



Utilizzare il registro di Windows (quarta parte)

http://www.vbsimple.net/howto/ht_044_4.htm

Difficoltà:  5 / 5

[<< Continua dalla parte 3](#)

Ci avviciniamo alla fine di questo lungo tutorial. Saranno trattate le ultime quattro funzioni della nostra classe *FBIRegistry* relative ai processi di esportazione ed importazione dei dati del registro in file.

```

319. Public Function Esporta(ByVal ChiavePrincipale As ChiaviPrincipali, ByVal
    SottoChiave As String, ByVal NomeFile As String, ByVal SottoChiavi As Boolean,
    ByVal Aggiunge As Boolean) As Boolean
320.     Dim FileNR As Integer
321.     Dim oldKey As Long
322.     Dim NomeChiave As String
323.
324.     If ChiavePrincipale = CHIAVE_APERTA Then ChiavePrincipale = lngKeyValue
325.     Select Case ChiavePrincipale
326.         Case HKEY_CLASSES_ROOT: NomeChiave = "HKEY_CLASSES_ROOT"
327.         Case HKEY_CURRENT_CONFIG: NomeChiave = "HKEY_CURRENT_CONFIG"
328.         Case HKEY_CURRENT_USER: NomeChiave = "HKEY_CURRENT_USER"
329.         Case HKEY_DYN_DATA: NomeChiave = "HKEY_DYN_DATA"
330.         Case HKEY_LOCAL_MACHINE: NomeChiave = "HKEY_LOCAL_MACHINE"
331.         Case HKEY_PERF_ROOT: NomeChiave = "HKEY_PERF_ROOT"
332.         Case HKEY_PERFORMANCE_DATA: NomeChiave = "HKEY_PERFORMANCE_DATA"
333.         Case HKEY_USERS: NomeChiave = "HKEY_USERS"
334.         Case Else
335.             MsgBox "Per salvare una chiave è necessario che la chiave principale sia una
                chiave di sistema.", vbCritical + vbOKOnly, "FBIRegistry"
336.             Esporta = False
337.             Exit Function
338.     End Select

```

La funzione **Esporta** consente il salvataggio di una chiave, delle sue sottochiavi e dei loro valori in un file di testo compatibile con Regedit. La funzione richiede tre argomenti fondamentali: la chiave di sistema in cui si trova la sottochiave, il percorso logico della sottochiave rispetto alla chiave di sistema ed il nome del file in cui salvare tali dati.

Gli altri due argomenti consentono di salvare soltanto la chiave specificata o anche tutte le sue sottochiavi. L'ultimo argomento specifica se creare un nuovo file di testo oppure aggiungere i dati a quelli già esistenti nel file.

Il controllo alle righe 325-338 assicura che la chiave specificata sia una chiave di sistema nell'enumerazione **ChiavePrincipali** ed in tal caso ne registra il nome nella variabile **NomeChiave**. Se la chiave specificata non è una chiave di sistema verrà generato un avviso e l'esportazione verrà interrotta.

```

339.     FileNR = FreeFile
340.     If Aggiunge Then
341.         Open NomeFile For Append As FileNR

```

```

342. Else
343.     Open NomeFile For Output As FileNR
344. End If
345. If LOF(FileNR) = 0 Then Print #FileNR, "REGEDIT4" & vbNewLine
346. oldKey = lngKeyValue
347. lngKeyValue = 0
348. Call RegOpenKeyEx(ChiavePrincipale, SottoChiave, ByVal 0&, lngKeySecurity,
    lngKeyValue)
349. If Left$(SottoChiave, 1) <> "\" Then SottoChiave = "\" & SottoChiave
350. If SottoChiave = "\" Then SottoChiave = ""
351. SalvaRamo CHIAVE_APERTA, NomeChiave & SottoChiave, FileNR, SottoChiavi
352. Print #FileNR, vbNullString
353. Close FileNR
354. FileNR = 0
355. Call RegCloseKey(lngKeyValue)
356. lngKeyValue = oldKey
357. Esporta = True
358. End Function
359.

```

In funzione dell'argomento **Aggiungi** verrà creato un nuovo file di testo sovrascrivendo i dati precedenti oppure il file sarà aperto in modalità di *Append* per inserire nuovi dati (righe 340-344). Inoltre se la dimensione del file aperto è di zero bytes (ovvero il file è nuovo oppure vengono aggiunti dati su un file vuoto) verrà inserita una breve intestazione per rendere il file riconoscibile da Regedit (riga 345).

Solo in seguito a questi necessari controlli verrà aperta la chiave richiesta per l'esportazione e passata con altri dati alla funzione **SalvaRamo** che vedremo tra poco. In sostanza la funzione SalvaRamo effettua l'operazione di esportazione dei dati vera e propria utilizzando l'handle di file passatole come argomento della funzione.

