TECH

Creazione e utilizzo dei file di risorse

Una delle novità del nuovo Visual Basic 4.0 è la possibilità di utilizzare i file RES: col presente articolo analizzeremo come vengono creati e utilizzati da Visual Basic

di Roberto Scano

Chiunque di voi crea applicazioni di tipo shareware spera in una loro larga diffusione sul territorio nazionale, tramite pubblicazione su riviste informatiche, masterizzazione su cd-rom con raccolte shareware/P.D. ecc. Se l'applicazione creata è poi di buona fattura, potrebbe esser possibile una sua diffusione all'estero inviandola, ad esempio, a qualche grosso sito FTP (come CICA, Simtel, CDROM.COM, ecc.). Mentre con le versioni precedenti di Visual Basic era necessario tradurre tutte le stringhe di testo e le caption dei vari controlli nella lingua straniera prescelta (dovendo così creare diverse sorgenti per ogni lingua utilizzata) con la versione 4.0 Microsoft ci offre finalmente la possibilità di sfruttare i file di risorse (RES) in modo che sia sufficiente richiamare la risorsa corrispondente alla lingua prescelta per una determinata stringa, caption, ecc. senza dover modificare il sorgente della nostra applicazione. Nei paragrafi seguenti analizzeremo come sia possibile creare questi file, come compilarli tramite il Resource Compiler e come utilizzarli dalle applicazioni Visual Basic 4.0. Le risorse possono essere suddivise in due gruppi:

Risorse di tipo stringa: contenenti delle stringhe di testo

Risorse di tipo binario: contenenti icone, immagini bitmap, cursori, effetti sonori WAV, filmati video AVI, ecc.

Per generare un file di risorse è sufficiente utilizzare un semplice text-editor per creare un file di testo con estensione RC che contiene le risorse di tipo stringa ed i riferimenti alle risorse di tipo binario da utilizzare con la nostra applicazione. Analizziamo ora le caratteristiche dei due tipi di risorse appena elencate.

GESTIONE DELLE RISORSE DI TIPO STRINGA

Questo tipo di risorse è contenuto in una tabella di stringhe nel file di definizione RC. La sintassi da utilizzare è la seguente:

```
STRINGTABLE [LOAD-OPTION] [MEM-OPTION]
    BEGIN
        STRING-ID STRING
        ........
    END
```

STRINGTABLE può definire una o più stringhe per una applicazione. Le risorse di tipo stringa non sono altro che stringhe di testo che possono essere utilizzate dall'applicazione Visual Basic, quando necessarie, utilizzando le funzioni descritte in seguito. I parametri utilizzati da STRINGTABLE sono i seguenti:

[LOAD-OPTION]: specifica il momento in cui le risorse devono esser caricate. Questo parametro è opzionale e può assumere uno dei seguenti valori: **preload** (le risorse vengono caricate immediatamente), **loadoncall** (le risorse vengono caricate quando richieste, come default).

[MEM-OPTION]: specifica in che modo avviene il caricamento delle risorse di tipo stringa in memoria. I valori possono essere i seguenti: **fixed** (le risorse rimangono in una locazione di memoria fissa), **moveable** (le risorse possono esser spostate da una locazione di memoria se è necessaria una compattazione della memoria), **discardable** (le risorse sono rimosse dalla memoria se non vengono più utilizzate).

Per quanto riguarda i paramentri identificativi delle stringhe, il valore **string-id** specifica un valore di tipo intero che identifica le stringa mentre il valore **string** specifica una o più stringhe ASCII, racchiuse da virgolette, Queste stringhe non possono essere più lunghe di 255 caratteri e devono esser scritte su un'unica riga nel file RC.

Se desideriamo, ad esempio, creare un file di risorse di tipo stringa da utilizzare in 5 lingue diverse, sarà necessario creare un gruppo di stringhe per ogni lingua.

GESTIONE DELLE RISORSE DI TIPO BINARIO

Le risorse di tipo binario, a differenza dal tipo stringa, non vengono memorizzate all'interno del file RC, il quale contiene invece dei riferimenti ai file contenenti le risorse di tipo binario che possono essere dei seguenti tipi:

- immagini bitmap (BMP)
- icone (ICO)
- cursori (CUR)
- altre risorse definite dall'utente

Per richiamare un qualsiasi file binario, è necessaria la seguente riga di comando nel file di definizione delle risorse:

```
NAMEID KEYWORD [LOAD-OPTION] [MEM-
    OPTION] FILENAME
```

