



Utilizzare il registro di Windows

(terza parte) 

http://www.vbsimple.net/howto/ht_044_3.htm

Difficoltà:  5 / 5

[<< Continua dalla parte 2](#)

Quella che segue è la parte più complessa dell'intero articolo: saranno trattati i metodi  pubblici e privati della nostra classe *clsFBIRegistry*.

```

185. Public Function ApriChiave(ByVal ChiavePrincipale As ChiaviPrincipali, ByVal
    SottoChiave As String) As Long
186.     Dim oldKey As Long
187.     oldKey = lngKeyValue
188.     If ChiavePrincipale = CHIAVE_APERTA Then ChiavePrincipale = lngKeyValue
189.     lngKeyValue = 0
190.     Call RegOpenKeyEx(ChiavePrincipale, SottoChiave, ByVal 0&, lngKeySecurity,
        lngKeyValue)
191.     ApriChiave = lngKeyValue
192.     Call RegCloseKey(oldKey)
193. End Function
194.
195. Public Sub ChiudiChiave()
196.     Call RegCloseKey(lngKeyValue)
197.     lngKeyValue = 0
198. End Sub
199.

```

Le prime due funzioni sono le più scontate: **ApriChiave** effettua l'apertura di una chiave o di una sottochiave, mentre **ChiudiChiave** chiude la chiave aperta. Per la prima funzione potrà essere fornito l'handle ad una qualsiasi chiave aperta (da un'altra istanza o processo) oppure una chiave di sistema (come definito dall'enumerazione  **ChiaviPrincipali**). Nel caso venisse utilizzato il valore **CHIAVE_APERTA** sarà utilizzato l'handle della chiave aperta dall'istanza della nostra classe. Il parametro sottochiave invece identifierà la sottochiave da aprire rispetto alla chiave specificata.

La funzione **ChiudiChiave** invece effettua semplicemente la chiusura della chiave e l'azzeramento dell'handle **lngKeyValue**.

```

200. Public Function CreaChiave(ByVal ChiavePrincipale As ChiaviPrincipali, ByVal
    SottoChiave As String, Optional ByVal NonCambiare As Boolean = False) As Long
201.     Dim newKeyValue As Long
202.     If ChiavePrincipale = CHIAVE_APERTA Then ChiavePrincipale = lngKeyValue
203.     Call RegCreateKeyEx(ChiavePrincipale, SottoChiave, ByVal 0&, vbNullChar,
        REG_OPTION_NON_VOLATILE, lngKeySecurity, ByVal 0&, newKeyValue, ByVal 0&)
204.     If NonCambiare = False Then
205.         Call RegCloseKey(lngKeyValue)
206.         lngKeyValue = newKeyValue
207.     End If
208.     CreaChiave = newKeyValue
209. End Function
210.

```

In maniera analoga alla funzione vista in precedenza, la funzione **CreaChiave** consente la creazione e l'apertura di una sottochiave. Se la sottochiave richiesta non esiste essa verrà creata ed aperta. Il funzionamento è analogo alla funzione **ApriChiave** ma il parametro aggiuntivo **NonCambiare** consente di effettuare le due succitate operazioni senza tuttavia chiudere e cambiare la chiave utilizzata dall'istanza, ed indicata dalla proprietà **Chiave**.

Questa possibilità si rivela utile quando si desidera creare una nuova sottochiave, ottenerne l'handle (come valore di ritorno della funzione) senza però chiudere la chiave sulla quale l'istanza sta lavorando.

```

211. Public Function ElencaChiave(ByVal ChiavePrincipale As ChiaviPrincipali, ByVal
    Indice As Long) As String
212.     Dim Buffer As String
213.     Dim Lunghezza As Long
214.     If ChiavePrincipale = CHIAVE_APERTA Then ChiavePrincipale = lngKeyValue
215.     Buffer = String$(255, 0)
216.     Lunghezza = 255
217.     Call RegEnumKeyEx(ChiavePrincipale, Indice, Buffer, Lunghezza, ByVal 0&,
        vbNullString, ByVal 0&, ByVal 0&)
218.     ElencaChiave = Left$(Buffer, Lunghezza)
219. End Function
220.

```

La funzione **ElencaChiave** consente di recuperare il nome di una sottochiave in base al suo ordine, ad esempio la prima, la seconda o la terza sottochiave della chiave aperta. Viene utilizzata a tale scopo la funzione API *RegEnumKeyEx*.

```

221. Public Function ElencaChiavi(ByVal ChiavePrincipale As ChiaviPrincipali) As Variant
222.     Dim Elenco() As String
223.     Dim Conta As Long
224.     Dim Buffer As String
225.     If ChiavePrincipale = CHIAVE_APERTA Then ChiavePrincipale = lngKeyValue
226.     Call RegQueryInfoKey(ChiavePrincipale, vbNullChar, ByVal 0&, ByVal 0&, Conta,
        ByVal 0&, ByVal 0&, ByVal 0&, ByVal 0&, ByVal 0&, ByVal 0&, ByVal 0&)
227.     If Conta > 0 Then
228.         ReDim Elenco(Conta - 1) As String
229.         Buffer = String$(255, 0)
230.         Conta = 0
231.         Do While RegEnumKeyEx(ChiavePrincipale, Conta, Buffer, Len(Buffer), ByVal 0&,
            vbNullString, ByVal 0&, ByVal 0&) = 0
232.             Elenco(Conta) = Left$(Buffer, InStr(1, Buffer, Chr$(0), vbBinaryCompare) - 1)
233.             Conta = Conta + 1
234.             Buffer = String$(255, 0)
235.         Loop
236.         ElencaChiavi = Elenco
237.     Else
238.         ElencaChiavi = Null
239.     End If
240. End Function
241.