La nostra funzione **Esporta** infatti si occuperà soltanto dei controlli iniziali, dell'apertura del file, del richiamo della funzione **SalvaRamo** e della chiusura di file e chiave. Questo per dare completa flessibilità alla funzione **SalvaRamo** che non dovrà occuparsi di tutti questi noiosi compiti.

```

360. Private Sub SalvaRamo(ByVal Chiave As ChiaviPrincipali, ByVal Percorso As String,
    ByVal FileNR As Integer, ByVal SottoChiavi As Boolean)
361. Dim Conta As Integer
362. Dim Conta2 As Long
363. Dim Elementi As Variant
364. Dim ValoreElemento As Variant
365. Dim TipoDati As TipoValoriRegistro
366. Dim Buffer As String
367. Dim Buffer2 As String
368. If Chiave = CHIAVE_APERTA Then Chiave = lngKeyValue
369. Print #FileNR, "[" & Percorso & "]"
370. Elementi = Me.ElencaValori(CHIAVE_APERTA)
371. If Not IsNull(Elementi) Then
372.     For Conta = LBound(Elementi) To UBound(Elementi)
373.         ValoreElemento = Me.Valore(Elementi(Conta), TipoDati)
374.         Buffer2 = ""
375.         Elementi(Conta) = Replace(Elementi(Conta), "\", "\\")
376.         Elementi(Conta) = Replace(Elementi(Conta), vbCr, "\r")
377.         Elementi(Conta) = Replace(Elementi(Conta), vbLf, "\n")
378.         Elementi(Conta) = Replace(Elementi(Conta), """", "\\\"")
379.         Buffer = "" & Elementi(Conta) & ""
380.         If Elementi(Conta) = "" Then Buffer = "@"

```

La funzione SalvaRamo, la più complessa dell'intero progetto, effettua il salvataggio del contenuto di una chiave su un file già aperto, mantenendo naturalmente il formato dei dati originali. La funzione richiede quattro parametri: **Chiave** è la chiave che si intende salvare,

Percorso è il percorso logico della chiave nella struttura ad albero, **FileNR** è l'handle del file già aperto in cui scrivere i dati ed infine **Sottochiavi** determina se effettuare il salvataggio anche delle sottochiavi e dei valori in esse contenuti.

La funzione si apre con il salvataggio su file del percorso (riga 369) ed il recupero di tutti i valori presenti nella chiave specificata (riga 370). Per ogni valore recuperato sarà ottenuto anche il suo contenuto (riga 373) ed effettuate alcune semplici conversioni per rendere l'esportazione coerente con il formato REGEDIT4 (righe 375-380). Il valore predefinito della chiave deve essere salvato come "@".

```

381.      Select Case TipoDati
382.      Case REG_SZ
383.          ValoreElemento = Replace(ValoreElemento, "\", "\\")
384.          ValoreElemento = Replace(ValoreElemento, vbCr, "\r")
385.          ValoreElemento = Replace(ValoreElemento, vbLf, "\n")
386.          ValoreElemento = Replace(ValoreElemento, " ", "\ ")
387.          Buffer2 = Buffer & "=" & ValoreElemento & " "

```

Per fortuna o purtroppo il registro contiene una varietà di tipologie di dati presenti e soprattutto non tutti questi dati sono scritti nella maniera corretta. Per ogni tipo di dato trattato sarà necessario effettuare alcuni aggiustamenti in esportazione.

L'esempio più semplice è il formato stringa **REG_SZ** che richiede soltanto la sostituzione di alcuni caratteri speciali ("\", Enter, a capo, virgolette) nelle corrispondenti sequenze escape e ciò viene effettuato utilizzando la funzione **Replace**.

Purtroppo il caso degli altri tipi di dati non è altrettanto semplice. Esistono infatti alcuni dati che, pur essendo ad esempio valori **REG_DWORD** non contengono un valore corretto ma soltanto una matrice di bytes, quasi che si trattasse di dati **REG_BINARY**.

```

388.      Case REG_BINARY, REG_NONE, REG_EXPAND_SZ, _
389.          REG_DWORD_LITTLE_ENDIAN, REG_DWORD, _
390.          REG_MULTI_SZ
391.      If IsArray(ValoreElemento) Then
392.          Buffer = Buffer & "=hex"
393.          If TipoDati = REG_NONE Then Buffer = Buffer & "(0)"
394.          If TipoDati = REG_EXPAND_SZ Then Buffer = Buffer & "(2)"
395.          If TipoDati = REG_DWORD Then Buffer = Buffer & "(4)"
396.          If TipoDati = REG_MULTI_SZ Then Buffer = Buffer & "(7)"
397.          Buffer = Buffer & ":"
398.          If Not IsNull(ValoreElemento) Then
399.              For Conta2 = LBound(ValoreElemento) To UBound(ValoreElemento)
400.                  Buffer = Buffer & Right$("00" & LCase$(Hex$(ValoreElemento
(Conta2))), 2)
401.                  If Conta2 < UBound(ValoreElemento) Then Buffer = Buffer & ","
402.                  If (Len(Buffer) > 76) And (Conta2 < UBound(ValoreElemento)) Then
403.                      Buffer2 = Buffer2 & Buffer & "\" & vbNewLine
404.                      Buffer = " "
405.                  End If
406.              Next Conta2
407.          End If
408.          Buffer2 = Buffer2 & Buffer

