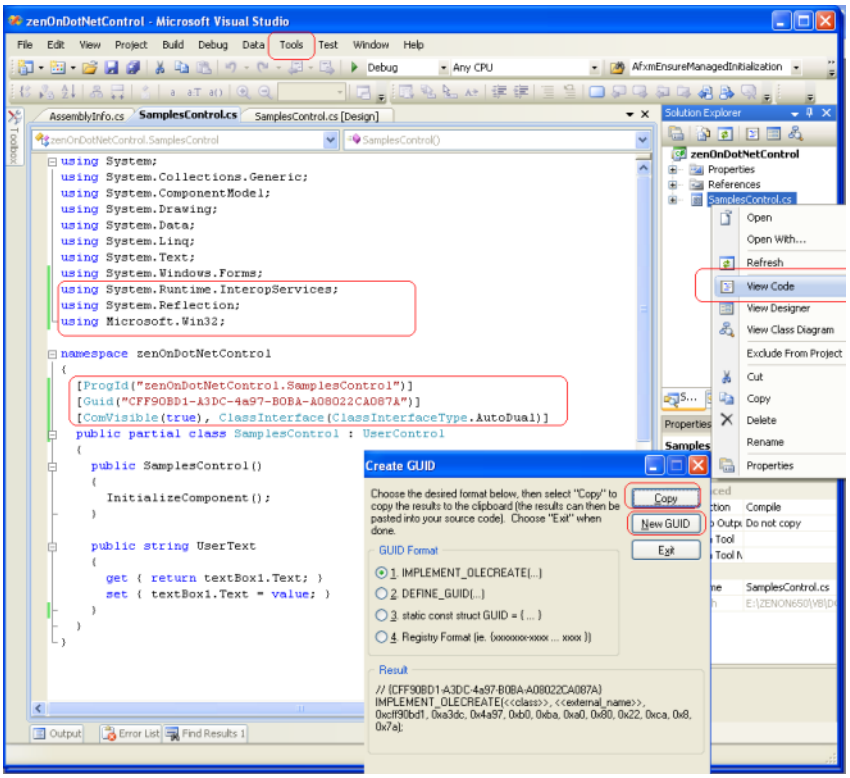
2. Aprire il file **AssemblyInfo.cs** e
   - Impostare l´attributo **ComVisible** su *true*
   - Aggiungere l´attributo **ClassInterface**

   [`assembly`: ComVisible(`true`)]

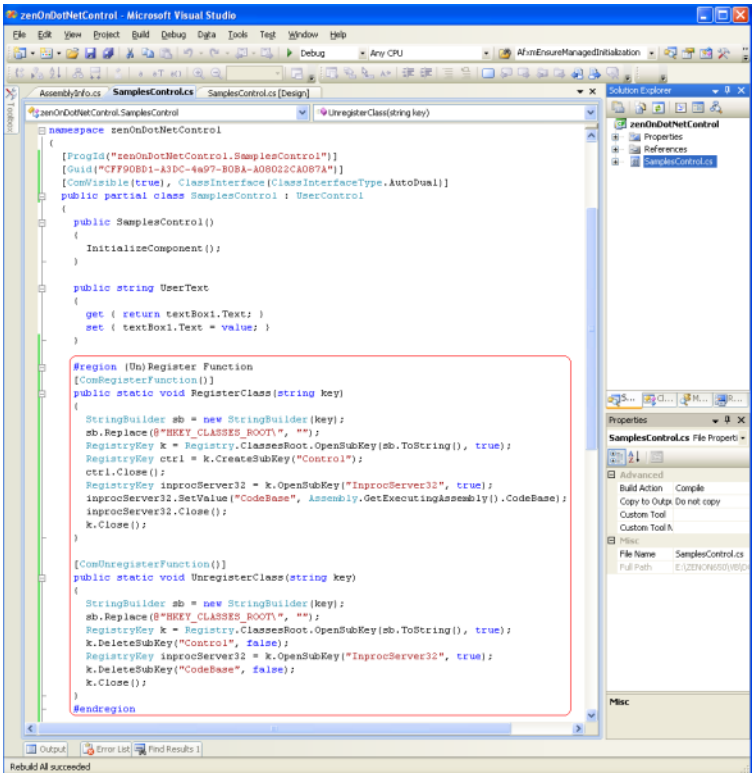
[assembly: ClassInterface(ClassInterfaceType.AutoDual)]



3. Aprire il Code Designer tramite **View Code** e aggiungere gli attributi ActiveX e gli inserimenti **using** necessari. Utilizzare il menù **Tools/Create GUID** per creare un nuovo GUID per l'attributo GUID:



4. Affinché il controllo possa essere selezionato anche come controllo di interfaccia ActiveX, alla classe di controllo devono essere aggiunte le seguenti funzioni:
   - RegisterClass
   - UnregisterClass



Il controllo può quindi essere registrato nella Registry.

5. Ricompilate il progetto.

   Il controllo Windows Form è ora abilitato ActiveX ed è stato registrato automaticamente durante il Rebuild . Un ulteriore file TypeLib **zenOnDotNetControl.tlb** è stato creato nella directory di output.

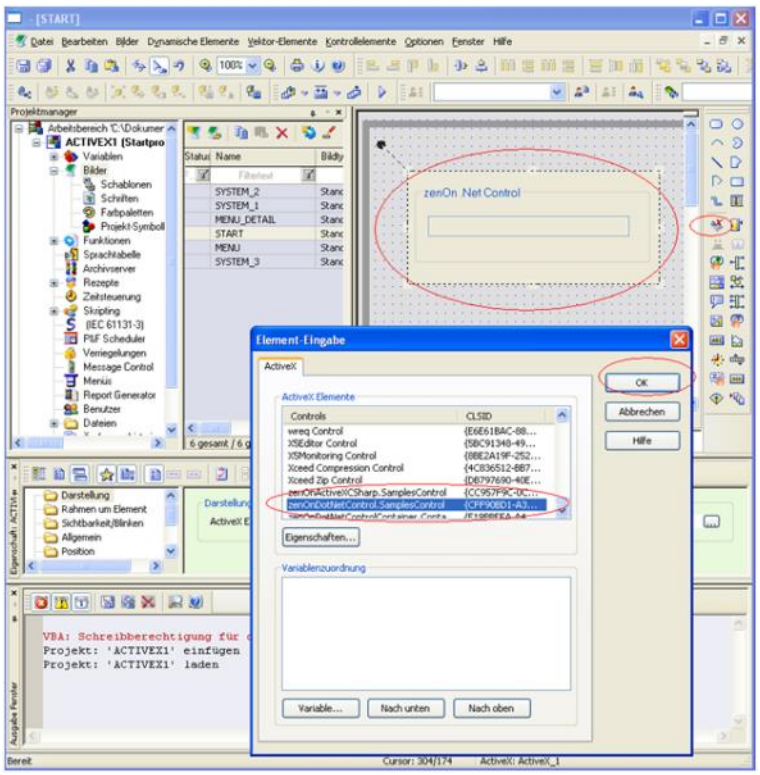
6. Per utilizzare il controllo su un altro computer:

   a) Copiare il file DLL e il file TLB sul computer di destinazione

   b) Registrare i file tramite la riga di comando con:
   [Posizione di memorizzazione .NET-Framework]\RegAsm.exe "MyControl.dll" /codebase

   Esempio a 64-Bit Sistema operativo Windows:
   C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe "MyControl.dll" /codebase
   Esempio a 32-Bit Sistema operativo Windows:
   C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe "MyControl.dll" /codebase

7. Inserire l'estensione del Windows Form control come ActiveX Control nell´Editor di zenon:



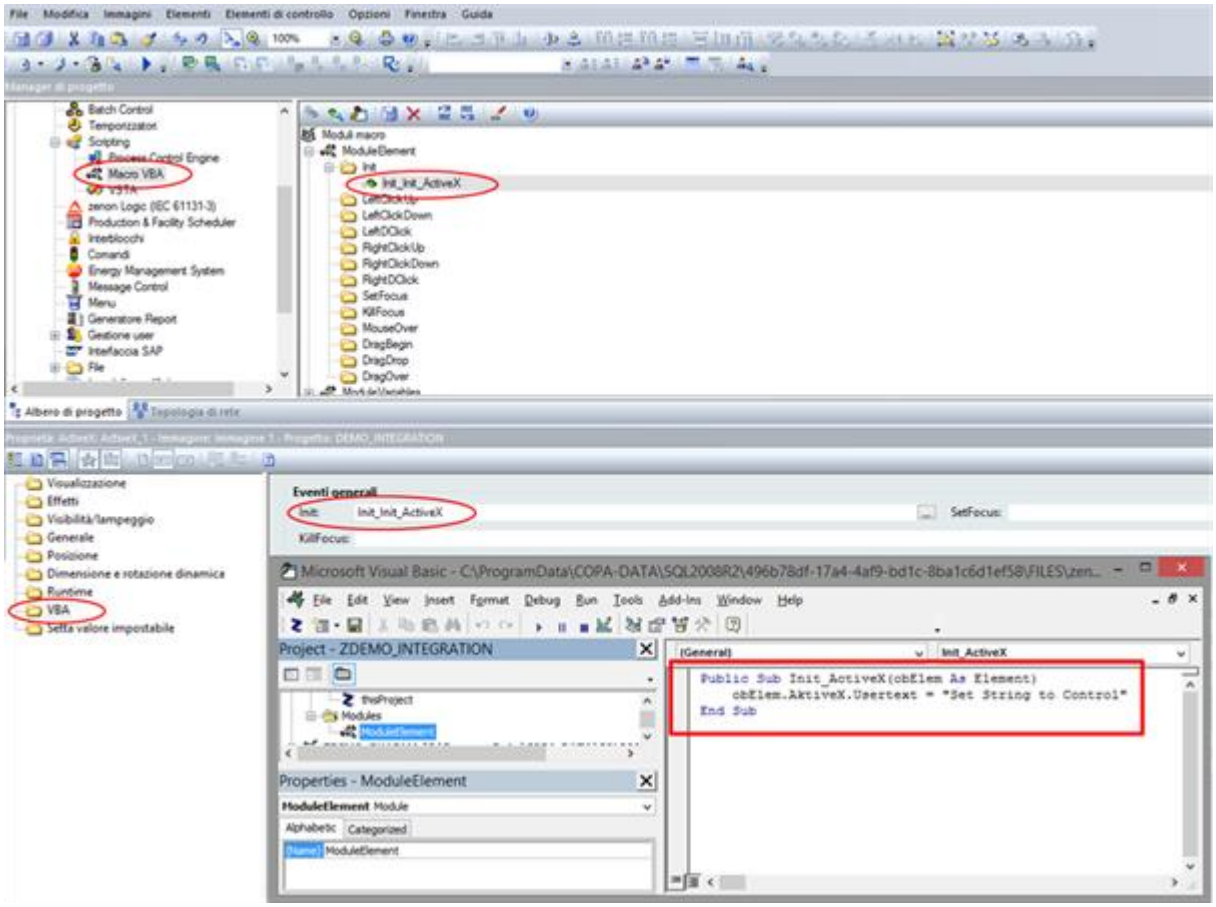
4.4.3 Lavorare nell´Editor con ActiveX via VBA

Per accedere alle proprietà del controllo nell´Editor di zenon:

1. Creare nell´Editor di zenon, nel nodo **Scripting/Macro-VBA**, una nuova macro **Init** con il nome **Init_ActiveX**.

   Con questa macro è poi possibile accedere a tutte le proprietà esterne tramite **obElem.AktiveX...**.

