Inviare un'email con allegati

http://www.vbsimple.net/howto/ht_020_2.htm

Difficoltà: 3 / 5

Nell'articolo precedente abbiamo visto com'è semplice inviare un'e-mail utilizzando il protocollo mailto; il limite maggiore di quella soluzione è quello di non poter allegare uno o più files al messaggio. Viste le numerosissime richieste sull'argomento abbiamo preferito affrontare anche una seconda soluzione che superasse questo limite. Quest'articolo espone una soluzione leggermente più complessa ma che consente di allegare files all'e-mail.

La soluzione è basata sull'uso delle funzioni Simple-MAPI (Messaging API) e pertanto richiede l'uso di un client di posta elettronica MAPI, come Outlook Express, correttamente configurato. L'insieme delle istruzioni è stato raggruppato in un modulo di classe semplicemente riutilizzabile.
Il codice si apre con la dichiarazione di una serie di enumerazioni, costanti e funzioni API: non saranno mostrate tutte per intero perché richiederebbero una lunga parte di quest'articolo; il loro utilizzo, tuttavia, è molto semplice ed intuitivo.

```
 1. Option Explicit
 2.
 3. Public Enum eMAPIRecipient
 4.     MAPI_TO = 1
 5.     MAPI_CC = 2
 6.     MAPI_BCC = 3
 7. End Enum
 8.
 9. Private Enum eMAPIErrors
10.     SUCCESS_SUCCESS
11.     MAPI_USER_ABORT
12.     ...
36. End Enum
37.
```

La prima delle due enumerazioni specifica il tipo di *Recipient* (ricevente) ammesso dal metodo **AddRecipient** trattato successivamente; l'enumerazione ***eMAPIErrors*** specifica gli errori conosciuti restituiti dalle funzioni MAPI.

```
38. Private Type MAPIRecip
39.     Reserved As Long
40.     RecipClass As Long
41.     Name As String
42.     Address As String
43.     EIDSize As Long
44.     EntryID As String
45. End Type
46.
47. Private Type MAPIFile
48.     Reserved As Long
49.     flags As Long
50.     Position As Long
51.     PathName As String
52.     FileName As String
53.     FileType As Long
54. End Type
```

```
55.
56. Private Type MAPIMessage
57.     Reserved As Long
58.     Subject As String
59.     NoteText As String
60.     MessageType As String
61.     DateReceived As String
62.     ConversationID As String
63.     Originator As Long
64.     flags As Long
65.     RecipCount As Long
66.     Recipients As Long
67.     FileCount As Long
68.     Files As Long
69. End Type
70.
```

I tre tipi di dati UDT saranno utilizzati per specificare, rispettivamente le informazioni sul Recipient, sul file allegato e sul messaggio da inviare. Ad un livello logico in realtà il tipo ***MAPIMessage*** conterrà un riferimento ad un array di recipients (campo **Recipients**) ed ad un array di files (campo **Files**) da allegare.

Prima di procedere è opportuno approfondire il meccanismo con il quale operano le funzioni di messaggistica MAPI: prima di poter accedere al sistema di ricezione, invio o memorizzazione del programma client MAPI è necessario creare o recuperare una sessione, che dovrà essere specificata durante le operazioni di recupero o di invio di messaggi di posta elettronica.

```
71. Private Const MAPI_DIALOG = &H8
72. Private Const MAPI_LOGON_UI = &H1
73. Private Const MAPI_PASSWORD_UI = &H20000
74.
75. Private Declare Function MAPILogon Lib "MAPI32.DLL" (ByVal UIParam As Long, ByVal
    User As String, ByVal Password As String, ByVal flags As Long, ByVal Reserved As
    Long, ByRef Session As Long) As Long
76. Private Declare Function MAPILogoff Lib "MAPI32.DLL" (ByVal Session As Long, ByVal
    UIParam As Long, ByVal flags As Long, ByVal Reserved As Long) As Long
77. Private Declare Function MAPISendMail Lib "MAPI32.DLL" (ByVal Session As Long,
    ByVal UIParam As Long, ByRef message As MAPIMessage, ByVal flags As Long, ByVal
    Reserved As Long) As Long
78.
```

La costante **MAPI_DIALOG** consente di confermare il messaggio prima del suo invio e verrà utilizzata su richiesta dell'utente del nostro esempio. Le due costanti **MAPI_LOGON_UI** e **MAPI_PASSWORD_UI** sono utilizzate per richiedere esplicitamente di mostrare le finestre di dialogo di accesso, se il cliente di posta lo permette e se si rende necessario.