Il parametro **nameid** può specificare sia un nome che un numero intero che contraddistinguerà la risorsa attuale dalle altre contenute nel file. Il valore specificato da **nameid** deve essere unico per qualsiasi categoria stabilita dal parametro successivo, **keyword**, il quale specifica il tipo di file da caricare. I parametri disponibili sono i seguenti:

BITMAP: definisce un riferimento ad un'immagine (BMP)
CURSOR: definisce un riferimento ad un cursore (CUR)
ICON: definisce un riferimento ad una icona (ICO)

Oltre a queste risorse ve ne sono altre, anche definibili dall'utente: riferimenti a file audio WAV, filmati AVI, dialog box, ecc. (vedi tabella 1). È da precisare che, per quanto riguarda la categoria ICON, è necessario utilizzare dei valori diversi da 1 (uno) per il parametro **nameid** in quanto il suddetto valore è riservato per l'icona dell'applicazione: se si utilizza questo valore, non viene generato alcun errore durante l'esecuzione all'interno dell'ambiente di sviluppo, ma solo al momento di creare l'eseguibile.

Per quanto riguarda i parametri **[load-option]** e **[mem-option]** valgono le stesse opzioni analizzate per le risorse di tipo stringa. Il parametro **filename**, invece, specifica il nome del file che contiene la risorsa di tipo binario: è necessario indicare il percorso completo dei file per non incorrere in errori durante la compilazione. Se ad esempio desideriamo assegnare ad un file bitmap di prova, che chiameremo PROVA.BMP, contenuto nella directory C:\WIN95 il valore 16, sarà necessario utilizzare la seguente riga nel file di definizione delle risorse:

```
16 BITMAP C:\WIN95\PROVA.BMP
```

Le risorse di tipo binario possono essere utilizzate da Visual Basic 4.0 tramite le funzioni *LoadResBitmap* (per icone, immagini e cursori) e *LoadResData* (per effetti sonori, filmati ed altri tipi di risorse definite dall'utente), che verranno descritte in uno dei prossimi paragrafi.

Per la creazione delle risorse, nella versione CD-ROM di Visual Basic 4.0 è presente il programma "Image Editor" che supporta immagini bitmap, cursori ed icone mentre per gli effetti sonori è possibile utilizzare il registratore di suoni di Windows: vi sono comunque buoni programmi shareware/P.D. che lavorano molto meglio (uno per tutti: Icon Master). Ora che abbiamo imparato come creare i file RC, analizziamo come vengono compilati tramite il Resource Compiler.

Figura 1

La schermata di apertura del programma ATM, fornito con il VB4 come esempio dell'uso dei file RES

IL RESOURCE COMPILER

Il Resource Compiler, contenuto nella directory \TOOLS\RESOURCE del CD-ROM di Visual Basic 4.0, è fornito con due versioni: una per la compilazione di risorse a 16 bit da utilizzare con applicazioni per Windows 3.1x (contenuta nella sottodirectory \RC16) ed una versione a 32 bit da utilizzare con applicazioni per Windows NT 3.5x e Windows95 (contenuta nella sottodirectory \RC32).

In entrambe le versioni, il Resource Compiler è un programma MS-DOS: speriamo che, com'è successo per l'Help Compiler, Microsoft distribuisca con la prossima versione di Visual Basic un Resource Compiler Workshop (in ogni caso spero che intanto escano delle applicazioni shareware che facciano ciò).

La riga di comando da utilizzare è la seguente:

```
RC /R [OPZIONI] FILE.RC
```

Il parametro **/R** specifica che il file di risorse RC deve esser compilato come file RES e non collegato ad un eseguibile. Per quanto riguarda il parametro **[OPZIONI]** vi è il valore /? per avere informazioni aggiuntive e il parametro **/FO FILENAME** che permette di compilare il file RC assegnandoli il nome indicato dal parametro FILENAME. Questo parametro è importante nel caso si voglia utilizzare un file RC per compilare delle risorse utilizzabili sia per applicazioni a 16 che a 32 bit. Se, ad esempio, abbiamo un file di definizione che chiameremo DEMO.RC contenuto nella directory C:\VB e vogliamo creare un file RES per applicazioni a 16 bit (chiamandolo, ad esempio, DEMO16.RES) sarà necessario posizionarsi nella directory del Resource Compiler a 16 bit e digitare la seguente riga di comando:

```
RC /R /FO DEMO16.RES C:\VB\DEMO.RES
```

Nel caso si voglia utilizzare lo stesso file di definizione per compilare un file RES per applicazioni a 32 bit (chiamandolo, ad esempio, DEMO32.RES) sarà necessario posizionarsi nella directory del Resource Compiler a 32 bit e digitare la seguente riga di comando:

```
RC /R /FO DEMO32.RES C:\VB\DEMO.RES
```

Ora che abbiamo il nostro file RES non ci resta altro che analizzare come utilizzarlo da un'applicazione Visual Basic.

