Come Usare le classi Toolbar e Toolbutton in dB2K

Ultima Modifica: 31 gennaio 2001
Ken Mayer, Senior SQA Engineer dBASE, Inc.

Files di esempio disponibili in tooluse.zip

Traduzione di Antonio Campagna – Dicembre 2002

NOTA: Questo documento fu scritto originariamente per Visual dBASE 7.0/7.01, è stato aggiornato per dB2K (release 1). Ciò che si tratterà in questo documento è destinato ad essere usato in dB2K, ma a meno che non si tratti di un aspetto specifico di dB2K, può essere usato per Visual dBASE 7.x.

Le classi Toolbar e Toolbutton che sono integrate in dB2K sono un po' un enigma per molti sviluppatori, a causa del loro modo di funzionamento. Comunque, molti sviluppatori desiderano usarle nelle loro applicazioni per renderle simili alle altre applicazioni Windows.
Questo documento è destinato ad aiutarvi ad imparare gli elementi di base delle barre strumenti. Si parlerà anche delle differenze che si riscontrano quando si usano schede MDI e SDI.
Per raggiungere gli obiettivi che si prefigge questo documento utilizzeremo gli esempi forniti con db2k. Creeremo un'unica barra degli strumenti che ha le funzionalità comuni necessarie alla maggior parte delle schede. Vedremo poi come usare questa barra con i diversi tipi di schede.

Che cosa è una Barra degli Strumenti?

Suppongo che la prima cosa da definire è capire cos'è una barra degli strumenti:

Una barra degli strumenti è un contenitore con proprietà speciali.

Gli unici controlli che le barre strumenti possono contenere sono i **toolButtons**.

- la barra strumenti può essere agganciata ad una scheda;
- la barra strumenti può essere una "tavolozza mobile";
- la barra strumenti ha un evento integrato che "sonda" continuamente il software, grazie al quale potete determinare lo stato dell'interfaccia (utile per determinare se un pulsante dovrebbe essere abilitato o disabilitato).

Un esempio di barra degli strumenti è ciò che vedete quando lavorate con db2k in cima all'IDE. Noterete pure che la barra "cambia" in funzione delle operazioni che state compiendo, ad esempio se siete nell'editor di sorgenti e non avete selezionato del testo, i pulsanti che fanno riferimento alla clipboard sono disabilitati.

La classe Toolbar in dB2K

Proprietà	Impostazione predefinita	Descrizione
className	TOOLBAR	Identifica la classe ToolBar.
flat	true	Valore logico che commuta l'aspetto dei pulsanti sulla barra degli strumenti da sempre sollevato (false) a sollevato solo quando il puntatore è su un pulsante (true), e viceversa.
Floating	false	Valore logico che permette di specificare la barra degli strumenti come ridotta (false) o mobile (true).
form	null	Restituisce il riferimento oggetto della scheda alla quale la barra di strumenti è collegata.
hWnd	0	Restituisce l'handle della barra degli strumenti.
imageHeight	0	Regola l'altezza predefinita per tutti i pulsanti sulla barra degli strumenti. Poiché tutti i pulsanti devono avere la stessa altezza, se ImageHeight è impostata a 0, tutti i pulsanti avranno la stessa altezza del pulsante più alto. Se ImageHeight è impostata ad un numero positivo diverso da zero, le immagini assegnate ai pulsanti sono espanse (aggiungendo la struttura del pulsante) o troncate (rimuovendo i pixel dal centro dell'immagine o rimuovendo il suo margine).
imageWidth	0	Specifica la larghezza, in pixel, per tutti i pulsanti sulla barra degli strumenti.
left	0	Specifica la distanza dal lato sinistro dello schermo al margine di una barra di strumenti mobile.
text		La stringa visualizzata nella barra del titolo di una barra di strumenti mobile.
top	0	Specifica la distanza tra la sommità dello schermo e la parte superiore di una barra di strumenti mobile.
visible	true	Proprietà logica che permette di nascondere o rivelare la barra degli strumenti.
Evento	**Evento**	**Descrizione**
onUpdate		Si verifica ripetutamente mentre l'applicazione è in attesa per aggiornare lo stato dei pulsanti della barra strumenti.
Metodo	**Parametri**	**Descrizione**
attach()	<scheda>	Stabilisce un collegamento fra la barra strumenti e la scheda.
detach()	<scheda>	Interrompe il collegamento fra la barra strumenti e la scheda.

Come si realizza una Barra degli Strumenti?

Sfortunatamente, a suo tempo gli ideatori di Visual dBASE 7 non ci diedero un modulo di progettazione visuale per le barre strumenti e la squadra di R&D a dBASE, Inc. non ha avuto ancora il tempo per crearne uno. Ciò renderebbe sicuramente le cose più facili. Comunque, data la natura delle barre degli strumenti ed il fatto che la loro visualizzazione è quasi completamente automatica, non avere un modulo di progettazione visuale non è un inconveniente serio. Se una barra degli strumenti è agganciata ad una scheda (per default), potete stabilire il "bitmap" che apparirà sui pulsanti, in che ordine appariranno e se sono abilitati o disabilitati. In più se la barra degli strumenti è una tavolozza mobile, si può assegnare il titolo e la posizione della tavolozza.

