Associare un programma ad un'estensione (prima parte)

http://www.vbsimple.net/howto/ht_054.htm

Difficoltà: 5 / 5

Lo sviluppo di un programma di frequente utilizzo che gestisce un particolare tipo di file è ben più semplice se si associa quel programma all'estensione dei files che esso richiama. Ad esempio i files con estensione VBP richiamano automaticamente Visual Basic, mentre quelli con estensione XLS ad esempio richiamano automaticamente Microsoft Excel.

Tale associazione è effettuata mediante registrazione di uno o più verbi assegnati all'estensione. Si raccomanda la consultazione delle Informazioni aggiuntive sulla Registrazione delle Estensioni in Windows.

Il problema non verrà risolto con la soluzione minima, sufficiente per registrare un'estensione nel sistema ed assegnargli un verbo per l'apertura, ma sarà invece sviluppata la classe ***clsFBIRegisterExtension*** in grado di registrare una o più estensioni ed assegnare con facilità uno o più verbi, ex-novo oppure riprendendo le impostazioni attuali del sistema.

Questo articolo presuppone anche una discreta conoscenza dell'organizzazione del registro di Windows ed utilizza la classe clsFBIRegistry trattata in un altro HowTo. Il presente articolo è pertanto soggetto alle medesime limitazioni della classe ***clsFBIRegistry*** e pertanto non funzionerà correttamente nei sistemi Windows NT.

La classe *clsFBIRegisterExtension* conta numerose proprietà ma un solo metodo pubblico di nome **Register**. Tutte le proprietà sono temporanee e saranno applicate soltanto dopo che viene richiamato il metodo Register. Esiste anche un evento di nome ***Error*** generato ogni volta che viene richiamato il metodo Register con dati non validi.

La classe consente la registrazione di un'estensione con e senza *deferring* dei verbi. È anche possibile lasciar rilevare le impostazioni attuali e/o preservare i verbi già esistenti.

Il codice in se stesso è semplice e l'unica difficoltà è data dal metodo Register e nei numerosi controlli effettuati. Si suppone che si conosca il funzionamento della classe ***clsFBIRegisty***.

```
1. Option Explicit
2.
3. Private m_strExt As String
4. Private m_strFileType As String
5. Private m_strDefaultIcon As String
6. Private m_strDefaultVerb As String
7. Private m_DeferredPath As String
8. Private m_DeferSettings As Boolean
```



```
 9. Private m_PreserveSettings As Boolean
10. Private m_SavedPath As String
11.
12. Private Declare Function ExtractIconEx Lib "SHELL32.DLL" Alias
    "ExtractIconExA" (ByVal lpszFile As String, ByVal nIconIndex As Long, phiconLarge
    As Long, phiconSmall As Long, ByVal nIcons As Long) As Long
13.
14. Public Event Error(ByRef Cancel As Boolean, ByVal Messaggio As String)
15.
```

Tutte le variabili locali dichiarate alle righe 3-10 corrispondono ai membri privati delle proprietà pubbliche della classe. La funzione *ExtractIconEx* dichiarata alla riga 12 verrà utilizzata per recuperare il numero di icone presenti nel file specificato. La stessa funzione API è trattata nell'HowTo dedicato all'estrazione di icone da un file.

L'evento **Error** dichiarato alla riga 14 sarà richiamato quando sarà rilevato qualche errore nel richiamo del metodo **Register**. L'argomento Cancel consentirà di nascondere la finestra con il messaggio di errore. L'argomento Messaggio invece conterrà una descrizione dell'errore.

```
16. Private Sub Class_Initialize()
17.   m_DeferSettings = True
18.   m_PreserveSettings = True
19. End Sub
20.
21. Private Function ShowError(ByVal blnCondition As Boolean, ByVal strMessage As
    String) As Boolean
22.   Dim blnCancel As Boolean
23.   If blnCondition Then
24.     blnCancel = False
25.     RaiseEvent Error(blnCancel, strMessage)
26.     If blnCancel = False Then MsgBox strMessage, vbCritical + vbOKOnly,
    "clsFBIRegisterExtension"
27.   End If
28.   ShowError = blnCondition
29. End Function
30.
```

All'istanza della classe i membri **m_DeferSettings** e **m_PreserveSettings** sono inizializzati a True, secondo le impostazioni normali della classe. Vedremo più avanti di cosa si tratta.

La funzione ***ShowError*** consente di valutare una condizione (specificata nel parametro **blnCondition**) e se tale espressione restituisce il valore Vero, sarà generato l'evento Error, accompagnato da una descrizione dell'errore (parametro **strMessage**). Se al ritorno dall'evento la variabile **blnCancel** contiene ancora il valore False sarà allora mostrato un avviso di errore.

La funzione restituisce in uscita la valutazione della condizione. Risulta molto utile per generare automaticamente degli errori in base alla condizione **blnCondition**, gestire l'evento Error e restituire il valore della condizione stessa.

```
31. Public Property Get DefaultIcon() As String
32.   DefaultIcon = m_strDefaultIcon
33. End Property
34.
35. Public Property Let DefaultIcon(ByVal newDefaultIcon As String)
36.   m_strDefaultIcon = Trim$(newDefaultIcon)
37. End Property
38.
```