La prima delle funzioni API dichiarate è *MAPILogon* e consente di recuperare un numero di sessione nuovo o preesistente, con il quale accedere al sistema di messaggistica. Nel nostro esempio proveremo a recuperare un ID di sessione esistente e se non esistente, la medesima funzione ne creerà uno nuovo. È comunque possibile richiedere forzatamente la creazione di un nuovo ID di sessione, senza recuperare quello preesistente, specificando il flag **MAPI_NEW_SESSION** (corrispondente al valore 2) nel richiamo della funzione *MAPILogon*.

La funzione *MAPILogoff*, com'è intuibile, effettua l'operazione di disconnessione dal sistema di posta elettronica liberando l'ID di sessione utilizzato. La funzione

MAPISendMail della riga 77 si occuperà quindi di comporre il messaggio di posta elettronica ed eventualmente spedirlo.

```
 79. Private m_Attachments() As MAPIFile
 80. Private m_AttachmentsCount As Long
 81. Private m_Message As MAPIMessage
 82. Private m_Recipients() As MAPIRecip
 83. Private m_RecipientsCount As Long
 84. Private m_SendImmediately As Boolean
 85. Private m_SessionID As Long
 86.
```

La funzione *MAPISendMail* utilizza una variabile di tipo ***MAPIMessage*** di nome **m_Message**; questa utilizzerà anche due array di nome **m_Attachments** e **m_Recipients**, rispettivamente per i files da allegare e per i destinatari del messaggio; le due variabili membro **m_AttachmentsCount** e **m_RecipientsCount** rifletteranno semplicemente il numero di allegati e di destinatari.

La variabile **m_SendImmediately** della riga 84 manterrà il valore della proprietà **SendImmediately**, che determinerà in fase di invio se dovrà essere mostrato il messaggio prima di inviarlo oppure l'invio sarà immediato. L'ultima variabile membro **m_SessionID** conterrà l'ID di sessione recuperato dalla funzione *MAPILogon* ed utilizzato per inviare il messaggio.

```
 87. Public Property Get AttachmentsCount() As Long
 88.     AttachmentsCount = m_AttachmentsCount
 89. End Property
 90.
 91. Public Property Get RecipientsCount() As Long
 92.     RecipientsCount = m_RecipientsCount
 93. End Property
 94.
 95. Public Property Get SendImmediately() As Boolean
 96.     SendImmediately = m_SendImmediately
 97. End Property
 98.
 99. Public Property Let SendImmediately(ByVal newValue As Boolean)
100.     m_SendImmediately = newValue
101. End Property
102.
103. Public Property Get SessionID() As Long
104.     SessionID = m_SessionID
105. End Property
106.
```

Le due proprietà in sola lettura **AttachmentsCount** e **RecipientsCount** restituiranno rispettivamente il numero di allegati e di recipients al momento associati al messaggio di posta. La prprietà **SendImmediately** regolerà il comportamento di invio del messaggio come già spiegato in precedenza. La proprietà **SessionID** restituirà invece l'ID di sessione recuperato dalla funzione *MAPILogon* in occasione del primo messaggio inviato.

```
107. Public Property Get Body() As String
108.     Body = m_Message.NoteText
109. End Property
110.
111. Public Property Let Body(ByVal newValue As String)
112.     m_Message.NoteText = newValue
113. End Property
114.
115. Public Property Get Subject() As String
116.     Subject = m_Message.Subject
```

```
117.  End Property
118.
119.  Public Property Let Subject(ByVal newValue As String)
120.      m_Message.Subject = newValue
121.  End Property
122.
```

Le ultime due proprietà **Body** e **Subject** assegnano e restituiscono il corpo e l'oggetto del messaggio lavorando direttamente sui campi ***NoteText*** e ***Subject*** della variabile **m_Message**.

```
123.  Public Sub AddRecipient(ByVal RecipType As eMAPIRecipient, ByVal Name As String,
      ByVal Address As String)
124.      Dim udtRecip As MAPIRecip
125.      With udtRecip
126.          .RecipClass = RecipType
127.          If Len(Name) > 0 Then .Name = StrConv(Name, vbFromUnicode)
128.          If Len(Address) > 0 Then .Address = StrConv(Address, vbFromUnicode)
129.      End With
130.      ReDim Preserve m_Recipients(m_RecipientsCount)
131.      m_Recipients(m_RecipientsCount) = udtRecip
132.      m_RecipientsCount = m_RecipientsCount + 1
133.  End Sub
134.
```

La routine **AddRecipient** è utilizzata per aggiungere un destinatario al messaggio, composto di nome (***Name***), indirizzo (***Address***) e tipo di recipient (***RecipType***). Sarà utilizzata una variabile di nome **udtRecip** e di tipo ***MAPIRecip***, che al termine dell'operazione verrà aggiunta all'array di destinatari.