Quindi l'unico modo di realizzare una barra degli strumenti è tramite il codice.

Per creare una barra degli strumenti dovete definirla come una sottoclasse della classe toolbar:

```
class myToolbar of toolbar

    // qui vanno inserite le proprietà

    // qui vanno inserite le definizioni dei pulsanti ( toolbuttons)

    //il codice  (toolbutton onClick, toolbar onUpdate ...)

    // va inserito qui

endclass
```

Nella directory "samples" di dB2K vi è un programma chiamato: CLIPBAR.PRG.
Si potrebbe usarlo "tale e quale", ma il codice è un po' complicato, così cercheremo di semplificarlo un po'. Il gruppo che ha sviluppato gli esempi a Borland (ora Inprise) tentò di rendere questi programmi simili a classi toolbar "generiche", in altre parole utilizzabili per la maggior parte delle applicazioni. Di conseguenza, quando si tenta di capire come funzionano le barre strumenti, ci sono degli inconvenienti, in quanto è stato aggiunto molto codice per soddisfare le varie contingenze che potrebbero non essere necessarie per le vostre barre degli strumenti. Quindi esamineremo versioni semplificate di queste barre strumenti.

Inserire dei Pulsanti (Toolbuttons) nella Barra degli Strumenti

I "toolbuttons" sono i pulsanti contenuti nella barra degli strumenti e fanno riferimento alla classe pushbutton di dB2K, ma non sono vere sottoclassi di questa classe. **L'unico posto dove potete usare un "toolbutton" è su una barra degli strumenti.** In altre parole non potete mettere direttamente un oggetto toolbutton in una scheda.

La classe Toolbuttons ha le seguenti proprietà ed eventi:

Proprietà	Impostazione predefinita	Descrizione
bitmap		File grafico (qualsiasi formato supportato) o riferimento di risorsa che contiene una o più immagini da visualizzare sul pulsante.
bitmapOffset	0	Specifica la distanza, in pixel, dalla sinistra del Bitmap specificato al punto in cui inizia l'immagine del pulsante. Questa proprietà è necessaria solo quando si specifica un Bitmap che contiene una serie di immagini disposte da sinistra a destra. Deve essere usata con BitmapWidth per specificare quanti pixel visualizzare dal Bitmap multimmagine. L'impostazione predefinita è 0 (primo elemento di unBitmap multimmagine).
bitmapWidth	0	Specifica il numero di pixel dal Bitmap specificato che si desidera visualizzare sul pulsante. Questa proprietà è necessaria solo quando si specifica un Bitmap che contiene una serie di immagini disposte da sinistra a destra. Deve essere usata con BitmapOffset, che specifica il punto iniziale dell'immagine da visualizzare.
checked	false	Restituisce true se il pulsante ha la sua proprietà TwoState impostata a true. Altrimenti restituisce false.
className	TOOLBUTTON	Identifica la classe ToolButton.
enabled	true	Valore logico che determina se il pulsante risponde o no quando si fa sopra clic. Quando è impostato a false, il sistema operativo cerca di modificare in modo visuale il pulsante con una versione tratteggiata o a basso contrasto del bitmap per indicare che il pulsante è non disponibile.
parent	N/A	Un riferimento oggetto che punta alla barra degli strumenti genitore.
separator	false	Valore logico che permette di impostare una riga verticale sulla barra degli strumenti per raggruppare visualmente i pulsanti. Se viene specificato un pulsante separatore, solo la sua proprietà Visible è significativa.
speedTip		Specifica il testo che appare quando il mouse rimane su un pulsante per più di un secondo.
twoState	true	Valore logico che determina se il pulsante appare in modo diverso quando è stato premuto e imposta conseguentemente la sua proprietà Checked a true.
visible	false	Valore logico che permette di nascondere (false) o mostrare (true) il pulsante.
Evento	**Parametri**	**Descrizione**
onClick		Dopo che si è fato clic su un pulsante.

Probabilmente un semplice toolbutton è quello che ci permette di navigare nella tabella, ad esempio un pulsante "primo" che ci sposti all'inizio della tabella. Per i nostri scopi, create MYTOOLBAR.CC (MODI COMM MYTOOLBAR.CC nella finestra comandi), ed immette i seguente codice:

```
class myToolbar of toolbar
    // qui inserite le proprietà
    this.flat = true
    this.floating = false

      // qui inserite la definizione dei pulsanti (toolbuttons)
    this.FirstToolButton = new ToolButton( this )
    with ( this.FirstToolButton )
        bitmap   := "RESOURCE TS_FIRST"
        speedTip := "Prima Riga"
        onClick  := class::First_onClick
    endwith

      // codice per gli eventi onclick
    function First_onClick
    return ( this.parent.form.rowset.first() )

  endclass
```

Il codice nella funzione "First_onClick" naviga semplicemente nel rowset che è agganciato alla scheda. Notate la sintassi:

this.parent.form.rowset.first ()

- **"this"** si riferisce al toolbutton che ha chiamato il codice;
- **"parent"** si riferisce alla barra degli strumenti che è il contenitore del toolbutton;
- **"form"** è la scheda alla quale la barra degli strumenti è agganciata;

Relativamente a quest'ultimo frammento di codice ci sono dei possibili problemi:

1. se questa barra degli strumenti è agganciata ad una scheda in esecuzione che non ha un rowset assegnato;
2. non avete bisogno di chiamare il metodo first() se già vi trovate sulla prima riga della tabella.