```

Nel caso del tipo di dati binario o DWORD dovrà quindi esser fatta un'investigazione più profonda. Il primo di questi controlli consiste nel verificare se i dati sono in forma di array ed in tal caso dovranno essere trattati come dati binari, pur mantenendo il formato di dati originale. Tutti i dati binari devono essere salvati nel formato "hex(x): XX, YY". dove x indica il tipo di dati e XX, YY sono i dati in forma binaria esadecimale. L'unica eccezione

sta nel caso dei dati **REG_BINARY**, che non richiedono le parentesi che racchiudono il tipo di dati. Tale formattazione di dati è fatta alle righe 392-397.

Soltanto adesso sarà possibile convertire i singoli bytes della matrice in stringhe esadecimali utilizzando l'istruzione *Hex\$*. Ma, ahinoi, non ancora finito qui: ogni riga da salvare che superi la lunghezza di 76 caratteri dovrà essere spezzata in altre righe (righe 402-405).

```

409.         ElseIf TipoDati = REG_DWORD Then
410.             Buffer = Buffer & "=dword:"
411.             Buffer2 = Buffer & LCase$(HexDouble(ValoreElemento, 8))
412.         Else
413.             Buffer = Buffer & "=hex"
414.             If TipoDati <> REG_BINARY Then Buffer = Buffer & "(" & CStr(TipoDati) &
" )"
415.             Buffer = Buffer & ":"
416.             If Not IsNull(ValoreElemento) Then Buffer = Buffer & CStr
(ValoreElemento)
417.             Buffer2 = Buffer
418.         End If
419.         Case REG_DWORD_BIG_ENDIAN, REG_LINK, _
420.             REG_RESOURCE_LIST
421.             MsgBox "Tipo di dati non implementato!", vbCritical + vbOKOnly,
"FBIRegistry"
422.         End Select
423.         Print #FileNR, Buffer2
424.     Next Conta
425. End If

```

Se invece i dati recuperati non sono in forma di matrice, sarà verificato che il tipo di dati sia **DWORD** perché anche questo tipo richiede una sua particolare formattazione come "dword: XX" dove XX è il valore **DWORD** convertito in stringa esadecimale (righe 409-411).

Se i dati non sono né una matrice di bytes né un valore **DWORD** allora saranno semplicemente scritti come numero decimale (se i dati esistono) oppure non verranno scritti affatto. Sarà cioè scritto soltanto il tipo di dati trattato come "hex(x):" ma non il loro contenuto. Non si tratta infatti di una possibilità remota ma di una consuetudine nell'universo del registro di Windows.

I tipi di dati **REG_DWORD_BIG_ENDIAN**, **REG_LINK** e **REG_RESOURCE_LIST** non saranno affatto trattati e quindi esclusi dal processo di esportazione.

Alla riga 423 verrà infine scritto il buffer di dati preparato mediante i controlli e le conversioni appena fatti.

```

426.     If (SottoChiavi = True) And (Me.NumeroSottoChiavi > 0) Then
427.         Elementi = ElencaChiavi(CHIAVE_APERTA)
428.         If IsNull(Elementi) Then Exit Sub
429.         For Conta = LBound(Elementi) To UBound(Elementi)
430.             Print #FileNR, vbNullString
431.             Conta2 = lngKeyValue
432.             Call RegOpenKeyEx(Chiave, Elementi(Conta), ByVal 0&, lngKeySecurity,
lngKeyValue)
433.             SalvaRamo CHIAVE_APERTA, Percorso & "\" & Elementi(Conta), FileNR,
SottoChiavi
434.             Call RegCloseKey(lngKeyValue)
435.             lngKeyValue = Conta2
436.         Next Conta
437.     End If

```

438. End Sub
439.

Un'intera chiave è stata salvata su file. Resta pertanto l'ultima possibilità ovvero quella di dover salvare ogni singola sottochiave della nostra chiave e così le sottochiavi delle sottochiavi, etc... Quest'operazione è svolta nel ciclo descritto alle righe 429-436.

Sarà quindi aperta una sottochiave per volta, ricreato il percorso e richiamata [ricorsivamente](#) la funzione **SalvaRamo** con i nuovi dati. La ricorsività assicurerà il salvataggio di tutti i valori presenti in ogni sottochiave di ogni sottochiave della prima chiave.

Questa è la ragione principale per cui abbiamo preferito separare la routine **Esporta** da quella **SalvaRamo**.

[Segue parte 5 >>](#)

[Fibia FBI](#)

1 Aprile 2002

Corretto il 20 Settembre 2002



[Torna all'indice degli HowTo](#)