2. Assegnare questa macro al controllo ActiveX tramite le proprietà **VBA Makros/Init** dell´elemento ActiveX.



ESEMPIO MACRO INIT

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

4.4.4 Collegare variabili di zenon al .NET User Control

In zenon si ha la possibilità di estendere un controllo ActiveX integrandolo con funzioni speciali, e di accedere all API di zenon.

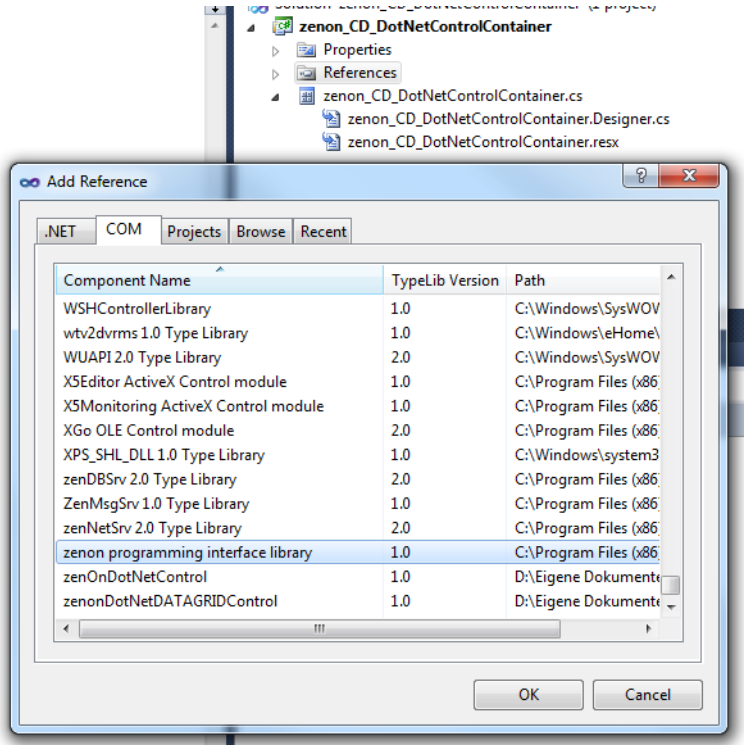
METODI NECESSARI

- public bool zenOnInit (A pagina: 40) (Chiamato durante l'inizializzazione del controllo nel Runtime di zenon.)
- public bool zenOnInitED (A pagina: 41) (viene utilizzato nell´Editor.)

- public bool zenOnExit() (A pagina: 41) (chiamato durante la distruzione del controllo nel Runtime di zenon.)
- public bool zenOnExitED() (A pagina: 41) (viene utilizzato nell´Editor)
- public short CanUseVariables() (A pagina: 41) (supporta il collegamento di variabili)
- public short VariableTypes() (A pagina: 41) (tipi di dati supportati dal controllo)
- public MaxVariables() (A pagina: 42) (numero massimo di variabili che possono essere collegate al controllo.)

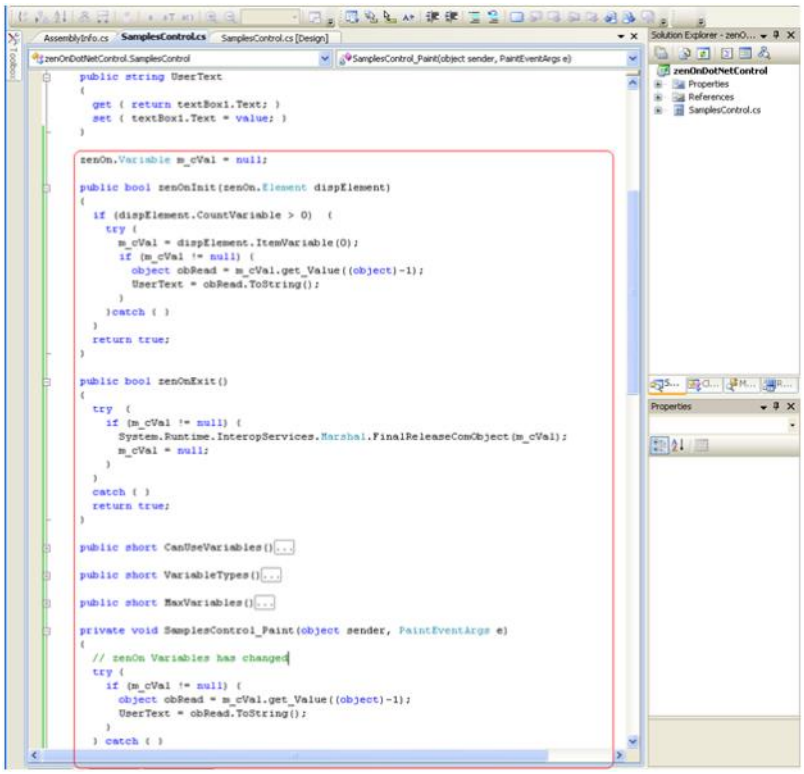
AGGIUNGI RIFERIMENTO

1. In Microsoft Visual Studio, selezionare alla voce **Add References** la zenon Runtime Object Library, per accedere all'API di zenon nel controllo.



2. Per accedere all'intera zenon API, includere le funzioni avanzate nel codice di classe del controllo.

Nel nostro esempio, l'oggetto COM di una variabile di zenon è memorizzato temporaneamente in un **Member**, in modo che sia possibile accedervi successivamente nell'evento **Paint** del controllo.



4.4.4.1 public bool zenOnInit(zenOn.Element dispElement)

Questo metodo viene utilizzato a Runtime per passare al controllo ActiveX un puntatore per la Dispatch Interface dell'elemento dinamico. Questo puntatore può quindi essere utilizzato per indirizzare le variabili zenon collegate all´elemento dinamico.

È possibile impostare l'ordine delle variabili trasferite nella finestra di dialogo "Introduzione elemento" utilizzando i pulsanti **Giù** o **Su**. Il dialogo "Introduzione elemento" si apre quando:

- Si fa doppio clic sull´elemento ActiveX
- Si seleziona **Proprietà**, o
- Si seleziona nella finestra delle proprietà la proprietà **ActiveX Einstellungen** del nodo **Darstellung**.

4.4.4.2 public bool zenOnInitED(zenOn.Element dispElement)

Corrisponde a public bool zenOnInit (A pagina: 40)e viene eseguito all'apertura dell'ActiveX nell'Editor (doppio clic su ActiveX).

4.4.4.3 public bool zenOnExit()

Questo metodo è richiamato dal Runtime di zenon alla chiusura del controllo ActiveX. Tutti i Dispatch Pointer per le variabili dovrebbero essere rilasciati.

4.4.4.4 public bool zenOnExitED()

Corrisponde a public bool zenOnExit() (A pagina: 41) e viene eseguito alla chiusura dell'ActiveX nell'Editor. Questo permette anche di reagire a cambiamenti (per es. cambiamenti di valore) nell'Editor.

4.4.4.5 public short CanUseVariables()

Questo metodo restituisce *1* se il controllo zenon può usare variabili, e *0* in caso contrario

- *1*: Solo variabili zenon dei tipi indicati con il metodo `VariableTypes` e nella quantità indicata con il metodo `MaxVariables`, possono essere specificati per l'elemento dinamico (usando il pulsante **Variabili**)
- *0*: Se `CanUseVariables` restituisce *0*, o se il controllo non ha questo metodo, possono essere specificate variabili di tutti i tipi e senza limitazioni di numero. Tuttavia, queste possono essere utilizzate a Runtime solo tramite VBA.

4.4.4.6 public short VariableTypes()

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

Parametro	Valore	Descrizione
WORD	*0x0001*	Corrisponde a posizione *0*
BYTE	*0x0002*	Corrisponde a posizione *1*
BIT	*0x0004*	Corrisponde a posizione *2*
DWORD	*0x0008*	Corrisponde a posizione *3*

Parametro	Valore	Descrizione
FLOAT	*0x0010*	Corrisponde a posizione *4*
DFLOAT	*0x0020*	Corrisponde a posizione *5*
STRING	*0x0040*	Corrisponde a posizione *6*
IN_OUTPUT	*0x8000*	Corrisponde a posizione *15*

4.4.4.7 public MaxVariables()

Qui si definisce il numero di variabili che possono essere selezionate dall'elenco delle variabili:

1: la selezione multipla è disattivata nella lista delle variabili. Viene visualizzato un avviso se sono state comunque selezionate più variabili.

5 .NET User Controls

La funzionalità del Runtime e dell´Editor di zenon può essere estesa con .NET Control

In questo manuale si leggerà:

- Differenze fra Control Container e ActiveX (A pagina: 43)
- Esempio di .NET Control Container (A pagina: 43)
- Esempio .NET Control come ActiveX (C#) (A pagina: 30)

Le informazioni sull'uso dei .NET Controls in ActiveX si trovano nel capitolo .NET Controls del manuale Immagini.

Attenzione

Quando si utilizzano gli oggetti COM di zenon nei controlli utente o nelle applicazioni esterne di creazione propria, questi devono essere rilasciati utilizzando il metodo **Marshal.ReleaseComObject**. Il rilascio tramite il metodo **Marshal.FinalReleaseComObject** non deve essere utilizzato, perché Add-In di zenon smettono di funzionare quando si usa questo metodo.

Nota: Durante l´installazione di zenon, viene installato .NET 4.6.2. .NET 3.5, richiesto per .Net Controls, non viene installato automaticamente e deve essere installato manualmente se necessario.

5.1 Differenze nell´uso di .NET Control in Control Container e ActiveX

Un .Net User Control può:

- Essere integrato direttamente nell'elemento ActiveX di zenon tramite CD_DotNetControlContainer Control.
- Può essere utilizzato come controllo ActiveX e integrato direttamente nell'elemento ActiveX di zenon.

Le differenze tra Container Control e ActiveX Control sono principalmente:

CD_DotNetControlContainer Control	ActiveX Control
Non deve essere registrato sul computer.	Deve essere registrato come ActiveX sul computer (regsvr32).
Quando si cambia il controllo, deve essere sostituita solo la DLL.	Se il controllo viene cambiato, il TLB deve essere nuovamente registrato.
L'accesso via VBA e VSTA è possibile solo tramite i metodi CD_DotNetControlContainer.	Accesso tramite VBA e VSTA molto facile.

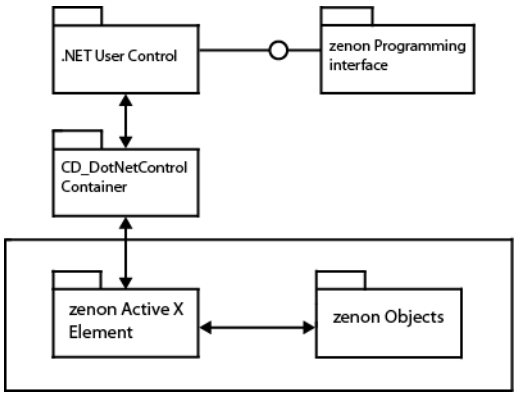
5.2 Esempio di .NET Control Container

Questo tutorial descrive come creare un semplice .NET User Control (linguaggio di programmazione **C#**) in Visual Studio 2010, e come integrarlo come come ActiveX in un elemento ActiveX di zenon utilizzando il zenon **CD_DotNetControlContainer** .

5.2.1 Generale

Il CD_DotNetControlContainer funge da Wrapper fra un User Control e l´elemento ActiveX di zenon. Tutti i metodi usati nell'esempio seguente, così come tutti i Public Methods e Properties, vengono passati dall´User Control all´ActiveX tramite il CD_DotNetControlContainer, e possono essere usati da zenon, anche in VBA e VSTA.