Usando l'evento "onUpdate" della barra degli strumenti possiamo trattare abbastanza agevolmente questo genere di problemi.

Abbiamo bisogno di aggiungere un nuovo metodo alla classe toolBar:

```
function onUpdate
    // controlla se c'è un rowset sulla scheda
    if type( "this.form.rowset" # "O" )
      this.FirstToolbutton.enabled = false
    else
        // controlla se già siamo sulla prima riga

      this.FirstToolbutton.enabled := ;
         NOT this.form.rowset.atFirst()
    endif
  return
```

L'evento onUpdate si verifica ripetutamente finché non c'è interazione con la barra degli strumenti. Se il mouse è sulla barra o l'utente sta cliccando sui pulsanti, allora l'evento non si verificherà, altrimenti il codice mostrato sopra sarà eseguito piuttosto spesso.

Rendere la barra degli strumenti più utile

La nostra barra degli strumenti di esempio non è al momento molto utile, perché contiene solamente un pulsante.
Adesso inseriremo i cosiddetti pulsanti di navigazione, cioè quelli che ci permettono di "scorrere" i dati.
Questi pulsanti sono: Precedente, Successivo e Ultimo. **Notate che i pulsanti nella barra degli strumenti appariranno nell'ordine in cui sono stati definiti nel codice sorgente.**

bitmaps

I bitmaps che sono usati in questi esempi hanno un nome di risorsa tipo "TS_xxxx". Le prime due lettere sono l'acronimo di "**T**oolbar **S**mall", il carattere di sottolineatura è un separatore e la parte "xxxx" è il nome descrittivo dell' immagine.
Ci sono anche i bitmaps di tipo "TL_xxxx" che rappresentano le stesse immagini un po' più grandi, infatti "**L**" sta per "large" e **T** significa sempre toolbar.
Per vedere quello che contiene il file di risorse standard RESOURCE.DLL andate nelle finestra comandi e digitate il seguente codice:

```
t = new ToolBar()
t1 = new ToolButton( t )
inspect( t1 )
```

Nel modulo ispezione nella categoria "varie" cliccate sulla voce bitmap e successivamente cliccate sul pulsante "strumento". Apparirà una finestra di dialogo. Nel combobox "ubicazione" scegliete "Risorsa" (per default sarà nomefile), e sul lato destro della finestra di dialogo cliccate sul pulsante "strumento". Apparirà la finestra di dialogo "Choose Resource" che vi permetterà di trovare l'immagine che vi serve. Si noti che per il nome delle immagini, per alcune si usano delle stringhe testo per altre i numeri.

Se l'immagine che volete usare ha il nome che è una stringa di testo, la stringa della proprietà bitmap conterrà:

"RESOURCE bitmapname"

Se l'immagine che volete usare ha il nome che è un numero la stringa della proprietà bitmap sarà:

"RESOURCE #number"

Il segno "#" è vitale. infine se la risorsa che volete usare ha due immagini (ad esempio il primo insieme di immagini sono chiamate con un nome simile a "PS_qualcosa") avete un bitmap cosiddetto "SPLIT" e la stringa nella proprietà bitmap dovrà essere modificata per esempio nel modo seguente:

"RESOURCE:2 PS_FIRST"

Ovviamente potete utilizzare le vostre immagini personali purché siano delle dimensioni giuste.
In Visual DBASE 7.x vi era una directory con un insieme di immagini per i pulsanti

.. \Visual dBASE\ARTWORK\BUTTON

ma questa cartella è stata rimossa in db2k. Queste immagini sono tutte di tipo Split. Per vederle è un po' più difficile, a meno che non creiate una scheda appositamente. Sempre nella proprietà bitmap si dovrebbe leggere:

"FILENAME:2 path\filename.bmp"

Disabilitare Bitmaps

Una differenza tra i pulsanti standard e i pulsanti di una barra degli strumenti (toolbuttons) è che quest'ultimi hanno solamente una proprietà chiamata "bitmap".