Poiché la funzione *MAPISendMail* si aspetta che nome ed indirizzo del destinatario siano forniti in formato ANSI, è necessario convertirli in questo formato mediante la funzione StrConv (righe 127 e 128).

Dalla riga 130 viene aggiunto un elemento alla matrice **m_Recipients**, cui viene assegnata la variabile **udtRecip** appena riempita, e viene incrementato il numero di recipients finora assegnati.

```
135.  Public Sub AddTo(ByVal Name As String, ByVal Address As String)
136.      Call AddRecipient(MAPI_TO, Name, Address)
137.  End Sub
138.
139.  Public Sub AddCC(ByVal Name As String, ByVal Address As String)
140.      Call AddRecipient(MAPI_CC, Name, Address)
141.  End Sub
142.
143.  Public Sub AddBCC(ByVal Name As String, ByVal Address As String)
144.      Call AddRecipient(MAPI_BCC, Name, Address)
145.  End Sub
146.
147.  Public Sub Clear()
148.      Erase m_Recipients
149.      Erase m_Attachments
150.      m_RecipientsCount = 0
151.      m_AttachmentsCount = 0
152.  End Sub
153.
```

I tre metodi **AddTo**, **AddCC** e **AddBCC** dichiarati alle righe 135-145 costituiscono delle mere semplificazioni del metodo **AddRecipient** dichiarato in precedenza; le stesse infatti non fanno altro che richiamare la routine precedente, aggiungendovi soltanto il tipo

di destinatario.

La routine **Clear** azzera semplicemente gli array **m_Recipients** e **m_Attachments** ed i due rispettivi contatori. Va utilizzata in occasione della creazione di un nuovo messaggio.

```
154.  Public Sub FileAdd(ByVal PathName As String)
155.      Dim udtFile As MAPIFile
156.      udtFile.PathName = StrConv(PathName, vbFromUnicode)
157.      ReDim Preserve m_Attachments(m_AttachmentsCount)
158.      m_Attachments(m_AttachmentsCount) = udtFile
159.      m_AttachmentsCount = m_AttachmentsCount + 1
160.  End Sub
161.
```

In maniera analoga alla routine AddRecipient, la **FileAdd** crea una nuova variabile di tipo ***MAPIFile***, vi assegna il nome del file specificato in ***PathName*** dopo averlo convertito dal formato Unicode. La stessa sarà quindi aggiunta alla matrice **m_Attachments** ed il contatore **m_AttachmentsCount** verrà incrementato.

```
162.  Public Function Send() As Long
163.      Dim lngFlags As Long
164.      Send = MAPI_E_INVALID_RECIPS
165.      With m_Message
166.          If m_AttachmentsCount > 0 Then
167.              .FileCount = m_AttachmentsCount
168.              .Files = VarPtr(m_Attachments(0))
169.          End If
170.          If m_RecipientsCount > 0 Then
171.              .RecipCount = m_RecipientsCount
172.              .Recipients = VarPtr(m_Recipients(0))
173.              If m_SessionID = 0 Then Call MAPILogon(ByVal 0&, vbNullChar,
      vbNullChar, MAPI_LOGON_UI Or MAPI_PASSWORD_UI, ByVal 0&, m_SessionID)
174.              If Not m_SendImmediately Then lngFlags = MAPI_DIALOG
175.              Send = MAPISendMail(ByVal 0&, ByVal 0&, m_Message, lngFlags, ByVal 0&)
176.          End If
177.      End With
178.  End Function
179.
```

La funzione **Send** recupera tutti i dati forniti ed invia il messaggio mediante le funzioni MAPI. Il valore di ritorno della funzione è inizializzato su ***MAPI_E_INVALID_RECIPS*** ovvero "Destinatari non validi"; questo perché il messaggio sarà inviato soltanto se esiste almeno un destinatario.

Alle righe 166-169 è verificata la presenza di almeno un file allegato; in tal caso i membri ***FileCount*** e ***Files*** della variabile **m_Message** sono riempiti con il numero di files allegati e con l'indirizzo dell'array contentente l'elenco dei files.

Alle righe 170-176 se esiste almeno un destinatario, viene completo il riempimento della struttura **m_Message** ed il messaggio viene inviato. Il numero di destinatari viene assegnato al membro ***RecipCount***, mentre l'array contenente i recipients è assegnato al membro ***Recipients***.