```

In maniera analoga alla funzione precedente, la funzione **ElencaChiavi** restituisce un [array](#) contenente i nomi di tutte le sottochiavi presenti all'interno della chiave specificata. Se nessuna sottochiave è presente, restituirà valore Null. Utilizza la funzione *RegQueryInfoKey* allo stesso modo della proprietà NumeroSottochiavi per recuperare il numero complessivo delle sottochiavi ed allocare così il buffer necessario e poi ne estrae uno per uno i nomi utilizzando la funzione *RegEnumKeyEx* nella stessa maniera della funzione **ElencaChiave**.

```

242. Public Function ElencaValore(ByVal ChiavePrincipale As ChiaviPrincipali, ByVal
    Indice As Long) As String

```

```

243. Dim Buffer As String
244. Dim Lunghezza As Long
245. If ChiavePrincipale = CHIAVE_APERTA Then ChiavePrincipale = lngKeyValue
246. Buffer = String$(255, 0)
247. Lunghezza = 255
248. Call RegEnumValue(ChiavePrincipale, Indice, Buffer, Lunghezza, ByVal 0&, ByVal
    0&, ByVal 0&, ByVal 0&)
249. ElencaValore = Left$(Buffer, Lunghezza)
250. End Function
251.

```

La funzione **ElencaValore** consente di recuperare il nome di un valore specifico, di cui si possiede l'indice, all'interno della chiave indicata. Il funzionamento è molto simile a quello della funzione **ElencaChiave** con l'unica differenza che utilizza la funzione *RegEnumValue* per recuperare l'informazione dal registro.

```

252. Public Function ElencaValori(ByVal ChiavePrincipale As ChiaviPrincipali) As Variant
253. Dim Elenco() As String
254. Dim Conta As Long
255. Dim Buffer As String
256. If ChiavePrincipale = CHIAVE_APERTA Then ChiavePrincipale = lngKeyValue
257. Call RegQueryInfoKey(lngKeyValue, vbNullChar, ByVal 0&, ByVal 0&, ByVal 0&, ByVal
    0&, ByVal 0&, Conta, ByVal 0&, ByVal 0&, ByVal 0&, ByVal 0&, ByVal 0&)
258. If Conta > 0 Then
259. ReDim Elenco(Conta - 1)
260. Buffer = String$(255, 0)
261. Conta = 0
262. Do While RegEnumValue(ChiavePrincipale, Conta, Buffer, Len(Buffer), ByVal 0&,
    ByVal 0&, ByVal 0&, ByVal 0&) = 0
263. Elenco(Conta) = Left$(Buffer, InStr(1, Buffer, Chr$(0), vbBinaryCompare) - 1)
264. Conta = Conta + 1
265. Buffer = String$(255, 0)
266. Loop
267. ElencaValori = Elenco
268. Else
269. ElencaValori = Null
270. End If
271. End Function
272.

```

E come la sua parente, la funzione **ElencaValori** restituisce un array contenente i nomi di tutti i valori presenti all'interno della chiave specificata. Restituisce invece Null in assenza di valori all'interno della chiave.

```

273. Public Function EliminaChiave(ByVal ChiavePrincipale As ChiaviPrincipali, Optional
    ByVal SottoChiave As String) As Boolean
274. If ChiavePrincipale = CHIAVE_APERTA Then ChiavePrincipale = lngKeyValue
275. EliminaChiave = (RegDeleteKey(ChiavePrincipale, SottoChiave) = 0)
276. End Function
277.
278. Public Function EliminaValore(Optional ByVal NomeValore As String = "") As Boolean
279. EliminaValore = (RegDeleteValue(lngKeyValue, NomeValore) = 0)
280. End Function
281.

```

Le due funzioni di eliminazione: **EliminaChiave** ed **EliminaValore** consentono l'eliminazione rispettivamente di una chiave e di un valore.