I pulsanti della barra degli strumenti non hanno proprietà separate come "upBitmap", "focusBitmap", "downBitmap" e "disabledBitmap". Ci sono diverse ragioni per questo motivo. Primo, i Toolbuttons non ottengono mai il "fuoco", così non hanno bisogno di una proprietà focusBitmap. Secondo, VdB gestisce automaticamente "l'oscuramento" del bitmap dei toolbuttons disabilitati, così non c'è bisogno di una proprietà disabledBitmap. Il downBitmap separato non è incluso.
Ora che abbiamo sollevato l'argomento di "oscurare" il bitmap di un toolbutton, consideriamo quello che accade al bitmap quando disabilitate un toolbutton. VdB interpreta automaticamente i colori del bitmap per determinarne una rappresentazione "offuscata". Nel fare così, un colore (e solamente uno) è visto come "trasparente". Con un bitmap di 16 colori, questo colore è rgb(128,0,128) cioè viola. I files di tipo GIF anche usando con un colore trasparente non avranno questa limitazione. GIF con i colori trasparenti sembrano essere ignorate. Se il vostro bitmap non contiene il colore trasparente come sfondo, il colore di sfondo sarà interpretato come leggero o scuro e sarà reso di conseguenza. Questo può, o non può offrire i risultati desiderati. In ogni caso, l'utente ha controllo dello schema dei colori di Windows e può cambiarli a suo piacimento. Questo vuol dire che i vostri bitmap non saranno visualizzati correttamente se l'utente cambia i colori. I bitmap o visualizzeranno lo sfondo quando non "oscurate", o non saranno oscurate propriamente o entrambi i casi. Quindi, per ottenere migliori risultati per i vostri pulsanti, si raccomanda di usare bitmap di 16 colori con lo sfondo impostato sul colore viola (Red=128, Green=0 Blue=128).

Aggiungete il seguente codice in MyToolbar.cc:

```
this.PreviousToolButton = new ToolButton( this )
  with ( this.PreviousToolButton )
      bitmap   := "RESOURCE TS_PREV"
      speedTip := "Riga Precedente"
      onClick  := class::Previous_onClick
  endwith

  this.NextToolButton = new ToolButton( this )
  with ( this.NextToolButton )
      bitmap   := "RESOURCE TS_NEXT"
      speedTip := "Riga Successiva"
      onClick  := class::Next_onClick
  endwith

  this.LastToolButton = new ToolButton( this )
  with ( this.LastToolButton )
      bitmap   := "RESOURCE TS_LAST"
      speedTip := "Ultima Riga"
      onClick  := class::Last_onClick
  endwith
```

Ed aggiungete gli eventi onClick seguenti:

```
 function Previous_onClick
   local bNext
   // va indietro nel rowset di una riga
   bNext = this.parent.form.rowset.next(-1)
   // Se siamo alla fine del set dobbiamo spostarci
   // in avanti
   if ( not bNext )
      this.parent.form.rowset.next()
    endif
  return

  function Next_onClick
    local bNext
    bNext = this.parent.form.rowset.next()
    if ( not bNext )
       this.parent.form.rowset.next(-1)
    endif
  return ( bNext )
```

```
function Last_onClick
return ( this.parent.form.rowset.last() )
```

Ed infine abbiamo bisogno di cambiare l'evento l'onUpdate della barra degli strumenti per includere tutti i pulsanti:

```
function onUpdate
    // controlla se c'è un rowset nella scheda:
    local bRowset, bAtFirst, bAtLast
    bRowset = type( "this.form.rowset" ) == "O"
    if bRowset
      bAtFirst = this.form.rowset.atFirst()
      bAtLast  = this.form.rowset.atLast()
      this.FirstToolbutton.enabled    = NOT bAtFirst
      this.PreviousToolbutton.enabled = NOT bAtFirst
      this.NextToolbutton.enabled     = NOT bAtLast
      this.LastToolbutton.enabled     = NOT bAtLast
    else // se non c'è  rowset ...
      this.FirstToolbutton.enabled    = false
      this.PreviousToolbutton.enabled = false
      this.NextToolbutton.enabled     = false
      this.LastToolbutton.enabled     = false
    endif
  return
```

Salvate il codice ed uscite dall'editor di sorgenti.

Proviamo la nostra barra degli strumenti.

Per poter provare la nostra barra degli strumenti abbiamo bisogno di una scheda agganciata con i dati cioè con un rowset. Potete usare una scheda esistente o crearne rapidamente una nuova con l'apposito wizard. Ci sono due modalità di apertura di una scheda (open() e readmodal()), analizzeremo il funzionamento della barra in entrambe modalità. Digitate il seguente codice nella finestra comandi, sostituendo a "formname" il nome della scheda di prova. Il primo test sarà effettuato con una scheda non modale:

```
set procedure to formname.wfm additive
set procedure to MyToolbar.cc additive
f = new formnameform()
t = new MyToolbar()
t.attach( f )
f.open()
```

Notate che quando la scheda è aperta, la barra degli strumenti appare in cima allo schermo (sopra la 'normale' barra degli strumenti) e se siamo all'inizio del rowset, i primi due pulsanti della barra non sono abilitati. Comunque, cliccando sui pulsanti navigherete sui dati come ci si aspetta.
Una volta che avete finito di sperimentare, chiudete la scheda e notate che la nuova barra è sparita.