UTILIZZO DELLE RISORSE IN UN PROGRAMMA VB

L'utilizzo delle risorse in un programma Visual Basic permette al programmatore di richiamare le risorse quando necessarie, anziché richiamarle ogni volta al caricamento di un form o di un modulo.

Come già accennato precedentemente, le risorse sono ottimali anche in caso si intenda distribuire software all'estero: sarà sufficiente creare il file RES contenente la lingua da utilizzare; e non sarà più necessario modificare il sorgente

della nostra applicazione. Per aggiungere un file di risorse ad un progetto Visual Basic è sufficiente scegliere la voce Add File dal menu File e scegliere come tipo di file Resource Files (*.RES). Visual Basic riconosce un file di risorse dall'estensione RES: se il file non possiede questa estensione, Visual Basic non lo carica nel progetto. Visual Basic permette di caricare nel progetto corrente soltanto un file di risorse: in caso si tenti di caricarne altri verrà generato un messaggio d'errore.

Se si seleziona il file RES nella finestra Project, i pulsanti View Form e View Code vengono disabilitati: purtroppo Microsoft ha lasciato le cose a metà, permettendo di utilizzare i file RES ma non di modificarli all'interno del progetto stesso (creatori di add-in, fatevi avanti!!!).

Per quanto riguarda le funzioni accennate nei paragrafi precedenti sull'utilizzo delle risorse in un programma Visual Basic, analizziamole ora una ad una nei parametri e utilizzando esempi pratici.

La funzione per l'utilizzo delle stringhe contenute in un file RES è **LoadResString (index)**. Il parametro *index*, obbligatorio, specifica il valore assegnato dal parametro string-id utilizzato nel file di definizione delle risorse. La funzione LoadResString restituisce la stringa di caratteri identificata dal parametro *string* utilizzato nel file di definizione delle risorse.

Utilizzare questo tipo di funzione per stringhe di caratteri di lunghezza consistente (inferiori a 255 caratteri) accorcia i tempi di caricamento in quanto le stringhe vengono caricate individualmente quando necessarie invece che tutte in una sola volta al caricamento di un form. Se, ad esempio, avevamo creato una risorsa come segue:

```
STRINGTABLE LOADONCALL DISCARDABLE
    BEGIN
        16      "Roberto Scano: 21
    anni, programmatore"
        24      "Leonia Sgnaolin: 22
    anni, studentessa universitaria"
    END
```

e desideriamo richiamare la stringa con il parametro index = 16, assegnando questo valore ad una label, sarà necessario utilizzare il seguente comando all'interno di un form:

```
label1.caption = LoadResString(16)
```

Naturalmente è possibile creare un modulo per assegnare delle costanti in riferimento ai valori numerici del parametro index:

```
Global Const resRoberto% = 16
Global Const resLeonia% = 24
```

e richiamarli con la funzione LoadResString (resRoberto) oppure LoadResString (resLeonia).

La funzione **LoadResPicture (index, format)** permette invece di utilizzare immagini bitmap, icone o cursori da un file RES. Il parametro index, obbligatorio, specifica il valore assegnato dal parametro name-id utilizzato nel file di definizione delle risorse per il quale il valore 1 (uno) è riservato all'icona dell'applicazione. Per quanto riguarda il parametro format, anch'esso obbligatorio, questo specifica un valore corrispondente al formato dei dati che devono essere estratti dal file RES.

Questo valore può essere 0 (zero) in caso di risorsa di tipo bitmap, 1 (uno) per risorsa di tipo icona e 2 (due) per risorse di tipo cursore. È possibile utilizzare, invece di questi valori delle costanti di Visual Basic: vbResBitmap per le risorse di tipo bitmap, vbResIcon per le risorse di tipo icona e vbRes Cursor per le risorse di tipo cursore.

È consigliabile utilizzare questa funzione per diminuire i tempi di caricamento dei form, in quanto le risorse, come già detto in precedenza, vengono caricate soltanto quando necessarie e non tutte in una volta al caricamento di un form. Se, ad esempio, abbiamo il seguente file di definizione di risorse:

```
16 BITMAP C:\WIN95\PROVA.BMP
24 ICON C:\WIN95\PROVA.ICO
```

e desideriamo utilizzare la risorsa di tipo bitmap per un controllo picture e la risorsa di tipo icon per l'icona del form sarà necessario utilizzare i comandi:

```
picture1.picture = LoadResPicture
    (16, 0)
form1.icon = LoadResPicture (24,1)
```

Logicamente, al posto i valori zero e uno è possibile usare le costanti vbResBitmap e vbResIcon.