Se nell´User Control si referenzia alla zenon Programming Interface, è possibile accedere direttamente agli oggetti di zenon.



Nel seguente esempio si farà quanto segue:

- Creare un .NET User Control (A pagina: 46)
- Aggiungere CD_DotNetControlContainer e .NET User Control (A pagina: 54)
- Consentire l´accesso all´User Control tramite VSTA (VBA) (A pagina: 59)

PERCORSO PER DLL IN EDITOR E RUNTIME

Il percorso di **.Net DLL** selezionato nell'Editor, viene utilizzato anche a Runtime. È impostato in modo assoluto e non deve essere modificato.

Assicurarsi che su tutti i computer della rete zenon venga utilizzato lo stesso percorso per Editor e Runtime.
Suggerimento: selezionare un percorso assoluto, ad esempio: *C:\Controls*. Inserire il percorso nel **Remote-Transport** e nel **.NET Control Container**. Sincronizzare questo percorso su tutti i computer utilizzando il **Remote-Transport**.

5.2.1.1 public bool zenOnInit(zenOn.Element dispElement)

Questo metodo viene utilizzato a Runtime per passare al controllo ActiveX un puntatore per la Dispatch Interface dell'elemento dinamico. Questo puntatore può quindi essere utilizzato per indirizzare le variabili zenon collegate all´elemento dinamico.

È possibile impostare l'ordine delle variabili trasferite nella finestra di dialogo "Introduzione elemento" utilizzando i pulsanti **Giù** o **Su**. Il dialogo "Introduzione elemento" si apre quando:

- Si fa doppio clic sull´elemento ActiveX
- Si seleziona **Proprietà**, o
- Si seleziona nella finestra delle proprietà la proprietà **ActiveX Einstellungen** del nodo **Darstellung**.

5.2.1.2 public bool zenOnExit()

Questo metodo è richiamato dal Runtime di zenon alla chiusura del controllo ActiveX. Tutti i Dispatch Pointer per le variabili dovrebbero essere rilasciati.

5.2.1.3 public short CanUseVariables()

Questo metodo restituisce *1* se il controllo zenon può usare variabili, e *0* in caso contrario

- *1*: Solo variabili zenon dei tipi indicati con il metodo `VariableTypes` e nella quantità indicata con il metodo `MaxVariables`, possono essere specificati per l'elemento dinamico (usando il pulsante **Variabili**)
- *0*: Se `CanUseVariables` restituisce *0*, o se il controllo non ha questo metodo, possono essere specificate variabili di tutti i tipi e senza limitazioni di numero. Tuttavia, queste possono essere utilizzate a Runtime solo tramite VBA.

5.2.1.4 public short VariableTypes()

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

Parametro	Valore	Descrizione
WORD	*0x0001*	Corrisponde a posizione *0*
BYTE	*0x0002*	Corrisponde a posizione *1*
BIT	*0x0004*	Corrisponde a posizione *2*
DWORD	*0x0008*	Corrisponde a posizione *3*
FLOAT	*0x0010*	Corrisponde a posizione *4*
DFLOAT	*0x0020*	Corrisponde a posizione *5*
STRING	*0x0040*	Corrisponde a posizione *6*
IN_OUTPUT	*0x8000*	Corrisponde a posizione *15*

5.2.1.5 public MaxVariables()

Qui si definisce il numero di variabili che possono essere selezionate dall'elenco delle variabili:

1: la selezione multipla è disattivata nella lista delle variabili. Viene visualizzato un avviso se sono state comunque selezionate più variabili.

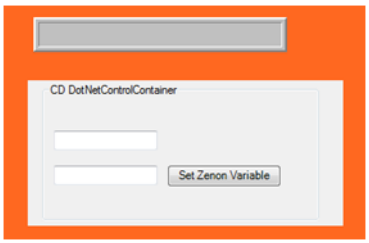
5.2.2 Creare .Net User Control

L´User Control è un semplice controllo che permette di impostare un nuovo valore utilizzando un campo di inserimento (casella di testo). Dopo che si è premuto il pulsante, il valore viene scritto sulla variabile zenon desiderata.

Un'altra funzione deve riconoscere automaticamente un cambiamento di valore della variabile in zenon e visualizzare automaticamente il nuovo valore nel controllo.

Info

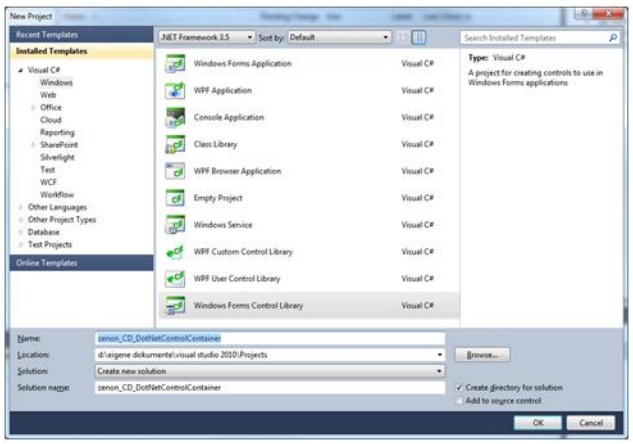
Gli screenshot per questo tema sono disponibili solo in inglese.



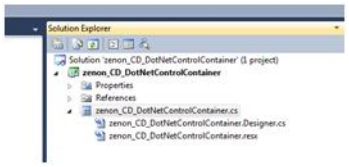
FASI DELLA PROCEDURA

1. Per prima cosa si crea un nuovo progetto in VS, utilizzando il tipo di progetto "**Windows Forms Control Library**"

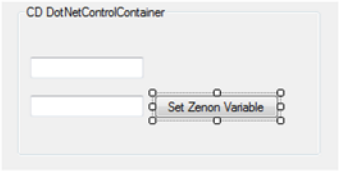
   Importante: impostare Framework su 3.5!
   Se non disponibile, .NET 3.5 deve essere installato sul sistema.



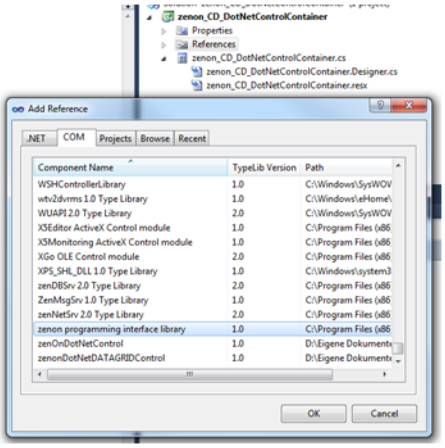
2. Quindi rinominare il file CS "**UserControl**" in "**zenon_CD_DotNetControlContainer.cs**" . I file **Designer.cs** e **.resx** vengono rinominati automaticamente.



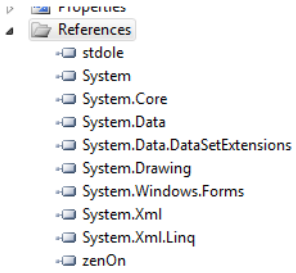
3. Il passo successivo è quello di creare l'User Control. Per questo sono necessarie due caselle di testo per l'input e l'output, nonché un pulsante per inserire un nuovo valore nella variabile zenon.
   Nominare:
   - La prima casella di testo "**txtGetZenonVariable**"
   - La seconda casella di testo "**txtSetZenonVariable**"
   - Il pulsante "**btnSetZenonVariable**"



4. Per accedere agli oggetti zenon, è necessario un riferimento ala zenon Programming Interface. Procedura:
   - Cliccare sul nodo "**References**" nel Solution Explorer
   - Aprire il menù contestuale
   - Selezionare **Add References...**
   - Passate alla scheda **COM**
   - Selezionare **zenon programming interface library**



A questo punto il riferimento "**zenOn**" dovrebbe essere visibile nell'elenco dei riferimenti.



5. Nel passo successivo, si crea una variabile globale di tipo *zenon.Variable* nel codice del programma di **zenon_CD_DotNetControlContainer.cs**.

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Drawing;
5  using System.Data;
6  using System.Linq;
7  using System.Text;
8  using System.Windows.Forms;
9  using zenOn;
10
11 namespace zenon_CD_DotNetControlContainer
12 {
13     public partial class zenon_CD_DotNetControlContainer : UserControl
14     {
15         //This will be needed to get the zenon Variable Container
16         zenOn.Variable m_cVal = null;
17
18         public zenon_CD_DotNetControlContainer()
19         {
20             InitializeComponent();
21         }
22
23
24     }
25 }
26
```

6. Questa variabile viene inizializzata tramite il public Methode **zenOnInit**:

```
23         /// <summary>
24         /// This public Method will be called by the initialization of the control during
25         /// the zenon Runtime.
26         /// </summary>
27         /// <param name="dispElement"></param>
28         /// <returns></returns>
29         public bool zenOnInit(zenOn.Element dispElement)
30         {
31             //Check if zenon Variables are added to the
32             //Control
33             if (dispElement.CountVariable > 0)
34             {
35                 try
36                 {
37                     //Take the first zenon Variable and added
38                     //to the global Variable
39                     m_cVal = dispElement.ItemVariable(0);
40                     //Set Value to the TextBox
41                     txtGetZenonVariable.Text = m_cVal.get_Value(0).ToString();
42                 }
43                 catch { }
44             }
45             return true;
46         }
```

E rilasciata tramite il public Methode **zenOnExit**:

```
47         /// <summary>
48         /// This public Method will be calles by the release of the control during
49         /// the zenon Runtime.
50         /// </summary>
51         /// <returns></returns>
52         public bool zenOnExit()
53         {
54             try
55             {
56                 if (m_cVal != null)
57                 {
58                     //Release the zenon Variable (Com-Object)
59                     System.Runtime.InteropServices.Marshal.FinalReleaseComObject(m_cVal);
60                     m_cVal = null;
61                 }
62             }
63             catch { }
64             return true;
65         }
```

I seguenti metodi determinano se vengono utilizzate variabili e tipi di dati di zenon e quante variabili possono essere passate:

```
107         /// <summary>
108         /// This public Method is needed to link zenon Variables
109         /// to the control.
110         /// </summary>
111         /// <returns></returns>
112         public short CanUseVariables()
113         {
114             return 1; // Only tis Variable is supported
115         }
116
117         /// <summary>
118         /// This public Method returns the Type of
119         /// supported zenon Variables
120         /// </summary>
121         /// <returns></returns>
122         public short VariableTypes()
123         {
124             return short.MaxValue; // all Data Types supported
125         }
126
127         /// <summary>
128         /// This public Method returns the number of
129         /// supported zenon Variables
130         /// </summary>
131         /// <returns></returns>
132         public short MaxVariables()
133         {
134             return 1; // Only 1 Variable should linked to the Control
135         }
```

7. Nel passo successivo, si definisce nel **Click-Event** del pulsante **btnSetZenonVariable** che, cliccando sul pulsante, il valore viene scritto nella casella di testo **txtSetZenonVariable** sul valore della variabile di zenon e quindi il contenuto della casella di testo viene cancellato.