Ora, proviamo l'apertura della stessa scheda come sdi e vedremo alcune differenze:

```
t.detach( f )     // Sganciate la barra dalla scheda
t=null           //  annullate il riferimento oggetto
f.mdi = false
t=new mytoolbar() // Ri-istanziate la barra
t.attach( f )     // Ri-agganciate la barra ...
f.open()
```

Notate che la barra adesso è sulla scheda e che gli oggetti sulla scheda si sono spostati tutti in giù per fare spazio alla barra.
Cosa accade invece su una scheda completamente modale? (Chiudete la scheda e digitate il seguente:)

```
f.readModal()
```

L'evento onUpdate non sembra si verifichi su una scheda modale. Questo fatto può essere così come progettato (la barra sta aspettando che si chiudi la scheda per proseguire...), o può essere un bug (ho sottoposto questo problema a dBASE, Inc. come se fosse un bug, vedremo cosa ci diranno). È possibile realizzare un "work around", ma non i questo documento.

NOTA:
Ci sono alcuni interessanti problemi "comportamentali" assegnando specifici valori in stringhe specifiche.
Gary White ci ha pensato su è ha realizzato il seguente codice da utilizzare quando si cambia la proprietà MDI delle schede:

```
f=new tbartestform()
f.mdi = true
f.t = new mytoolbar()
f.t.attach(f)
f.open()
```

Questo codice funziona eccellentemente. Notate che la proprietà mdi è impostata prima di agganciare la barra. In questo caso, la barra dovrebbe apparire nella _app.framewin (Cornice dell'Applicazione).

```
f=new tbartestform()
f.mdi=true // per scopi di test
f.t = new mytoolbar()
f.t.attach(f)
f.mdi = false // <-- Problema
f.open()
```

Nell'esempio mostrato su, se impostate la proprietà MDI a falso dopo avere agganciato la barra essa apparirà agganciata a _app.framewin, piuttosto che alla scheda, come vi aspettereste per una scheda SDI.
Ora, prima che esultiate pensando che possiate usarla tranquillamente date uno sguardo all'ultimo esempio.

```
f=new tbartestform()
f.mdi=false
f.t = new mytoolbar()
f.t.attach(f)
f.open()
```

Questo funziona come vi aspettate, perché la proprietà MDI è messa a falso prima di agganciare la barra. In questo caso, la barra sarà agganciata alla scheda come in effetti dovrebbe essere.
Infine, se la scheda è istanziata con la proprietà MDI impostata su falso, poi agganciate una barra degli strumenti e successivamente cambiate la proprietà MDI a vero, Si verificherà GPF.

```
f=new tbartestform()
f.mdi=false // for testing
f.t = new mytoolbar()
f.t.attach(f)
f.mdi=true  // questo causerà un gpf, ma non apparirà
        // fino a quando non aprirete la scheda
f.open()
```

Una cosa importante da ricordare è impostare la proprietà mdi prima di agganciare la barra. Se avete bisogno di cambiare la proprietà mdi dopo avere agganciato una barra, avrete bisogno di sganciare la barra ed annullare il riferimento oggetto e successivamente ricreare la barra per evitare problemi. Tenendo conto dell'esempio precedente dove f è la scheda e t è la barra e la proprietà form.mdi=false:

```
f.t.detach(f)      // sganciate la barra
f.t=null          // annullate il riferimento oggetto
f.mdi=true        // adesso cambiate la proprietà mdi
f.t=new mytoolbar() // ricreate la barra
f.t.attach(f)      // e ri-agganciatela
```

Aggiungere Funzionalità alla Barra

Possiamo aggiungere altre funzionalità in due modi:

1. aggiungere più barre alla scheda;
2. mettere un gruppo di pulsanti in una barra.

Lo svantaggio di aggiungere barre multiple è che esse si accatastano semplicemente una sotto l'altra. Questo potrebbe non essere desiderabile (non lo è per nessuna mia applicazione ma...).
Il vantaggio è che l'utente può "tirare" via ciascuna barra dalla scheda o dall'applicazione come una tavolozza mobile, o spostarla dalla posizione in cui si trova. Questo vuole dire che le barre possono essere indipendenti l'una dall'altra. Un altro vantaggio è più facile riutilizzare il codice. Tutte le vostre barre non avranno bisogno di tutti i pulsanti. Si possono raggruppare in barre "logiche" più piccole e aggiungere semplicemente la funzionalità necessaria ad una barra esistente.
Per i nostri scopi, aggiungeremo semplicemente alla barra.
Chiaramente, una cosa probabile da tenere in considerazione è quella di mettere un separatore tra le

sezioni della barra. L'interruzione ci aiuta a capire lo scopo delle diverse parti di una barra. Quindi la prima cosa che dovremmo fare è aggiungere un nuovo pulsante alla nostra classe di toolbar:

```
this.Separator1ToolButton = new ToolButton( this )
with ( this.Separator1ToolButton )
    separator = true
endwith
```

Notate che **separator** è l'unica proprietà della quale abbiamo bisogno.
La prossima cose che potremmo probabilmente aggiungere sono i pulsanti che fanno riferimento alla clipboard (taglia, incolla..). In dB2K i controlli visuali che hanno la capacità di usare la Clipboard di Windows hanno metodi incorporati che si occupano di gestire queste funzionalità, così non dobbiamo scrivere alcun codice. Dobbiamo creare i pulsanti ed agganciare il codice negli eventi onClick e nell'evento onUpdate della barra.