La funzione **LoadResData (index, format)** è invece quella che permette di utilizzare più tipi di risorse ritornando come valore un array di valori di tipo byte. I dati che vengono letti dal file di risorse dalla funzione LoadResData possono essere superiori ai 64K. Come per le altre funzioni analizzate, il parametro index, obbligatorio, specifica il valore assegnato dal parametro name-id utilizzato nel file di definizione delle risorse per il quale il valore 1 (uno) è riservato all'icona dell'applicazione.

Per quanto riguarda il parametro format, anch'esso obbligatorio, esso specifica il formato originale dei tipo di dati restituiti che può assumere i valori in tabella 2 (oltre ad un valore di tipo stringa per le risorse definite dall'utente)

Se, ad esempio, si desidera utilizzare una risorsa di tipo video AVI, sarà necessario definirla nel file RC in questo modo:

```
1   PROVA_VIDEO"PROVA.AVI"
```

poi compilare il file RES e richiamarlo dal progetto con la seguente istruzione:

```
LoadResData(1, "PROVA_VIDEO"),
    vbUnicode
```

L'utilizzo di questa funzione con risorse di tipo bitmap, icona o cursore restituisce una stringa contenente i bit attuali nella risorsa: se si desidera utilizzare queste risorse sotto forma di immagini, è necessario utilizzare la funzione LoadResPicture.

Tabella 1

1	Risorsa di tipo Cursore
2	Risorsa di tipo Bitmap
3	Risorsa di tipo Icona
4	Risorsa di tipo Menu
5	Risorsa di tipo Finestra di dialogo
6	Risorsa di tipo Stringa
7	Risorsa di tipo Directory dei caratteri di sistema
8	Risorsa di tipo Carattere
9	Risorsa di tipo Tasti di scelta rapida
10	Risorsa di tipo definito dall'utente
12	Risorsa di tipo Gruppo di cursori
14	Risorsa di tipo Gruppo di icone

I tipi di risorse disponibili

IL FILE RISORSE MULTI-LINGUA

Molti di voi si chiederanno come ovviare al seguente inconveniente: si desidera creare un'applicazione internazionale e si vuole utilizzare un unico file di risorse per tutte le lingue supportate, in modo da compilare una sola volta il programma e lasciando all'utente la scelta della lingua da utilizzare a runtime. In questo modo, se le risorse vengono compilate con l'opzione di default loadoncall

(ovvero caricamento della risorsa soltanto quando effettivamente utilizzata), è vero che il nostro software conterrà nell'eseguibile tutte le risorse nelle varie lingue ma caricherà in memoria soltanto quelle della lingua prescelta. Un esempio di questa tecnica ci viene fornito dal progetto ATM.VBP contenuto nella directory \SAMPLES\RESOURCE: nel modulo *modATM.BAS* vi è la seguente dichiarazione:

```
Public i As Integer
```

che in questo programma assume valori differenti a seconda della lingua selezionata dall'utente mediante un click su un pulsante del form di apertura (vedi fig. 1):

```
i = 16  'Inglese
i = 48  'Francese
i = 80  'Tedesco
i = 112 'Italiano
i = 144 'Spagnolo
i = 176 'Giapponese
```

Ecco il contenuto dell'evento Form_Load di FRMINPUT.FRM, con le istruzioni che assegnano ai vari elementi dell'interfaccia le stringhe relative alla lingua selezionata

```
imgFlag.Picture = LoadResPicture(0 +
    i, vbResBitmap)
Me.Caption = LoadResString(0 + i)
lblPINCode.Caption = LoadResString
    (1 + i)
fraAccount.Caption = LoadResString
    (2 + i)
optChecking.Caption = LoadResString
    (3 + i)
optSavings.Caption = LoadResString
    (4 + i)
lblAmount.Caption = LoadResString
    (5 + i)
cmdOK.Caption = LoadResString(6 + i)
```

Se l'utente sceglie di utilizzare la lingua italiana, alla variabile *i* il programma assegna il valore 112, l'immagine del controllo *imgFlag* sarà la bandiera italiana e i valori assunti dalla caption dei controlli del form sono i seguenti:

```
Me.Caption = "Benvenuti"
lblPINCode.Caption = "Digitare il
    proprio codice segreto:"
fraAccount.Caption = "Scegliere il
    tipo di conto:"
optChecking.Caption = "Conto corrente"
optSavings.Caption = "Libretto di
    risparmio"
lblAmount.Caption = "Digitare la cifra
    richiesta"
cmdOK.Caption = "Attendere"
```