```
98          /// <summary>
99          /// This will be triggert by clicking the Button. The new Value will
100         /// be set to the zenon Variable
101         /// </summary>
102         /// <param name="sender"></param>
103         /// <param name="e"></param>
104         private void btnSetZenonVariable_Click(object sender, EventArgs e)
105         {
106             //Set Value from TextBox to the zenon Variable
107             m_cVal.set_Value(0,txtSetZenonVariable.Text.ToString());
108             this.txtSetZenonVariable.Text = string.Empty;
109         }
```

8. Per reagire ad un cambiamento di valore della variabile, è necessario il **Paint-Event** del controllo. Il **Paint-Event** si attiva anche quando il valore della variabile di zenon inizializzata cambia, e può quindi essere utilizzato per aggiornare i valori. Poiché le variabili a cui si fa riferimento nell'elemento ActiveX di zenon, sono automaticamente *advised*, il **zenon.OnlineVariable** Container può generalmente essere omesso nel controllo.

```
111         /// <summary>
112         /// This will be triggert by painting the User Control or the Value of the Variable changed.
113         /// After the value of the Variable changed the Control will be new painted and the new Value
114         /// will be set to the Textbox.
115         /// </summary>
116         /// <param name="sender"></param>
117         /// <param name="e"></param>
118         private void zenon_CD_DotNetControlContainer_Paint(object sender, PaintEventArgs e)
119         {
120             if (m_cVal != null)
121             {
122                 this.txtGetZenonVariable.Text = m_cVal.get_Value(0).ToString();
123                 return;
124             }
125             else
126             {
127                 this.txtGetZenonVariable.Text = "Variable Value";
128                 return;
129             }
130
131         }
```

PANORAMICA DEL CODICE

Qui di seguito una panoramica dell´intero codice:

```
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 called 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.ReleaseComObject(m_cVal);
m_cVal = null;
}
}
catch { }
return true;
}
 /// <summary>
```

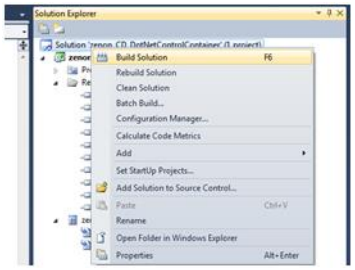
```
 /// This public Method is needed to link zenon Variables
 /// to the control.
 /// </summary>
 /// <returns></returns>
public short CanUseVariables()
{
 return 1; // Only this Variable is supported
}
/// <summary>
/// This public Method returns the Type of
/// supported zenon Variables
/// </summary>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

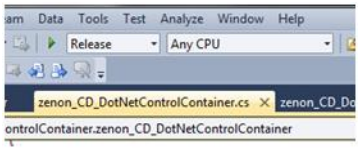
}

CREARE RELEASE

Infine creare un Release per poter integrare la DLL completa in zenon e/o nel **CD_DotNetControlContainer**.



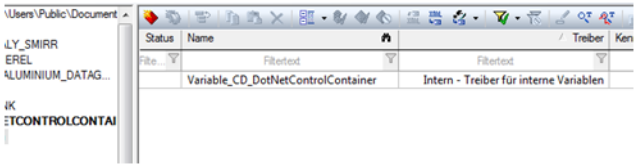
A tal scopo è necessario passare da **Debug** a **Release** nei Settings.



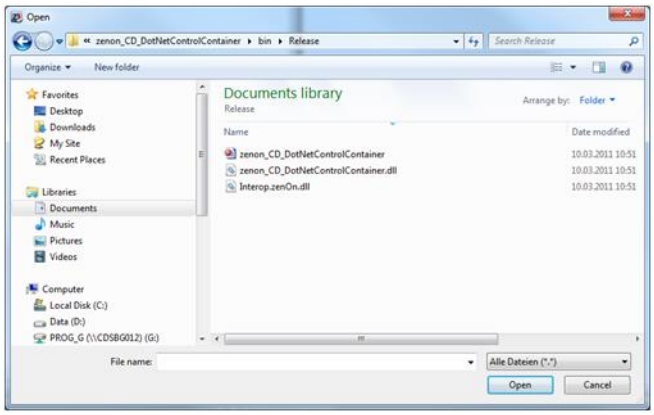
5.2.3 Aggiungere CD_DotNetControlContainer e .NET User Control

Per preparare il progetto zenon e aggiungere il **CD_DotNetControlContainer** e il **.NET User Control**, eseguire le seguenti operazioni:

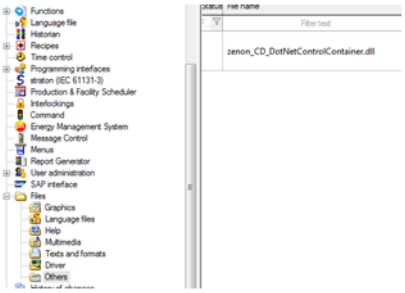
1. Creare una variabile interna di tipo *stringa* e impostare la lunghezza della stringa su *30*.



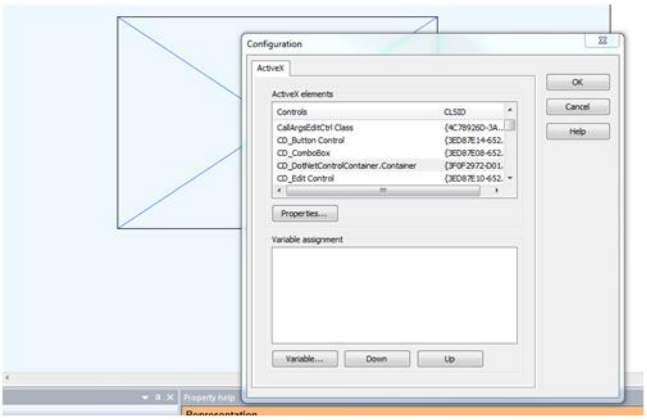
2. Nel nodo *Progetto/File/Altri* di zenon, inserire la DLL del .NET User Control creato. Nel corso di questo processo, il file viene copiato nella cartella **Additional** a livello di file system.

3. 

La DLL si trova nella cartella di output del progetto Visual Studio, alla voce *...\bin\Release\zenon_CD_DotNetControlContainer.dll*.

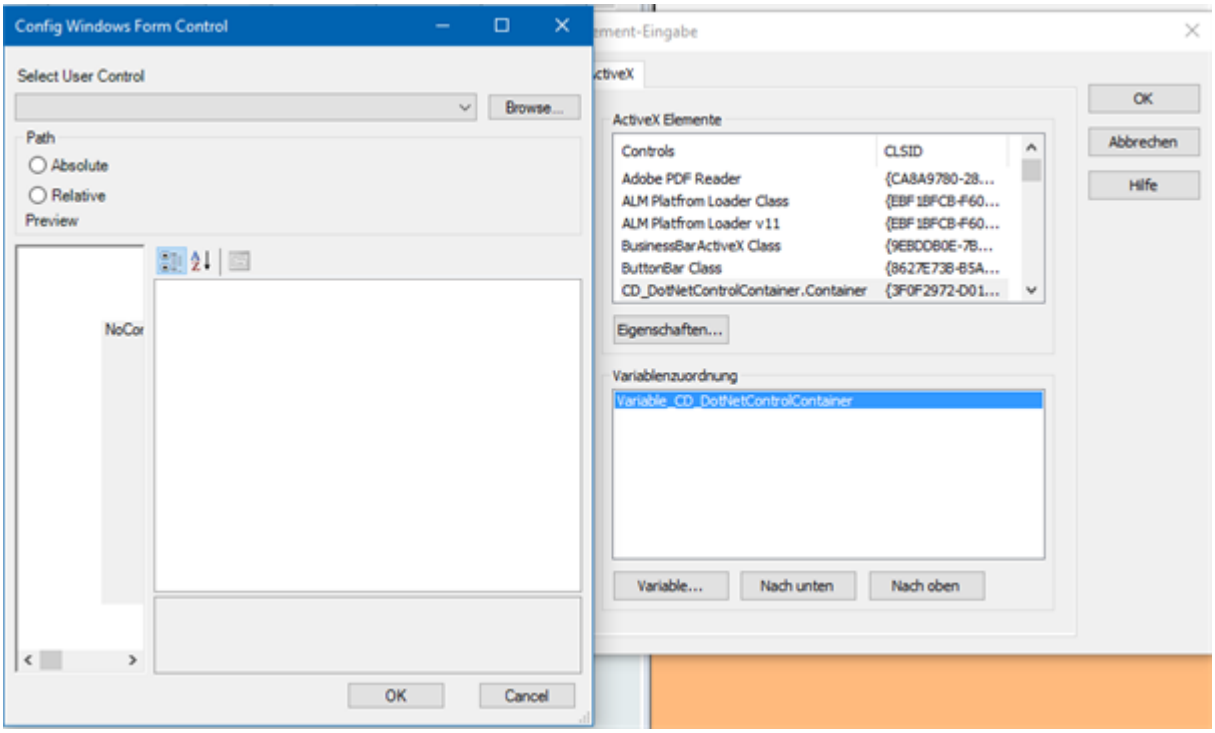


4. Nel progetto, selezionare l'elemento ActiveX e trascinarlo in un'immagine zenon.
   - Si apre la finestra di dialogo **Configurazione**.
   - Selezionare il **CD_DotNetControlContainer.Container** Control.



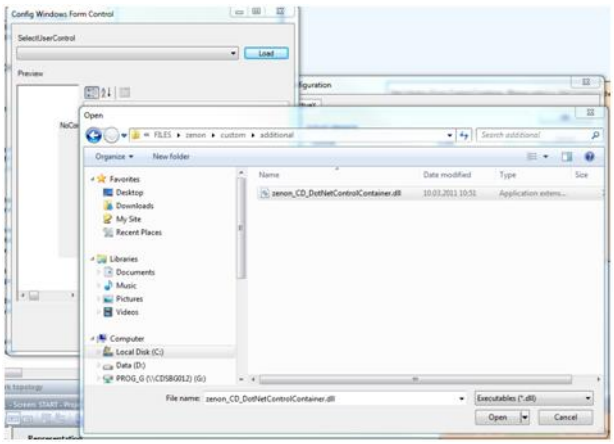
5. Per incorporare il .NET User Control nel **CD_DotNetControlContainer** Control:
   - Fare clic sul pulsante **Proprietà.**

- Si apre una nuova finestra di dialogo.



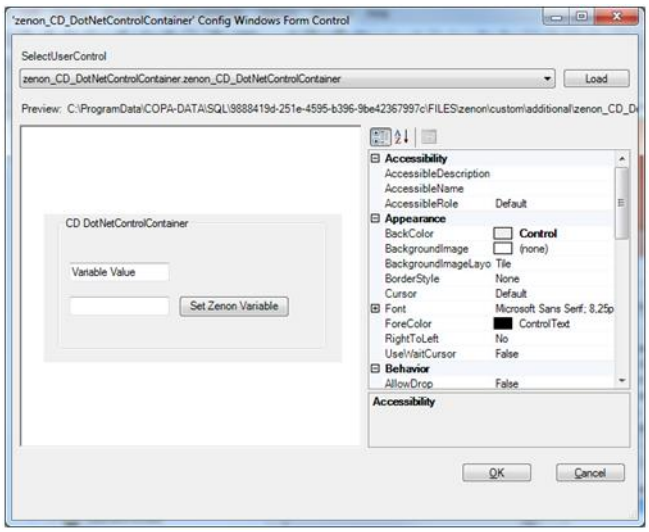
- Fare clic sul pulsante **Load** per selezionare il percorso della cartella di progetto, ad esempio: *C:\ProgramData\COPA-DATA\SQL\9888419d-251e-4595-b396-9be42367997c\FILES\zenon\custom\additional\zenon_CD_DotNetControlContainer.dll*

Nota: i controlli dovrebbero essere salvati sempre nella cartella **Additional** del progetto. In questo caso, infatti, vengono presi in considerazione dal sistema per i backup e trasferimento di file. Il percorso deve essere definito in modo relativo al luogo di salvataggio. In alternativa, si può usare per **Path** l´opzione *Absolute*, ma è necessario assicurarsi che la stessa struttura di directory esista sul sistema di destinazione.