Prima aggiungiamo i pulsanti seguenti (dopo il separatore che abbiamo inserito prima):

```
this.CutToolButton = new ToolButton( this )
with ( this.CutToolButton )
    bitmap   := "RESOURCE TS_CUT"
    speedTip := "Taglia"
    onClick  := class::Cut_onClick
endwith

this.CopyToolButton = new ToolButton( this )
with ( this.CopyToolButton )
    bitmap   := "RESOURCE TS_COPY"
    speedTip := "Copia"
    onClick  := class::Copy_onClick
endwith

this.PasteToolButton = new ToolButton( this )
with ( this.PasteToolButton )
    bitmap   := "RESOURCE TS_PASTE"
    speedTip := "Incolla"
    onClick  := class::Paste_onClick
endwith
```

Negli eventi onClick di questi pulsanti dobbiamo inserire:

```
function Cut_onClick
return ( this.parent.form.activeControl.cut() )

function Copy_onClick
return ( this.parent.form.activeControl.copy() )

function Paste_onClick
return ( this.parent.form.activeControl.paste() )
```

Infine l'evento onUpdate

```
this.LastToolbutton.enabled     = NOT bAtLast
    // New Code:
    bClip = ( TYPE("this.form.activeControl.copy") == "FP" )
    this.CutToolButton.enabled      = bClip
    this.CopyToolButton.enabled     = bClip
    this.PasteToolButton.enabled    = bClip
```

Potete aggiungere anche il codice che controlla se non c'è rowset sulla scheda per disabilitare questi pulsanti...

Un altro insieme di pulsanti che aggiungeremo alla nostra barra sono quelli che gestiscono le operazioni di modifica del rowset, che ci permetteranno di modificare la riga corrente, aggiungere una nuova riga, cancellare una riga, salvare le modifiche, abbandonare le modifiche...
Sto usando il programma EDITBAR come base per costruire i nostri pulsanti di editing e rispetto al programma originario non ci sarà il pulsante EDIT. Questo pulsante è utile se avete la proprietà autoEdit del vostro rowset impostata su falso. Tutte le mie applicazioni funzionano così (altrimenti non c'è modo per l'utente per modificare una riga!).

```
this.Separator2ToolButton = new ToolButton( this )
with ( this.Separator2ToolButton )
    separator = true
endwith

this.EditToolbutton = new ToolButton( this )
with ( this.EditToolButton )
    bitmap   := "RESOURCE TS_EDIT"
    speedTip := "Modifica Riga"
    onClick  := class::Edit_onClick
endwith

this.AppendToolbutton = new ToolButton( this )
with ( this.AppendToolButton )
    bitmap   := "RESOURCE TS_APPEND"
    speedTip := "Aggiungi Riga"
    onClick  := class::Append_onClick
endwith

this.DeleteToolButton = new ToolButton( this )
with ( this.DeleteToolButton )
    bitmap   := "RESOURCE TS_DELETE"
    speedTip := "Cancella Riga"
    onClick  := class::Delete_onClick
endwith
```

```
this.SaveToolButton = new ToolButton( this )
with ( this.SaveToolButton )
    bitmap   := "RESOURCE TS_SAVE"
    speedTip := "Salva Riga"
    onClick  := class::Save_onClick
endwith

this.AbandonToolButton = new ToolButton( this )
with ( this.AbandonToolButton )
    bitmap   := "RESOURCE TS_ABANDON"
    speedTip := "Abbandona Modifiche"
    onClick  := class::Abandon_onClick
endwith
```

Ed il codice dell' evento onClick seguente:

```
function Edit_onClick
 return ( this.parent.form.rowset.beginEdit() )

 function Append_onClick
 return ( this.parent.form.rowset.beginAppend() )

 function Delete_onClick

  local bDelete, bFirst
   bDelete = false
   bFirst  = false
   if ( not this.parent.form.rowset.endOfSet )
     if ( MSGBOX("Siete sicuri di voler cancellare la riga corrente." ;
             + CHR(13) ;
             + "Cliccate su pulsante Si per cancellare.", ;
               "Attenzione", ;
               4) == 6 )
       bFirst  := this.parent.form.rowset.atFirst()
       bDelete := this.parent.form.rowset.delete()
       if ( not bFirst )
         this.parent.form.rowset.next(-1)
       endif
       if ( this.parent.form.rowset.endOfSet )
         this.parent.form.rowset.next()
       endif
     endif
   endif
 return ( bDelete )
```

```
function Save_onClick
return ( this.parent.form.rowset.save() )

function Abandon_onClick
return ( this.parent.form.rowset.abandon() )
```