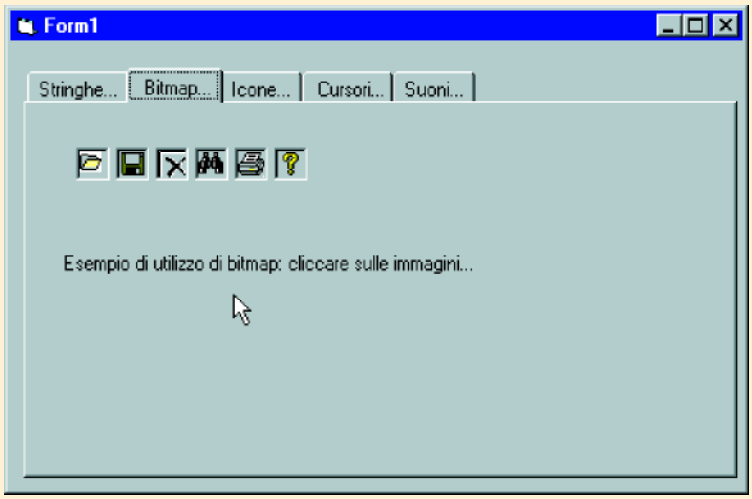

Figura 2

La schermata di apertura del programma ATM, fornito con il VB4 come esempio dell'uso dei file RES

UN ESEMPIO COMPLETO

Per aiutarvi nella creazione e nell'utilizzo dei file di risorse, ho creato un file RC ed un progetto di Visual Basic per dimostrare come utilizzare i vari tipi di risorse sino ad ora analizzate.

Il sorgente del file RC contiene sia risorse di tipo stringa che binarie. Il progetto di esempio, contenuto nel dischetto, è molto semplice: ho creato un form aggiungendogli il controllo SSTAB.OCX, visualizzando in ogni scheda un tipo di risorsa. Per prima cosa, all'evento *Form_Load( )* vengono caricate tutte le risorse: era possibile caricarle separatamente al momento il cui le singole schede ricevono il focus, ma poiché non sono poi così tante da creare problemi di memoria ho preferito questa soluzione più semplice.

Per quanto riguarda la gestione di risorse definite dall'utente, nell'esempio ho utilizzato una risorsa di tipo audio Wave, definendola come segue nel file RC:

```
1  SOUND   MOVEABLE    "CHIMES.WAV"
2  SOUND   MOVEABLE    "CHORD.WAV"
3  SOUND   MOVEABLE    "DING.WAV"
```

All'evento click dei pulsanti Command2, Command3, Command4 viene richiamata la seguente procedura "Suona"

```
Sub Suona(ByVal ResourceId As Integer)
Dim Ret As Variant
SoundBuffer = StrConv(LoadResData
    (ResourceId, "SOUND"), vbUnicode)
Ret = sndPlaySound(SoundBuffer,
    SND_ASYNC Or SND_NODEFAULT Or
    SND_MEMORY)
DoEvents
End Sub
```

Naturalmente, è necessario definire la chiamata a *mmsystem.dll* per la funzione *sndPlaySound* ed assegnare correttamente i *flag* richiesti dalla funzione stessa:

```
Declare Function sndPlaySound Lib
    "winmm.dll" Alias "sndPlaySoundA"
    (ByVal lpszSoundName As Any, ByVal
    uFlags As Long) As Long
Global Const SND_ASYNC = &H1
Global Const SND_NODEFAULT = &H2
Global Const SND_MEMORY = &H4
Global SoundBuffer As String
```

La stessa funzione LoadResData può essere utilizzata per richiamare qualsiasi tipo di risorsa multimediale (AVI, MID, ecc.) che altro tipo. Per quanto riguarda i comandi MCI per le risorse multimediali, vi rimando all'articolo "Multimedia e Visual Basic" di Roberto Ruggeri, apparso sul numero 4 di VB Journal.

Abbiamo terminato il nostro viaggio sulla creazione e l'utilizzo dei file RES: spero che questo articolo possa aiutarvi nel creare "applicazioni internazionali", in modo da diffondere sempre più il software "Made in Italy".

Roberto Scano ha conseguito il Diploma di Ragioniere e Perito Commerciale nel 1995. Ha esperienza di produzione e vendita software sia nel mercato shareware che commerciale, collabora spesso con riviste informatiche e da due anni si occupa di programmazione Windows in Visual Basic. Può essere raggiunto tramite telefono o fax al numero 041-5263540