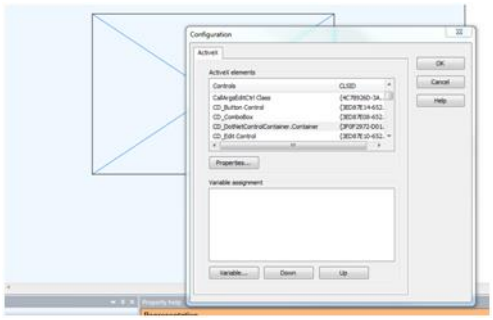


Ora dovrebbe essere visualizzato il .NET User Control.

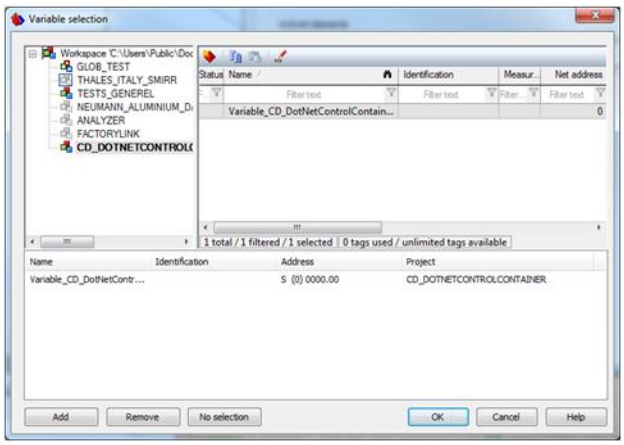
Confermare la finestra di dialogo cliccando su **OK**.



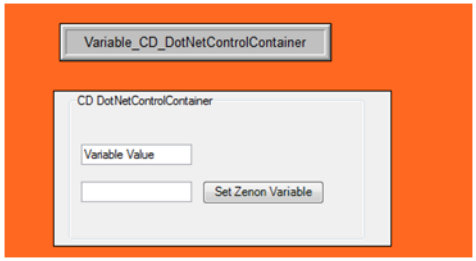
6. Nell'ultimo passo, si usa il pulsante **Variabili** per collegare una variabile al controllo.



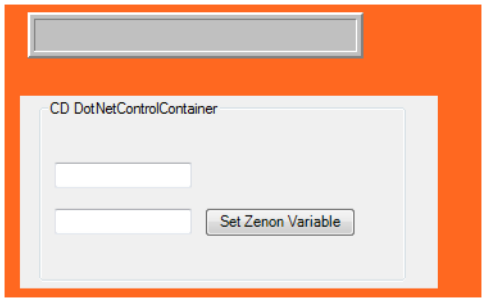
La prima variabile selezionata viene automaticamente collegata alla variabile definita globalmente (.NET UserControl) tramite il metodo `public zenonInit`. Tuttavia, il collegamento al controllo viene creato solo dopo l'avvio del Runtime.



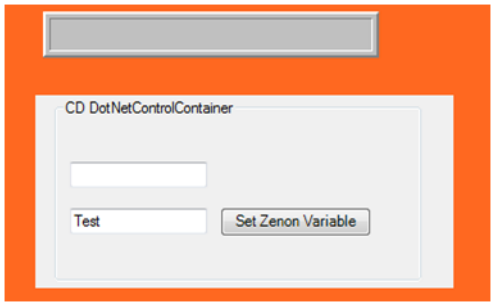
Quindi collegare la variabile interna con un elemento di testo.



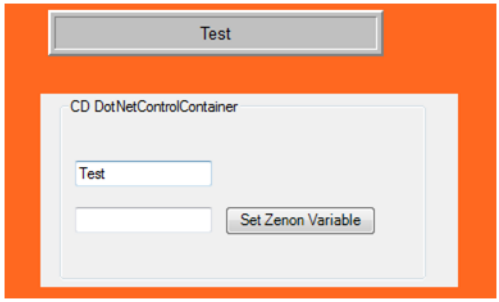
7. Dopo l´avvio del Runtime, il controllo è ancora vuoto.



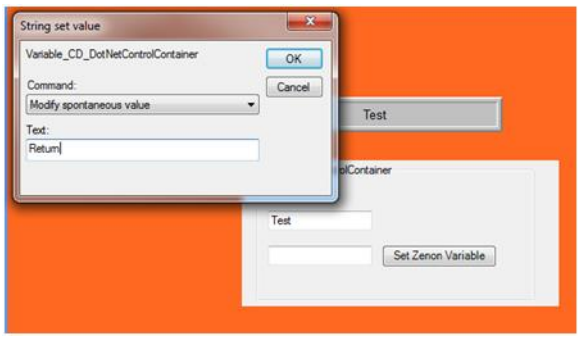
Se un valore viene scritto nella seconda casella di testo e poi confermato con il pulsante **Set zenon Variable**, il valore viene scritto nella variabile zenon. (L´evento **btnSetZenonVariable_Click** viene eseguito.)



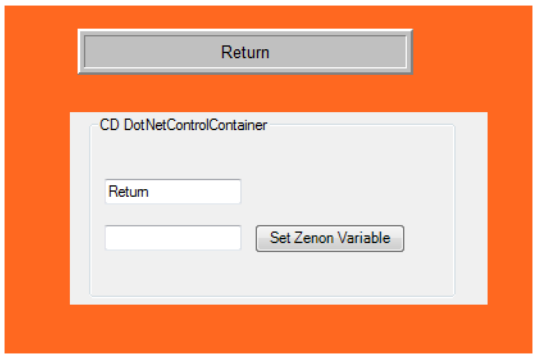
Questo viene visualizzato anche nell´elemento di testo zenon.



Se il valore viene modificato direttamente nell´elemento di testo di zenon,



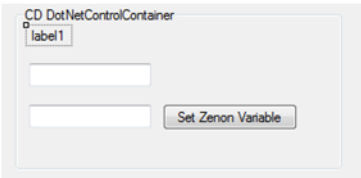
il valore viene scritto direttamente nella prima casella di testo tramite il **Paint Event** del.NET Control.



5.2.4 Accesso al controllo utente tramite VSTA o VBA

Questo esempio illustra l´accesso via VSTA. La procedura è la stessa con VBA.

1. Estendere il controllo con un **label** e denominarlo l**blzenonInfo**. In questo **label** deve essere visualizzato il valore di un´altra variabile zenon. Il nuovo valore deve essere impostato tramite una macro VSTA.

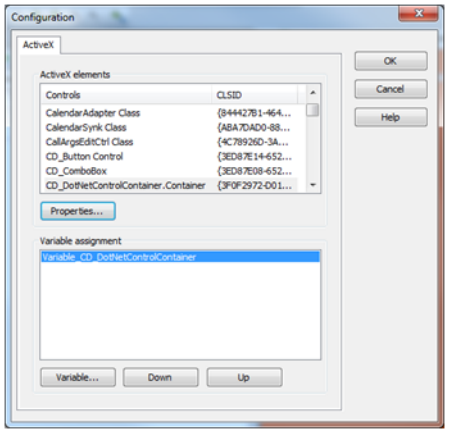


2. Estendere il Code di una Property (**Information**) e aggiungere le proprietà **get** e **set** alle Properties. Queste consentono di leggere e scrivere il testo del **label**.

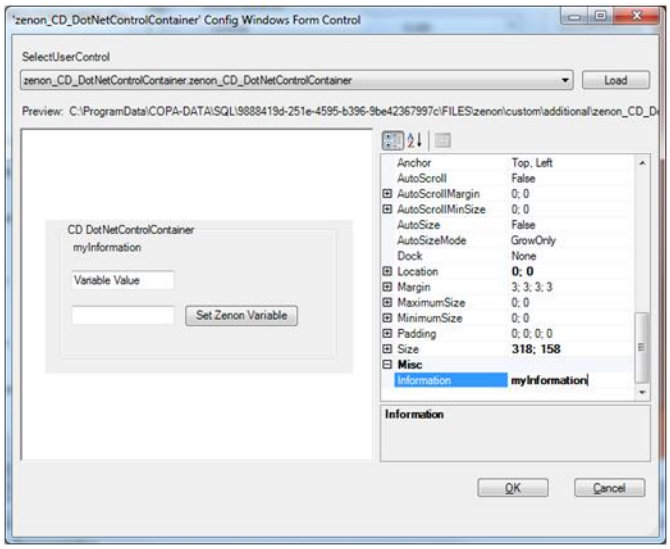
```
13      public partial class zenon_CD_DotNetControlContainer : UserControl
14      {
15          //This will be needed to get the zenon Variable Container
16          zenOn.Variable m_cVal = null;
17
18          public zenon_CD_DotNetControlContainer()
19          {
20              InitializeComponent();
21          }
22
23          public string Information
24          {
25              set{this.lblZenonInfo.Text = value;}
26              get { return this.lblZenonInfo.Text; }
27          }
28
```

3. Creare un nuovo release per l´User Control e copiarlo nella cartella *additional* del progetto zenon.
   Non dimenticare: chiudere prima l´Editor di zenon!
   Cancellare la vecchia DLL e riavviare l'Editor di zenon. Se la DLL si trova ancora nella cartella, cancellarla una seconda volta. Ora la DLL modificata può essere importata di nuovo. Il **CD_DotNetContainerControl** e l´ActiveX vengono aggiornati automaticamente.

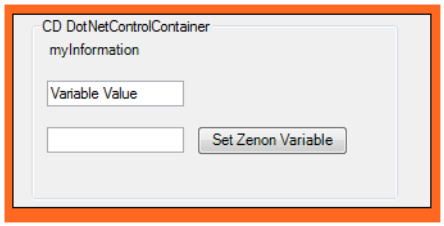
4. Cliccare nell´Editor di zenon sull'ActiveX e aprire la finestra delle proprietà.



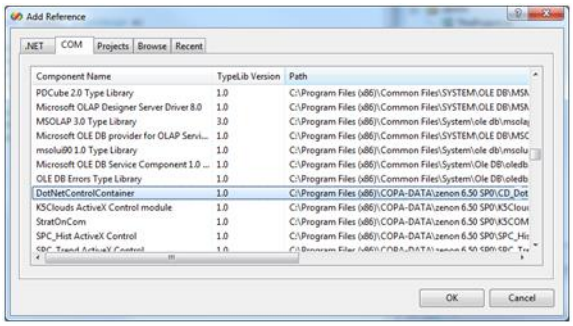
Ora è possibile visualizzare la nuova proprietà **Information** nella finestra di selezione del controllo e impostare anche un valore.



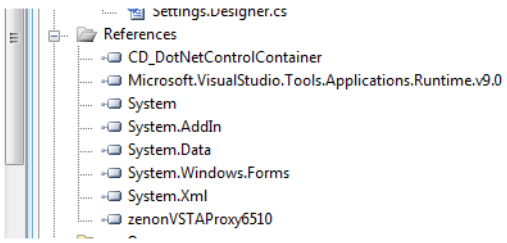
Questo valore viene impostato anche nel Control ("**myInformation**")



5. Per poter lavorare con il **CD_DotNetControlContainer** in VSTA oppure VBA, è necessario prima il riferimento al controllo. Dopo l'apertura del VSTA per il progetto **(ProjectAddin)**, deve essere aggiunto il riferimento al **CD_DotNetControlContainer**.