E questo per l'evento onUpdate della barra

```
function MyToolBar_onUpdate
  // controlla se la scheda è dotata di rowset:
  local bRowset, bAtFirst, bAtLast
  bRowset = type( "this.form.rowset" ) == "O"
  if bRowset
    bAtFirst = this.form.rowset.atFirst()
    bAtLast  = this.form.rowset.atLast()
    this.FirstToolbutton.enabled    := NOT bAtFirst
    this.PreviousToolbutton.enabled := NOT bAtFirst
    this.NextToolbutton.enabled     := NOT bAtLast
    this.LastToolbutton.enabled     := NOT bAtLast
    bClip = ( TYPE("this.form.activeControl.copy") == "FP" )
    this.CutToolButton.enabled      := bClip
    this.CopyToolButton.enabled     := bClip
    this.PasteToolButton.enabled    := bClip
    bEoF = this.form.rowset.endOfSet
    this.EditToolButton.enabled     := NOT bEof
    this.AppendToolButton.enabled   := true
    this.DeleteToolButton.enabled   := NOT bEoF
    bModified = this.form.rowset.modified
    this.SaveToolButton.enabled     := bModified
    this.AbandonToolButton.enabled  := bModified

  else // la scheda non ha rowset…
    this.FirstToolbutton.enabled    := false
    this.PreviousToolbutton.enabled := false
    this.NextToolbutton.enabled     := false
    this.LastToolbutton.enabled     := false
    this.CutToolbutton.enabled      := false
    this.CopyToolbutton.enabled     := false
    this.PasteToolbutton.enabled    := false
    this.EditToolButton.enabled     := false
    this.AppendToolButton.enabled   := false
    this.DeleteToolButton.enabled   := false
    this.SaveToolButton.enabled     := false
    this.AbandonToolButton.enabled  := false
  endif
return
```

Notate che c'è una copia di MyToolbar.CC fornita con questo documento e può essere usata tale e quale nelle vostre applicazioni

Cose da Notare

dB2K Toolbar

Se non avete mai fatto prima questo e state lavorando con una applicazione MDI, potreste essere un pò disturbati nel notare che la vostra barra appare sopra quella di dB2K. Per la vostra applicazione potrebbe non essere esteticamente piacevole.
Non disperate, c'è una semplice soluzione, l'applicazione è un oggetto, con il nome di "_app" ed ha delle proprietà. Una di queste proprietà è: "speedbar" che può assumere un valore logico (vero/falso). Per default è impostata su vero. Il comando è "spegnere" la VdBASE speedbar,:

```
_app.speedbar := false
```

Non dimenticate di "accenderla" nuovamente quando la vostra applicazione ha terminato l'esecuzione.

Sganciare le Barre

Se, quando chiudete la vostra scheda non sganciate la vostra barra, potete riscontrare alcuni interessanti risultati. La barra sarà replicata (ne avrete due o piu..). La soluzione è abbastanza semplice, quando chiudete la vostra scheda, nell'evento canClose o nell'evento onClose chiamate il metodo di sganciamento della barra:

```
t.detach( this )
```

Il problema così facendo è che comunque il riferimento "t" è ancora disponibile. Dovreste considerare forse l'opportunità di creare un riferimento alla barra della vostra scheda, quando l'istanziate:

```
f = new form()
f.toolbar = new MyToolBar()
f.toolbar.attach( f )
// etc.
```

Questo vi permetterà di liberarsi abbastanza facilmente della barra, ed ancora meglio, se rilasciate la scheda, il riferimento alla barra è andato completamente.

In più, avete un modo facile di accedere alla barra durante l'esecuzione della scheda, nel caso che per qualsiasi ragione desiderate cambiare la barra, il riferimento alla barra è la e potete cambiare le proprietà di un singolo pulsante della barra aggiungendo solo il loro riferimento a "f.toolbar."

Condividere Barre degli Strumenti in una applicazione MDI

Un'altra cosa che potete desiderare considerare è avere schede multiple (in una applicazione MDI) condividendo la stessa barra.

Ciò risulta essere abbastanza semplice. Piuttosto che istanziare la barra come un oggetto/proprietà di una specifica scheda, potete agganciarla alla struttura dell'applicazione:

```
_app.ToolBar = new MyToolBar()
```

E quando volete agganciarla ad una scheda che state per aprire:

```
fMyForm1 = new MyFormForm()
_app.ToolBar.attach( fMyForm1 )
fMyForm1.open()
```

Se volete condividere questa barra con un'altra scheda dovete fare:

```
fMyForm2 = new MyFormForm2() // whatever ...
_app.ToolBar.attach( fMyForm2 )
fMyForm1.Open()
```

Una cosa da ricordare quando chiudete la scheda è che dovete sganciare la barra, o potreste incorrere in diversi problemi (come suddetto in questo documento, con lo stessa barra che appare diverse volte...).

```
_app.ToolBar.detach( fMyForm2 )
```

L'utente che consigliò questo (Todd Kreuter, sul dBASE newsgroups) suggerì anche che se volete usare barre multiple, create un array che contenga i riferimenti oggetto delle barre e poi agganciatele alle singole schede che ne hanno bisogno:

```
_app.Toolbars = new Array()
_app.ToolBars[1] = new ToolBarClip()
_app.ToolBars[2] = new ToolBarEdit()
// etc.
fMyForm1 = new MyFormForm()
_app.ToolBars[1].attach( fMyForm1 )
_app.ToolBars[2].attach( fMyForm1 )
// etc.
// Do whatever you need to, and when the
// form closes:
_app.ToolBars[1].detach( fMyForm1 )
_app.ToolBars[2].detach( fMyForm1 )
// etc.
```

Se volete che la barra compaia sempre sulla finestra della cornice applicazione, potreste agganciarla a framewin, piuttosto che ad una scheda:

```
_app.ToolBars[1].attach( _app.FrameWin )
```

Il problema è che così facendo il vostro codice sarà necessariamente più complesso. Provate ad agganciare una barra al framewin ed ad una scheda (sarà generato un errore che vi informa che non è

possibile farlo).

