

DIDASCALIA,C,254
Miglioramenti
Scelta

Inizio

Definizione delle Classi

Interface Part - Livelli di accesso

Definizione delle propriet...

Definizione prototipi

Dettagli

Nota

Procedura Interna

Procedura Interna
Dettagli

Dettagli
L'altro modo

Definizione Oggetto

Definizione delle propriet...

Definizione delle propriet...

Dettagli

Prototipo

Implementation Part

Funzione

Nota

Procedure Register

Modifica Procedura Register

Nota

Conclusione

Nel Prossimo Numero

CODICE,C,254	NUILU
	-1 0
	-1 0
unit T_MioVCL1;	
interface	
uses	
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;	
	0 7
type	
TMioVCL1a = class(TComponent)	7 2
private	
{ Private declarations }	
N1,N2,R:Integer;	9 3
Property Num1:Integer Read N1 Write SetNum1;	
Property Num2:Integer Read N2 Write SetNum2;	20 2
Procedure SetNum1(Value:Integer);	
Procedure SetNum2(Value:Integer);	12 2
Procedure SetNum1(Value:Integer);	-1 0
//Procedure interne del componente TMioVCL1a	46 1
Procedure TMioVCL1a.SetNum1(Value:Integer);	
Begin	
N1:=Value;	
R:=N1+N2;	
End;	48 5
Procedure TMioVCL1a.SetNum2(Value:Integer);	
Begin	
N2:=Value;	
R:=N1+N2;	
End;	54 5
N1:=Value;	-1 0
R:=N1+N2;	-1 0
	-1 0
type	
TMioVCL1b = class(TComponent)	
private	
{ Private declarations }	
N1,N2,R:Integer;	25 5
Property Num1:Integer Read N1 Write N1;	
Property Num2:Integer Read N2 Write N2;	37 2

Sheet1

Property Risultato:Integer Read GetRisultato Write R;	39	1
Property Risultato:Integer Read GetRisultato Write R;	-1	0
Function GetRisultato:Integer;	30	1
//Funzione interna del componente TMioVCL1b	60	1
Function TMioVCL1b.GetRisultato:Integer;		
Begin		
Result:=N1+N2;		
End;	62	4
Result:=N1+N2;	64	1
procedure Register;	42	1
procedure Register;		
begin		
RegisterComponents('Samples', [TMioVCL1a,TMioVCL1B]);		
end;	67	4
RegisterComponents('Samples', [TMioVCL1a,TMioVCL1B]);		
Š uguale a		
RegisterComponents('Samples', [TMioVCL1a]);		
RegisterComponents('Samples1', [TMioVCL1B]);	-1	0
	-1	0
	-1	0

DESCRIZIONE, C, 254

Come si era detto, apporteremo alcune semplicissime modifiche al componente creato nella Lezione 1a. Elimineremo il metodo
Per fare ciò è possibile procedere in due modi che portano ovviamente allo stesso risultato, ma che sono strutturalmente molto

Manteniamo identica l'Unit Header e l'Uses Clause del componente, ricordando che comunque potranno essere modificati in

Definiamo, come nella lezione precedente, il tipo di componente che creeremo. Il nome è stato modificato per maggiore comode

Ridefiniamo le variabili interne come per il componente precedentemente creato.

Ridefiniamo le proprietà.... La sintassi è stata modificata, in particolare allo specificatore Write non è stata assegnata una variabile

Definiamo come procedure Private i prototipi delle due procedure che vengono automaticamente chiamate da Delphi al momento
Il parametro Value della procedura può avere un qualsiasi nome valido di variabile, ma deve essere dello stesso tipo della proprietà
Delphi assegnerà... automaticamente il valore del parametro anche a DesignTime.

Spostiamoci quindi nella sezione Implementation della unit per scrivere le due procedure che abbiamo appena dichiarato.

Scriviamo la procedura. Questa assegna il nuovo valore della variabile alla proprietà... del componente ed aggiorna il valore de

Identica alla procedura precedente, l'unica differenza sta nel fatto che aggiorna la proprietà... Num2 invece di Num1.
Mi sembra inutile far notare che questa riga assegna il valore della variabile Value alla variabile interna N1 (Proprietà... Num1).
Questa è la riga che aggiorna automaticamente la variabile R (Proprietà... Risultato).
Semplice vero?
Vediamo adesso l'altro metodo per conseguire l'obiettivo che ci siamo proposti.

Ridefiniamo il nome dell'oggetto, il tipo e le tre solite variabili interne.

La definizione delle proprietà... Num1 e Num2 resta identica a quella della Lezione 1a.

Sheet1

La sintassi è stata modificata, in particolare allo specificatore Read non è stata assegnata una variabile, ma il nome della funzione.

Come per le Procedure in scrittura, questa Funzione viene automaticamente chiamata da Delphi ogni qual volta sia richiesto il calcolo.

Come per il componente precedente, definiamo il prototipo della funzione nella sezione Private del componente.

Ed ora scriviamo la funzione appena dichiarata.

La funzione ovviamente restituisce la somma delle proprietà Num1 e Num2.

È stato utilizzato Result, ma può essere tranquillamente sostituito dal nome della Funzione, in questo caso 'GetRisultato'.
I due Componenti sono pronti. Non scordiamoci Mai la procedura Register!

I parametri della procedura RegisterComponents sono stati modificati in modo da installare tutti e due i componenti.

Se i due componenti dovranno essere installati nella stessa sezione della ComponentPalette, è consigliabile utilizzare una sola procedura.
Possiamo finalmente provare i due componenti. Non vi diremo noi le differenze, provate voi a scoprirle:
Aggiungete una ShowMessage alle due Procedure e alla funzione per vedere quando e quante volte viene eseguito il calcolo.
Non pensate che sia finita qui...
Vedremo come creare componenti visibili, come creare nuovi eventi, come ereditare metodi e tanto altro ancora!