Inoltre, è necessario aggiungere anche l´assembly **System.Windows.Forms**.



6. Il seguente codice può essere utilizzato per resettare il valore della proprietà **Information**:

```
public void Macro_Test()
{
    try
    {
        zenOn.IElements zElements = this.DynPictures().Item("START").Elements();
        zenOn.IElement zElement = zElements.Item("ActiveX_1");

        // Create a Variable of Type CD_DotNetControlContainer.Container and get the zenon AktiveX Element
        // with a cast
        CD_DotNetControlContainer.Container zAktiveX = (CD_DotNetControlContainer.Container)zElement.AktiveX();

        //With using SetExternalUserControlProperty and the name of the Property "Information" we can set
        // a new Value "New Information" to the Property
        if (zAktiveX.GetExternalUserControlProperty("Information").Equals("myInformation"))
        {
            zAktiveX.SetExternalUserControlProperty("Information", "New Information");
        }
        else
        {
            zAktiveX.SetExternalUserControlProperty("Information", "myInformation");
        }

    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.Print("ERROR : " + ex.Message + " " + ex.Source);
    }
}
```

7. Alla fine:

- Creare una funzione zenon **Esegui macro VSTA**
- Collegare la funzione ad un pulsante.

A Runtime, il **label** passa da **myInformation** a **New Information** quando si clicca sul pulsante.

Cliccando di nuovo sul pulsante, si torna alla situazione precedente.

5.3 Esempio .NET Control come ActiveX (C#)

L'esempio seguente descrive un controllo .NET che viene eseguito come ActiveX Control in zenon.

La creazione e l'integrazione avviene in quattro fasi:

1. Creare Windows Form Control (A pagina: 30)
2. Convertire .NET User Control in Dual Control (A pagina: 33)
3. Lavorare nell´Editor con ActiveX via VBA (A pagina: 37)
4. Collegare variabili di zenon al .NET User Control (A pagina: 38)

Attenzione

Quando si utilizzano gli oggetti COM di zenon nei controlli utente o nelle applicazioni esterne di creazione propria, questi devono essere rilasciati utilizzando il metodo **Marshal.ReleaseComObject**. Il rilascio tramite il metodo **Marshal.FinalReleaseComObject** non deve essere utilizzato, perché Add-In di zenon smettono di funzionare quando si usa questo metodo.

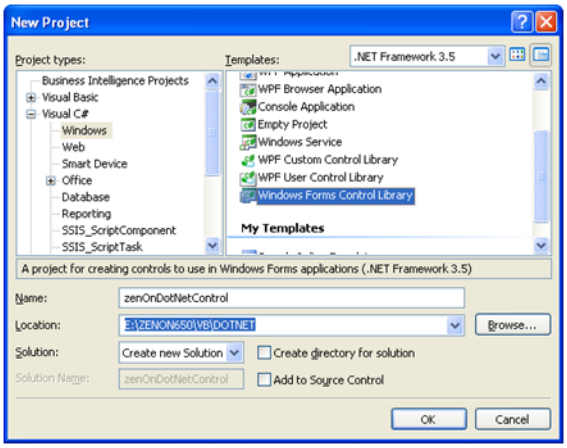
Info

Gli screenshot per questo tema sono disponibili solo in inglese.

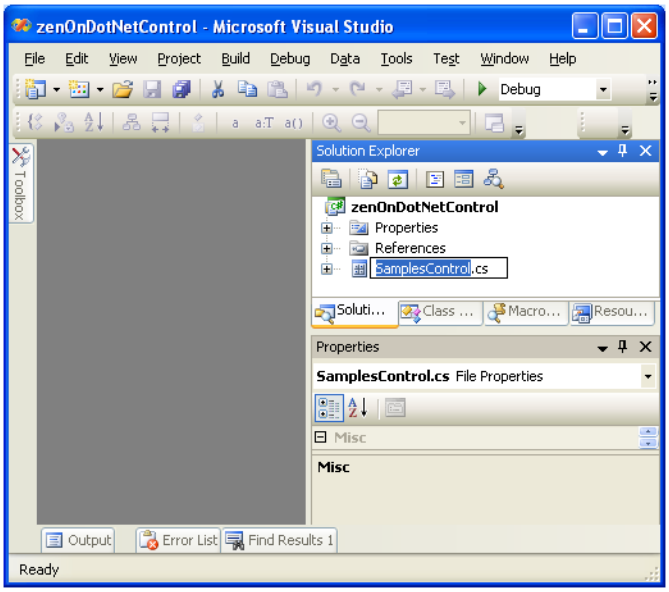
5.3.1 Creare Windows Form Control

Per creare un controllo Windows Form:

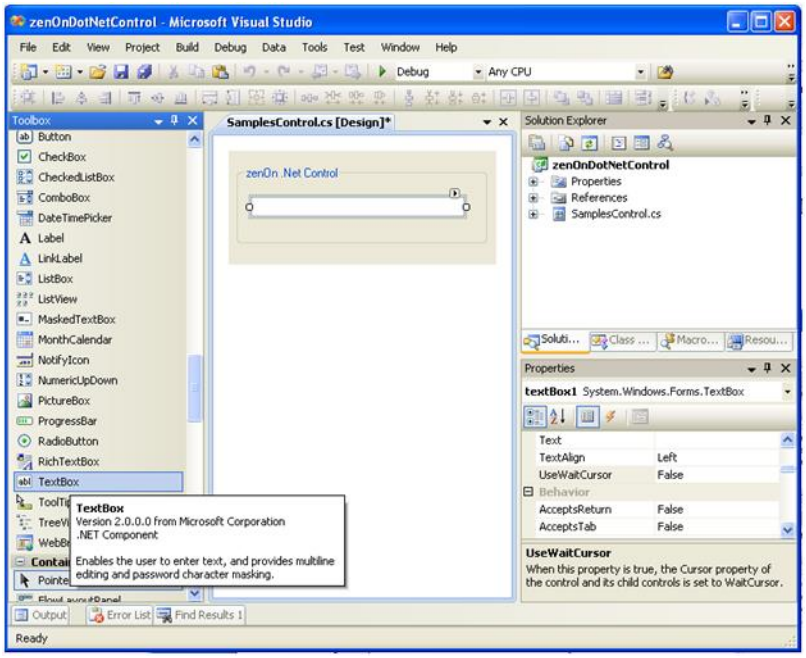
1. Avviare Visual Studio 2008 e e creare un nuovo progetto Windows **Form Control Library**:



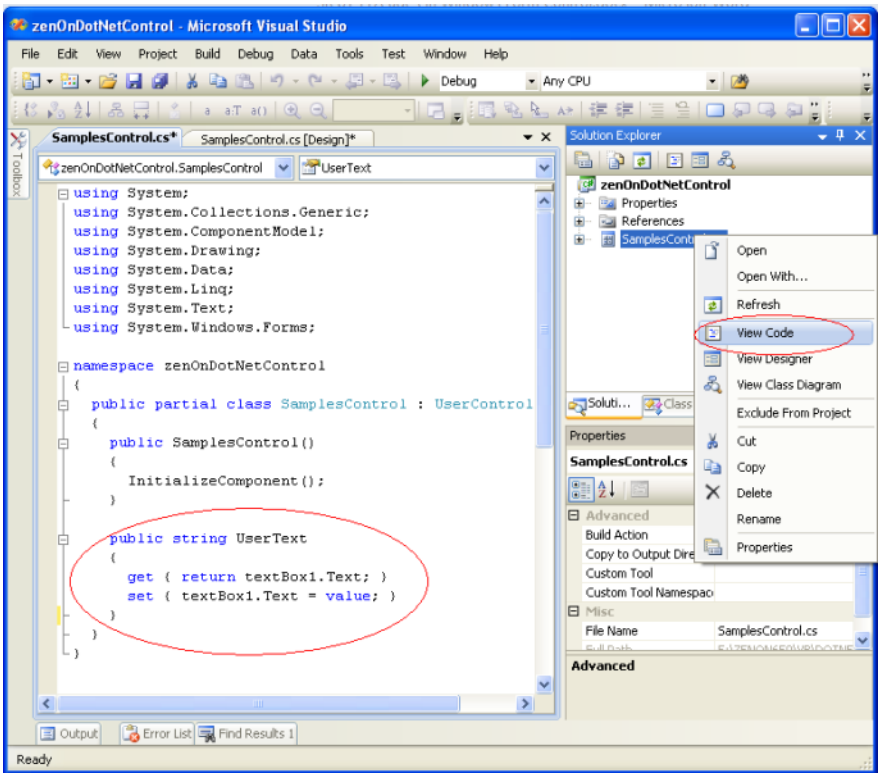
2. Rinominare il controllo predefinito assegnandogli il nome desiderato.
   In questo esempio: **SampesControl.cs**.



3. Aprire il Control Designer e inserire i controlli desiderati, nel nostro caso una casella di testo:



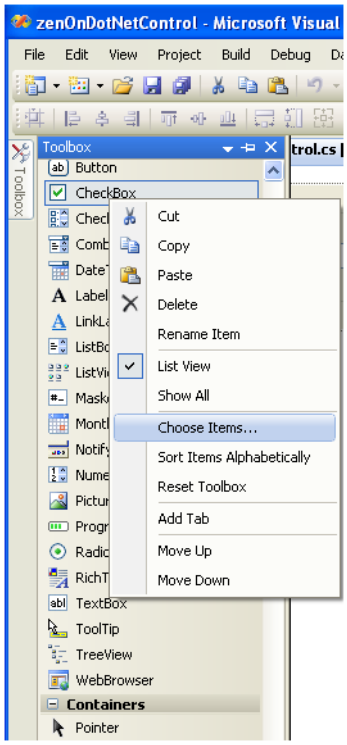
4. I controlli di solito hanno delle proprietà. Aprire Code Designer tramite **View Code** e aggiungere le proprietà disponibili esternamente.
   In questo esempio: Proprietà visibile esternamente "**UserText**" con accesso **get** e **set**, che contiene il testo della casella di testo:



5. Compilare il progetto.

Il controllo Windows Form può ora essere utilizzato in altri progetti Windows Forms.

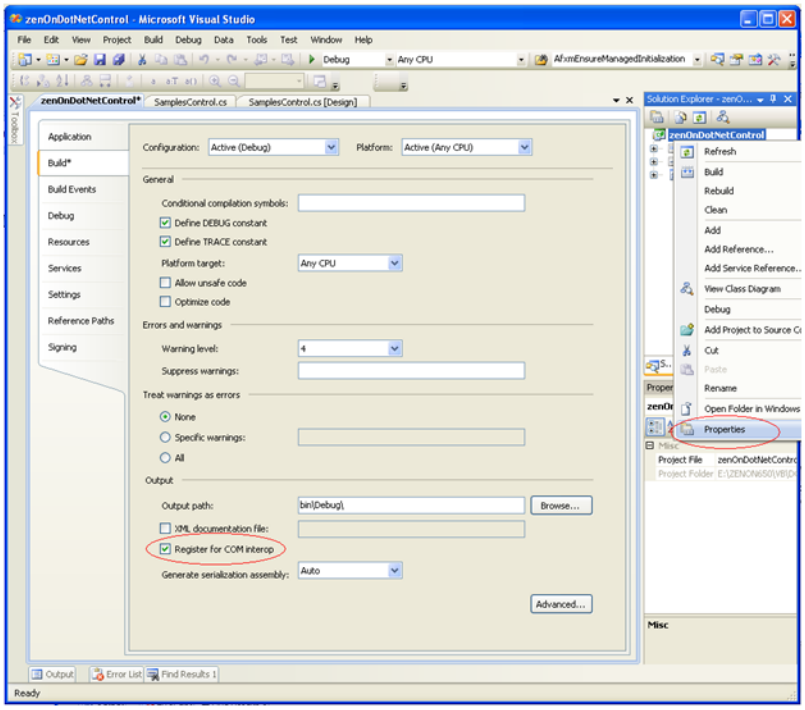
Importante: il controllo deve essere inserito manualmente nel Control Toolbox utilizzando **Choose Items**.