Come già detto il vostro codice in un caso simile a quello sopra accennato dovrebbe essere cambiato. Il codice agganciato alla barra e i pulsanti della barra non avranno idea di quale scheda usare. Questo significa che dovreste cambiare il codice per cercare un riferimento oggetto alla scheda "corrente", usando una proprietà custom agganciata alla barra, come "currentForm":

```
_app.ToolBars[1].CurrentForm = fMyForm1
```

Questo significa che avrete bisogno di impostare la proprietà currentForm ogni volta che aprite o chiudete una scheda, usando l'onGotFocus della scheda e quando la scheda perde il fuoco (incluso quando la scheda è chiusa) avrete bisogno di impostare la proprietà a NULL.

Non dimenticate quando avete finito di sganciare le barre e così via...

Come potete vedere, questa situazione rende il problema più complesso... ma è fattibile. (Ci sono degli esempi di questa situazione nei due file allegati: MYTOOLBAR2.CC ed APP.PRG)

SDI Forms

Come già detto in questo documento, la vostra barra apparirà in cima alla scheda, spostando in giù tutti gli oggetti della scheda per fare spazio alla barra stessa.

Ciò è fatto automaticamente, il problema è che quando gli oggetti sulla scheda sono spostati, non essendo la scheda stessa ridimensionata, se alcuni oggetti erano posizionati in fondo alla scheda non saranno più visibili.
Una soluzione è lasciare libero il fondo della scheda.

La domanda è come assicurarsi di lasciare abbastanza spazio?

La barra è visibile solo in esecuzione, così è difficile sapere precisamente quanto spazio riservare. Bisogna fare un po' di prove.. io suggerisco di creare un pulsante con la proprietà visibile impostata su falso, con il solo scopo di stare li e riservare spazio. Mettetelo in fondo alla scheda. Ridimensionatelo in modo tale da essere della stessa grandezza della barra e fate un pò di prove. Mandate in esecuzione la scheda con la barra agganciata e vedete se tutto funziona.
Impostate la proprietà text del pulsante con qualcosa che vi ricordi perché si trova là.

Un altro problema che ho notato è che le barre strumenti non hanno una linea di base quando sono inserite in una scheda. A me sembra che dia un senso di incompletezza. La soluzione è abbastanza semplice e consiste nell'usare un rettangolo e impostare le relative proprietà segue:

```
left = 0
width = form.width
top = 0
height = 0.1
text = ""
```

Lo stile predefinito del bordo è "inciso" che si abbinerà alla linea in alto della barra. Il rettangolo si sposterà verso il basso con tutti gli altri controlli quando la barra è agganciata ed apparirà sotto la barra. Chiaramente, se permettete alla scheda di essere ridimensionata, avrete bisogno di scrivere il codice che ridimensioni l'ampiezza del rettangolo nell'evento di onSize della scheda.

Schede modali

Se la vostra applicazione è di tipo SDI usate schede modali, c'è un inconveniente notevole come notato sopra,l'evento onUpdate non si attiva finché la scheda non viene chiusa il che non è certamente utile.

Una soluzione che comporta molto lavoro aggiuntivo consiste nel creare i vostri metodi agganciati ai metodi della barra che abilitano/disabilitano i pulsanti in funzione della situazione, basata sullo stato del rowset, o basata sui risultati di atFirst (), e così via. Quando la scheda si apre, potreste inserire del codice che controlla se siete sulla prima riga e conseguentemente disabilitare i pulsanti interessati, ecc.

Un'altra soluzione è creare la vostra barra usando la proprietà speedbutton di un pulsante (pushbutton) e forse un oggetto container. Questo vi permetterà di fare le medesime cose di una barra, ma non dovete contare sull'evento onUpdate del barra. (L'unico problema usando un pulsante è che loro non hanno l'opzione per avere una barra dall'aspetto "piatto" (flat).

Infine, l'ultima soluzione è quella di usare una barra degli strumenti custom (vedi nella duflp).

Barre Strumenti di tipo Tavolozze mobili contro Barre Strumenti agganciate alle schede

Un utente sul newsgroup ha fatto notare che anche con la proprietà floating della barra impostata su falso, è possibile rimuovere la barra dalla scheda e trasformarla in una tavolozza mobile.
Ciò è progettato così, credo sia un aspetto specifico di Windows, anche se non sono i grado di garantirlo

Se volete che la barra sia agganciata in modo permanente alla scheda potete utilizzare il codice seguente nell'evento onUpdate della barra:

```
if this.floating
  this.floating := false
endif
```

Ciò ci assicurerà che la proprietà floating sarà forzata a rimanere su falso.

Copyright 2001, Kenneth J. Mayer. Tutti i diritti riservati

Informazioni su dBASE, Inc. possono essere trovate al seguente indirizzo: http://www.dbase.com