Tralasciamo per il momento l'ordine alfabetico e, saltando la funzione *Esporta*, passiamo direttamente alla funzione **ForzaAggiornamentoChiave**. Vista la loro complessità, le due funzioni di esportazione ed importazione saranno trattate nella prossima parte.

```

282. Public Sub ForzaAggiornamentoChiave()

```

```
283. Call RegFlushKey(lngKeyValue)
284. End Sub
285.
```

La sua utilità è dubbia e potrebbe generare sospetti sul corretto funzionamento della classe. Nel momento in cui viene richiesta la modifica di un valore nel registro i dati sono effettivamente alterati ma non immediatamente scritti su disco. Se ipoteticamente, un secondo dopo un'operazione di scrittura sul registro, andasse via la luce, al riavvio del computer il valore potrebbe non essere stato ancora scritto.

Questo perché Windows utilizza un sistema di cache e di scrittura ritardata (*lazy write*) che non impegna troppo il computer in lente operazioni di scrittura. Essa seguirà infatti nei momenti più liberi del computer.

La funzione **ForzaAggiornamentoChiave** e quindi *RegFlushKey* obbligano il sistema operativo a scrivere su disco tutte le modifiche relative alla chiave specificata, garantendo l'immediata scrittura su disco ma rallentando pericolosamente la velocità del sistema. In linea generale questa funzione non è necessaria ma in rarissimi casi può richiedersi utile. La funzione *RegFlushKey* viene chiamata dal sistema operativo sulle chiavi di sistema al momento dello spegnimento della macchina per assicurare la corretta scrittura dei dati su disco.

Prima di vedere le ultime due funzioni pubbliche relative all'esportazione ed all'importazione dei dati, diamo un rapido sguardo ad alcune funzioni private utilizzate in altre parti del codice.

```
286. Private Function DeQuote(ByVal Stringa As String) As String
287.   If Left$(Stringa, 1) = "\"" Then Stringa = Mid$(Stringa, 2)
288.   If Right$(Stringa, 1) = "\"" Then Stringa = Left$(Stringa, Len(Stringa) - 1)
289.   DeQuote = Stringa
290. End Function
291.
292. Private Function Ceil(ByVal Numero As Double) As Double
293. Ceil = Fix(Numero)
294. If Numero > Fix(Numero) Then Ceil = Ceil + 1
295. End Function
296.
```

La funzione **DeQuote** consente di rimuovere l'eventuali virgolette (") alla sinistra ed alla destra di una stringa. Non rimuovono tutte le virgolette ma soltanto un singolo paio. Viene utilizzata per recuperare il nome reale dei valori durante la fase di importazione da file.

La funzione **Ceil** prende il nome dalla funzione analoga del linguaggio JavaScript e restituisce il numero arrotondato **sempre** per eccesso. È utilizzata per determinare l'ampiezza del buffer durante il processo di importazione.

```
297. Private Function HexDouble(ByVal Numero As Double, Optional Padding As Integer) As
String
298.   Dim HighPart As Double
299.   Numero = Fix(Numero)
300.   HighPart = Fix(Numero / 65536)
301.   Numero = Numero - HighPart * 65536
302.   HexDouble = Hex$(HighPart) & Right$("0000" & Hex$(Numero), 4)
303.   If Padding > 0 Then HexDouble = Right$(String$(Padding, "0") & HexDouble,
Padding)
```

```
304. End Function
305.
```

Una funzione un po' unusuale è la **HexDouble** ed in un linguaggio quale il C non avrebbe probabilmente senso di esistere. Viene utilizzata durante il processo di esportazione per convertire un numero Double ovvero il valore DWORD (visto che in VB non è possibile definire Long senza segno) in una stringa esadecimale. L'istruzione *Hex* arriva infatti soltanto al 31° bit ed al superamento di questo, genera *Overflow*.

La funzione scompone il numero Double in due parti: la parte alta è mantenuta nella variabile **HighPart** mentre la parte bassa rimane nel parametro Numero. I due valori sono poi convertiti separatamente e le due stringhe unite solo in seguito alla conversione.

Il parametro Padding è utile per assicurare una lunghezza minima di quella specificata in tale argomento; eventuali cifre mancanti saranno riempite con zeri alla sinistra.

```
306. Private Function Replace(ByVal Stringa As String, ByVal Trova As String, ByVal
    Rimpiazza As String) As String
307.     Dim Pos As Integer
308.     Dim Buffer As String
309.     Pos = InStr(1, Stringa, Trova)
310.     Buffer = ""
311.     Do While Pos > 0
312.         Buffer = Buffer & Left$(Stringa, Pos - 1) & Rimpiazza
313.         Stringa = Mid$(Stringa, Pos + Len(Trova))
314.         Pos = InStr(1, Stringa, Trova)
315.     Loop
316.     Replace = Buffer & Stringa
317. End Function
318.
```

La funzione **Replace** è del tutto inutile ai programmatori VB6 ma per chi usufruisce (*come me*) della versione 5 di VB essa è indispensabile. Com'è ovvio, consente di sostituire tutte le occorrenze di una data stringa con un'altra e presenta alcune marginali differenze rispetto alla funzione *Replace* nativa in VB6.

[Segue parte 4 >>](#)

[Fibia FBI](#)

1 Aprile 2002

Corretto il 20 Settembre 2002



[Torna all'indice degli HowTo](#)