La proprietà ***DefaultIcon*** consente di associare un'icona all'estensione utilizzata. È fondamentale ricordare che l'icona sarà impostata soltanto successivamente alla chiamata del metodo Register. La lettura della proprietà prima della registrazione è assolutamente inutile e non sarà in alcun modo rilevata l'icona attuale. Inoltre non sarà effettuato alcun tipo di verifica sul valore fornito.
Lasciando questa proprietà in bianco sarà mantenuta l'icona attuale registrata nel sistema.

```
39. Public Property Get DefaultVerb() As String
40.   DefaultVerb = m_strDefaultVerb
41. End Property
42.
43. Public Property Let DefaultVerb(ByVal newDefaultVerb As String)
44.   m_strDefaultVerb = newDefaultVerb
45. End Property
46.
```

La proprietà **DefaultVerb** consente di definire il verbo predefinito che sarà impostato durante la registrazione.

```
47. Public Property Get DeferredPath() As String
48.   DeferredPath = m_DeferredPath
49. End Property
50.
51. Public Property Let DeferredPath(ByVal newDeferredPath As String)
52.   If ShowError(Len(newDeferredPath) = 0, Error$(380)) Then Exit Property
53.   m_DeferredPath = newDeferredPath
54. End Property
55.
```

La proprietà **DeferredPath** consente di assegnare un nuovo valore in caso si scelga di utilizzare il deferring della registrazione. Alla proprietà non può però essere assegnato un valore nullo.

```
56. Public Property Get DeferSettings() As Boolean
57.   DeferSettings = m_DeferSettings
58. End Property
59.
60. Public Property Let DeferSettings(ByVal newDeferSettings As Boolean)
61.   m_DeferSettings = newDeferSettings
62. End Property
63.
```

La proprietà **DeferSettings** consente di decidere se utilizzare o meno il deferring della registrazione. Tale valore sarà però ignorato se si sceglie di preservare le impostazioni attuali (proprietà **PreserveSettings**) e naturalmente la registrazione esiste già.

```
64. Public Property Get Extension() As String
65.   Extension = m_strExt
66. End Property
67.
68. Public Property Let Extension(ByVal newExtension As String)
69.   If Left$(newExtension, 1) = "." Then newExtension = Mid$(newExtension, 2)
70.   newExtension = Trim$(newExtension)
71.   If ShowError(Len(newExtension) = 0, Error$(380)) Then Exit Property
72.   m_strExt = newExtension
73.   If Len(m_DeferredPath) = 0 Then m_DeferredPath = m_strExt & "File"
74. End Property
75.
```

La proprietà **Extension** definisce l'estensione utilizzata ed alla quale sarà applicata la registrazione. Se l'estensione contiene il punto iniziale esso sarà rimosso alla riga 69. Il

nuovo valore della proprietà non può essere nullo ed in tal caso sarà generato un errore (riga 71).

Alla riga 73 invece è assegnato il nuovo valore corrispondente alla proprietà **DeferredPath**, soltanto se non è stato assegnato precedentemente. Il valore predefinito di tale proprietà corrisponde quindi all'estensione utilizzata seguita dal suffisso ***File***, nel rispetto delle definizioni standard del registro di Windows.

```
 76. Public Property Get FileType() As String
 77.   FileType = m_strFileType
 78. End Property
 79.
 80. Public Property Let FileType(ByVal newFileType As String)
 81.   m_strFileType = Trim$(newFileType)
 82. End Property
 83.
```

La proprietà **FileType** consente di associare una descrizione alla nuova estensione da registrare.

```
 84. Public Property Get IconCount() As Long
 85.   Dim strPath As String
 86.   Dim intPos As Integer
 87.   If ShowError(Len(m_strDefaultIcon) = 0, "Specificare un valore per la proprietà
     DefaultIcon") Then Exit Property
 88.   strPath = m_strDefaultIcon
 89.   intPos = InStr(1, strPath, ",")
 90.   If intPos > 0 Then strPath = Left$(m_strDefaultIcon, intPos - 1)
 91.   IconCount = ExtractIconEx(strPath, -1, ByVal 0&, ByVal 0&, ByVal 0&)
 92. End Property
 93.
```

La proprietà in sola lettura **IconCount** restituisce il numero di icone contenute nel file indicato dalla proprietà **DefaultIcon**. In caso venga letto il numero di icone prima di assegnare la nuova icona sarà generato un errore (riga 87).

Dal percorso dell'icona sarà eliminato l'indicatore dell'icona da utilizzare, cioè quello che segue la virgola nel nome del file. Ottenuto quindi il nome completo del file dell'icona sarà estratto il numero di icone contenute in esso mediante l'API *ExtractIconEx*. Il valore -1 consente infatti di recuperare soltanto il numero di icone contenute senza estrarre null'altro dal file.

```
 94. Public Property Get PreserveSettings() As Boolean
 95.   PreserveSettings = m_PreserveSettings
 96. End Property
 97.
 98. Public Property Let PreserveSettings(ByVal newPreserveSettings As Boolean)
 99.   m_PreserveSettings = newPreserveSettings
100. End Property
101.
```

La proprietà **PreserveSettings** consente di preservare le impostazioni di registrazione attuali ovvero se la registrazione attuale utilizza deferring anche la nuova registrazione continuerà ad utilizzarlo; sarà inoltre impossibile sovrascrivere un verbo esistente.

```
102. Public Property Get SavedPath() As String
103.   SavedPath = m_SavedPath
104. End Property
105.
```


L'ultima proprietà della classe è **SavedPath**, in sola lettura e consente di recuperare il percorso in cui è stata registrata l'ultima estensione. Utile per determinare la chiave in cui sono stati registrati i verbi nell'operazione Register, quando la proprietà **PreserveSettings** è impostata su True.

Segue Parte 2 >>

Fibia FBI
22 Dicembre 2002

Torna all'indice degli HowTo