5.3.2 Convertire .NET User Control in Dual Control

Per convertire il .NET in un Dual Control, deve prima essere attivata l'interfaccia COM per l'ActiveX.

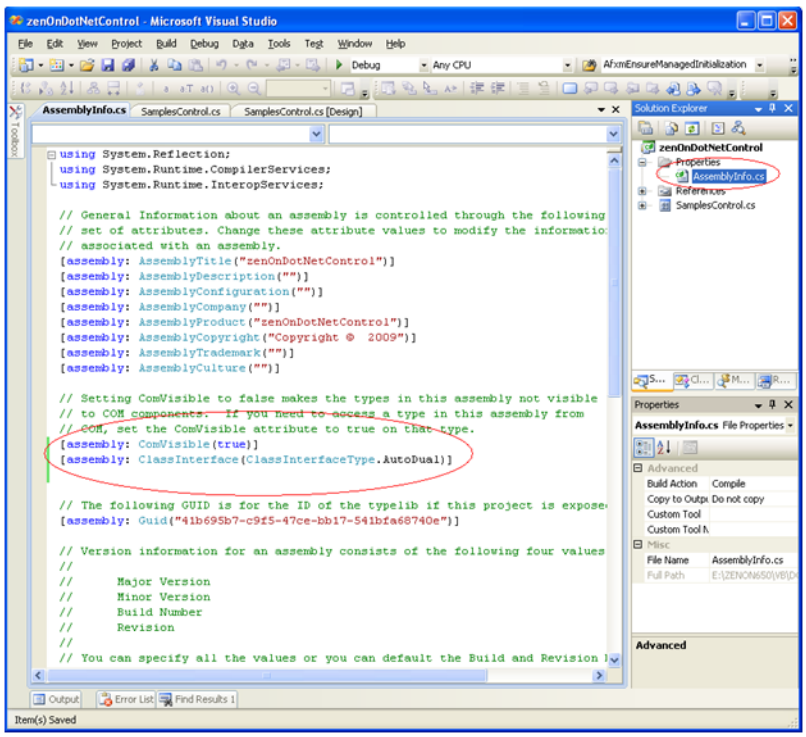
1. Aprire il progetto e attivare la proprietà **Register for COM interop** nelle impostazioni **Build**:



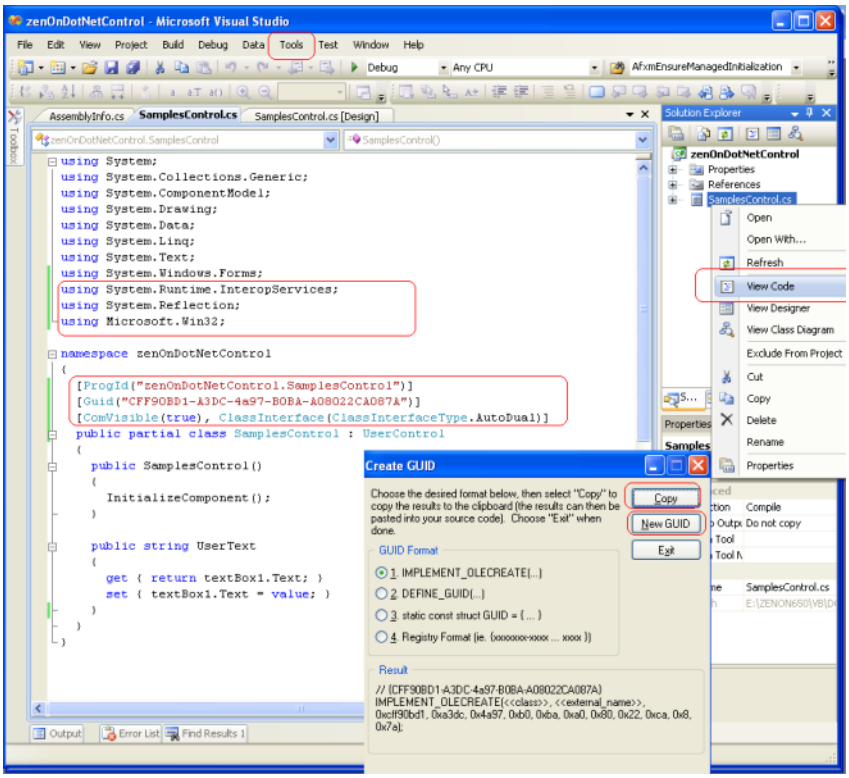
2. Aprire il file **AssemblyInfo.cs** e
   - Impostare l´attributo **ComVisible** su *true*
   - Aggiungere l´attributo **ClassInterface**

   [`assembly`: ComVisible(`true`)]

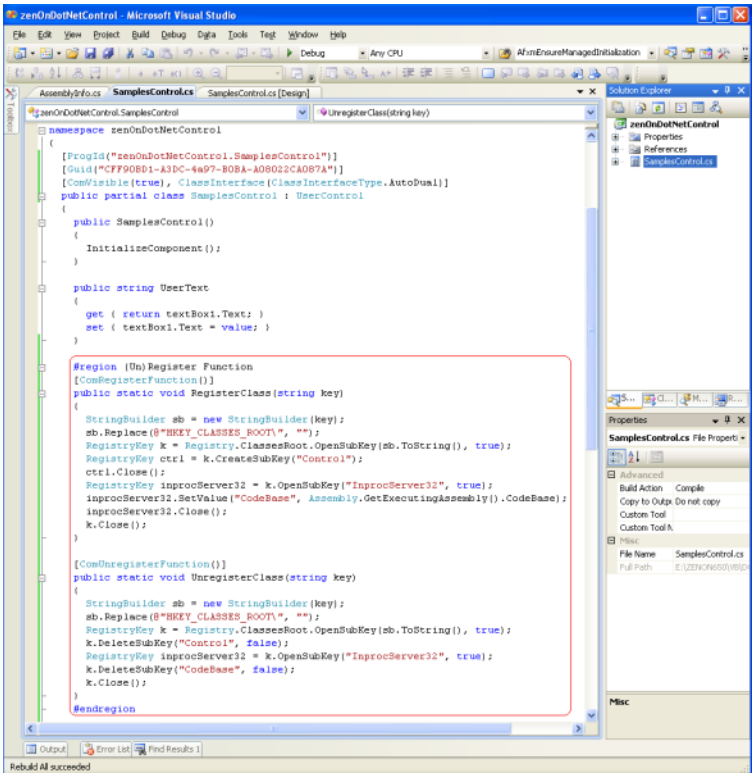
[assembly: ClassInterface(ClassInterfaceType.AutoDual)]



3. Aprire il Code Designer tramite **View Code** e aggiungere gli attributi ActiveX e gli inserimenti **using** necessari. Utilizzare il menù **Tools/Create GUID** per creare un nuovo GUID per l'attributo GUID:



4. Affinché il controllo possa essere selezionato anche come controllo di interfaccia ActiveX, alla classe di controllo devono essere aggiunte le seguenti funzioni:
   - RegisterClass
   - UnregisterClass



   Il controllo può quindi essere registrato nella Registry.

5. Ricompilate il progetto.

   Il controllo Windows Form è ora abilitato ActiveX ed è stato registrato automaticamente durante il Rebuild . Un ulteriore file TypeLib **zenOnDotNetControl.tlb** è stato creato nella directory di output.

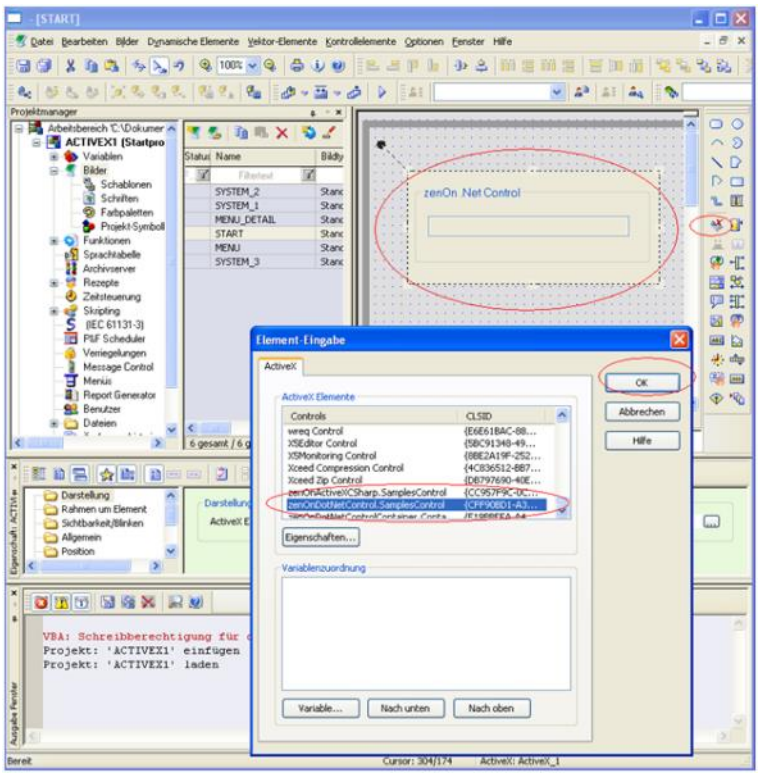
6. Per utilizzare il controllo su un altro computer:

   a) Copiare il file DLL e il file TLB sul computer di destinazione

   b) Registrare i file tramite la riga di comando con:
   [Posizione di memorizzazione .NET-Framework]\RegAsm.exe "MyControl.dll" /codebase

   Esempio a 64-Bit Sistema operativo Windows:
   C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe "MyControl.dll" /codebase
   Esempio a 32-Bit Sistema operativo Windows:
   C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe "MyControl.dll" /codebase

7. Inserire l'estensione del Windows Form control come ActiveX Control nell´Editor di zenon:



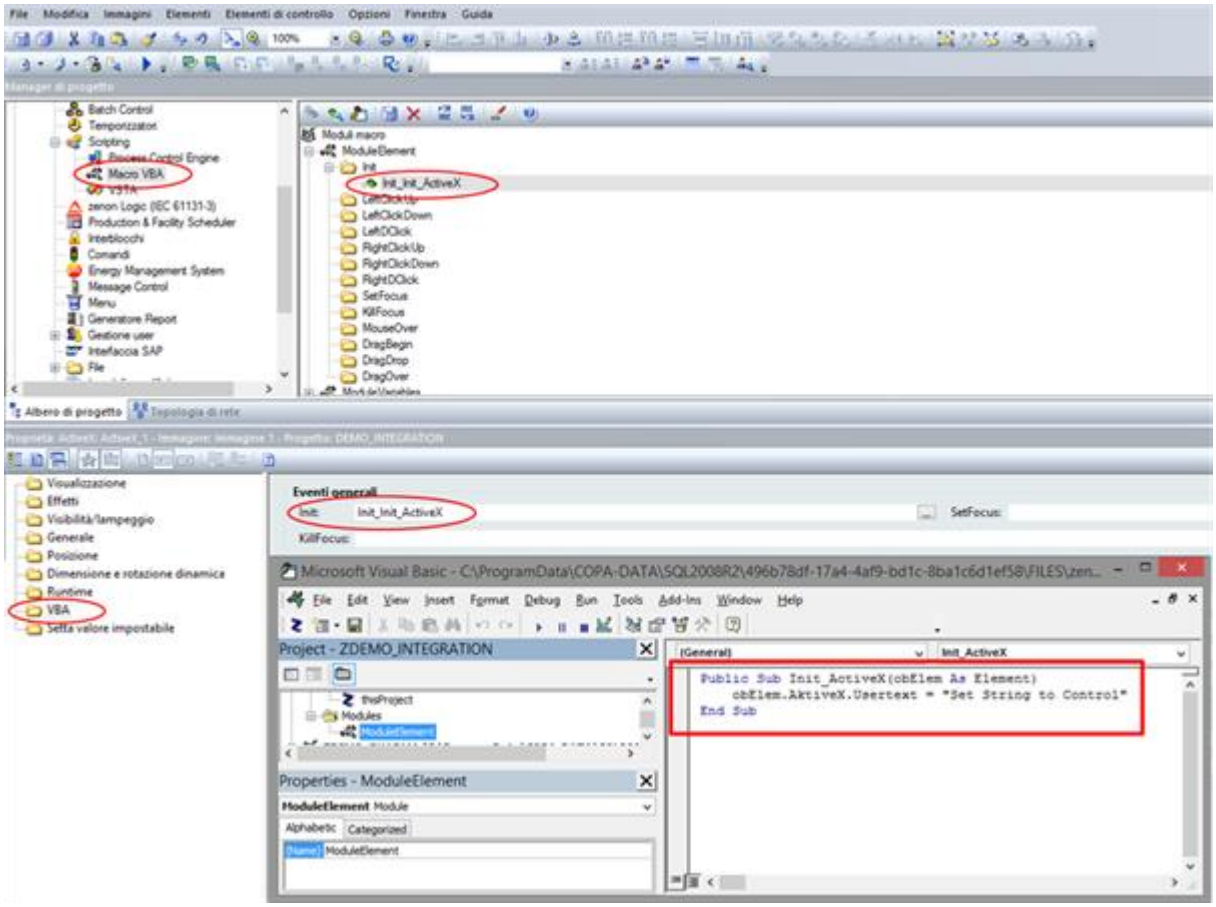
5.3.3 Lavorare nell´Editor con ActiveX via VBA

Per accedere alle proprietà del controllo nell´Editor di zenon:

1. Creare nell´Editor di zenon, nel nodo **Scripting/Macro-VBA**, una nuova macro **Init** con il nome **Init_ActiveX**.

   Con questa macro è poi possibile accedere a tutte le proprietà esterne tramite **obElem.AktiveX...**.

2. Assegnare questa macro al controllo ActiveX tramite le proprietà **VBA Makros/Init** dell´elemento ActiveX.



ESEMPIO MACRO INIT

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

5.3.4 Collegare variabili di zenon al .NET User Control

In zenon si ha la possibilità di estendere un controllo ActiveX integrandolo con funzioni speciali, e di accedere all API di zenon.

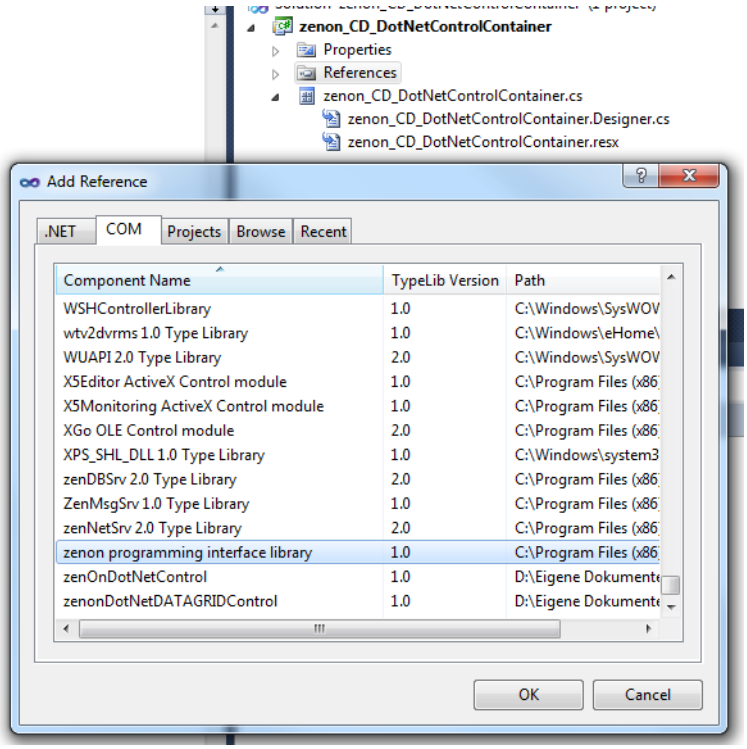
METODI NECESSARI

- public bool zenOnInit (A pagina: 40) (Chiamato durante l'inizializzazione del controllo nel Runtime di zenon.)
- public bool zenOnInitED (A pagina: 41) (viene utilizzato nell´Editor.)

- public bool zenOnExit() (A pagina: 41) (chiamato durante la distruzione del controllo nel Runtime di zenon.)
- public bool zenOnExitED() (A pagina: 41) (viene utilizzato nell´Editor)
- public short CanUseVariables() (A pagina: 41) (supporta il collegamento di variabili)
- public short VariableTypes() (A pagina: 41) (tipi di dati supportati dal controllo)
- public MaxVariables() (A pagina: 42) (numero massimo di variabili che possono essere collegate al controllo.)

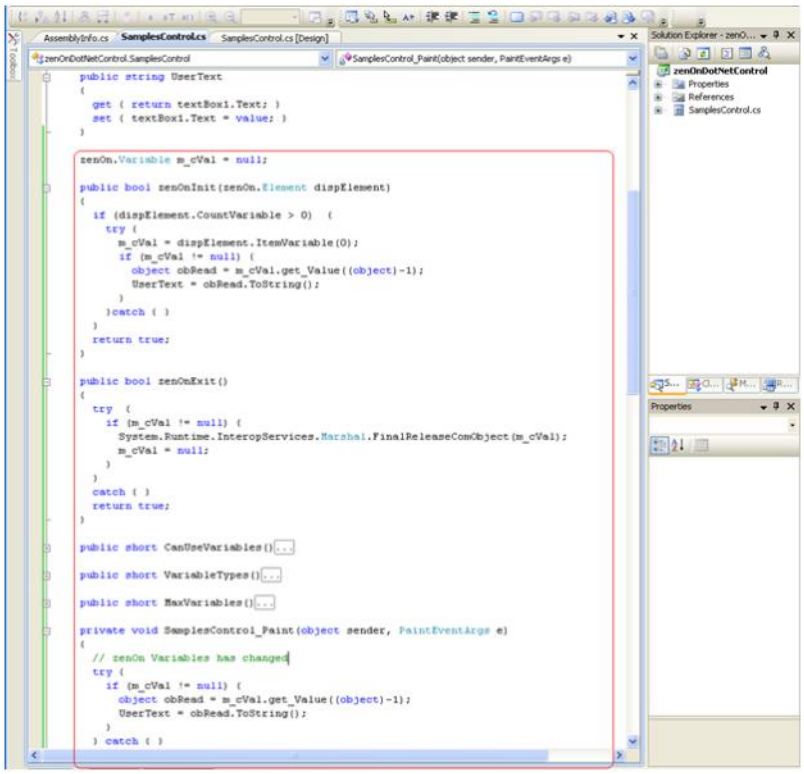
AGGIUNGI RIFERIMENTO

1. In Microsoft Visual Studio, selezionare alla voce **Add References** la zenon Runtime Object Library, per accedere all'API di zenon nel controllo.



2. Per accedere all'intera zenon API, includere le funzioni avanzate nel codice di classe del controllo.

Nel nostro esempio, l'oggetto COM di una variabile di zenon è memorizzato temporaneamente in un **Member**, in modo che sia possibile accedervi successivamente nell'evento **Paint** del controllo.



5.3.4.1 public bool zenOnInit(zenOn.Element dispElement)

Questo metodo viene utilizzato a Runtime per passare al controllo ActiveX un puntatore per la Dispatch Interface dell'elemento dinamico. Questo puntatore può quindi essere utilizzato per indirizzare le variabili zenon collegate all´elemento dinamico.

È possibile impostare l'ordine delle variabili trasferite nella finestra di dialogo "Introduzione elemento" utilizzando i pulsanti **Giù** o **Su**. Il dialogo "Introduzione elemento" si apre quando:

- Si fa doppio clic sull´elemento ActiveX
- Si seleziona **Proprietà**, o
- Si seleziona nella finestra delle proprietà la proprietà **ActiveX Einstellungen** del nodo **Darstellung**.

5.3.4.2 public bool zenOnInitED(zenOn.Element dispElement)

Corrisponde a public bool zenOnInit (A pagina: 40)e viene eseguito all'apertura dell'ActiveX nell'Editor (doppio clic su ActiveX).

5.3.4.3 public bool zenOnExit()

Questo metodo è richiamato dal Runtime di zenon alla chiusura del controllo ActiveX. Tutti i Dispatch Pointer per le variabili dovrebbero essere rilasciati.

5.3.4.4 public bool zenOnExitED()

Corrisponde a public bool zenOnExit() (A pagina: 41) e viene eseguito alla chiusura dell'ActiveX nell'Editor. Questo permette anche di reagire a cambiamenti (per es. cambiamenti di valore) nell'Editor.

5.3.4.5 public short CanUseVariables()

Questo metodo restituisce *1* se il controllo zenon può usare variabili, e *0* in caso contrario

- *1*: Solo variabili zenon dei tipi indicati con il metodo `VariableTypes` e nella quantità indicata con il metodo `MaxVariables`, possono essere specificati per l'elemento dinamico (usando il pulsante **Variabili**)
- *0*: Se `CanUseVariables` restituisce *0*, o se il controllo non ha questo metodo, possono essere specificate variabili di tutti i tipi e senza limitazioni di numero. Tuttavia, queste possono essere utilizzate a Runtime solo tramite VBA.

5.3.4.6 public short VariableTypes()

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

Parametro	Valore	Descrizione
WORD	*0x0001*	Corrisponde a posizione *0*
BYTE	*0x0002*	Corrisponde a posizione *1*
BIT	*0x0004*	Corrisponde a posizione *2*
DWORD	*0x0008*	Corrisponde a posizione *3*

Parametro	Valore	Descrizione
FLOAT	*0x0010*	Corrisponde a posizione *4*
DFLOAT	*0x0020*	Corrisponde a posizione *5*
STRING	*0x0040*	Corrisponde a posizione *6*
IN_OUTPUT	*0x8000*	Corrisponde a posizione *15*

5.3.4.7 public MaxVariables()

Qui si definisce il numero di variabili che possono essere selezionate dall'elenco delle variabili:

1: la selezione multipla è disattivata nella lista delle variabili. Viene visualizzato un avviso se sono state comunque selezionate più variabili.

6 WPF

Con l´elemento dinamico **WPF** si possono implementare e visualizzare file WPF/XAML validi in zenon.

La documentazione dettagliata sugli elementi WPF in zenon, così come i tutorial per designer, sviluppatori e progettisti, si trovano nel manuale Elemento WPF di zenon.

Questo manuale tratta i seguenti argomenti:

- Nozioni di base
- Guida per i progettisti
- Guida per gli sviluppatori
- Progettazione nell´Editor di zenon