Alla riga 173, se non esiste già un ID di sessione per l'istanza della classe, viene recuperato mediante la funzione *MAPILogon*. Tale ID è memorizzato nella variabile **m_SessionID** e verrà conservato ed utilizzato dalla funzione *MAPISendMail* successivamente. Anche all'uscita della funzione l'ID di sessione sarà mantenuto, per velocizzare l'eventuale invio di un ulteriore messaggio.

Alla riga 174 è verificato il valore della variabile **m_SendImmediately** e se il suo valore è False, il messaggio non sarà spedito immediatamente ma verrà mostrato all'utente prima di eseguire l'operazione di spedizione. L'opzione è specificata nella variabile **lngFlags** che sarà quindi passata alla funzione successiva.

L'ultima istruzione della funzione richiamerà la funzione *MAPISendMail*, fornendogli la variabile di tipo ***MAPIMessage*** e le opzioni per l'invio, specificati in **m_Message** e **lngFlags**. Il valore di ritorno sarà assegnato come valore di ritorno del metodo Send.

Prima di concludere la visione del codice della classe si vuole specificare che il codice da scaricare per questo articolo conterrà anche altre due routine qui non presentate: si tratta di **NewMail** che in una sola chiamata genera un nuovo messaggio, completo di destinatari multipli, oggetto, corpo ed allegati e provvede anche al suo invio, utilizzando le funzioni sin qui presentate. L'altra funzione non specificata è **MAPIError** che, ricevendo il numero di errore restituito dalle funzioni **Send** e **NewMail** (quindi da *MAPISendMessage*) restituisce una descrizione del tipo di errore. Costituisce in realtà un lungo elenco di assegnazioni condizionali.

La classe si conclude con la routine **Class_Terminate** che provvederà a disconnettersi dal client MAPI e liberare l'ID di sessione:

```
180.  Private Sub Class_Terminate()
181.      If m_SessionID <> 0 Then Call MAPILogoff(m_SessionID, ByVal 0&, ByVal 0&, ByVal
      0&)
182.      m_SessionID = 0
183.  End Sub
```

La disconnessione è effettuata semplicemente richiamando la funzione *MAPILogoff* fornendogli l'ID di sessione utilizzato.

La dimostrazione d'uso della classe *clsMAPISendMail* sin qui affrontata passa per la costruzione di uno spartano client di posta elettronica che consenta la specifica di destinatari, oggetto, corpo e files allegati.

Per necessità di spazio non sarà trattata la disposizione dei controlli sopra il form, ma il solo codice che, fra l'altro, farà uso del modulo **modCommonDialog** trattato nell'[HowTo dedicato alla scelta di un file].

Il codice si compone di tre sole routine legate alla pressione sui pulsanti per l'aggiunta di un destinatario (**-> A:**, **-> Cc:**, **->Ccn:**), di un file allegato (**Aggiungi allegato**) e per l'invio del messaggio composto (**Invia email**):

```
1.  Option Explicit
2.
3.  Private Sub cmdAddRecipient_Click(Index As Integer)
4.      If Len(Trim$(txtNewRecipient.Text)) > 0 Then
5.          lbRecipients.AddItem Choose(Index + 1, "To", "Cc", "Bcc") & ": " &
```

```
    txtNewRecipient.Text
 6.         txtNewRecipient.Text = ""
 7.     End If
 8. End Sub
 9.
```

Se è presente un indirizzo nella casella **txtNewRecipient** esso sarà accodato alla ListBox **lbRecipients**; l'operazione è svolta utilizzando la funzione *Choose* che, lo ricordiamo, è a base 1; pertanto l'indice del pulsante premuto (**Index**) dovrà essere incrementato di 1. Al termine dell'operazione la casella di testo è svuotata (riga 6).

```
10. Private Sub cmdNewAttachment_Click()
11.     Dim strFileName As String
12.     strFileName = ShowOpen(Me, "Tutti i files (*.*)" & vbNullChar & "*.*")
13.     If Len(strFileName) > 0 Then lbAttachments.AddItem strFileName
14. End Sub
15.
```

La pressione del pulsante **cmdNewAttachments** richiamerà la funzione **ShowOpen** del modulo *modCommonDialog* per richiedere il nome di un file da allegare. Se il risultato di quest'operazione è un nome di file (ovvero l'utente non ha premuto il pulsante ***Annulla***) esso sarà aggiunto alla ListBox **lbAttachments**.

```
16. Private Sub cmdSendEmail_Click()
17.     Dim SendMail As clsMAPISendMail
18.     Dim lngLoop As Long
19.     Dim strRecipient As String
20.     Dim lngResult As Long
21.     Set SendMail = New clsMAPISendMail
22.     With SendMail
23.         .Clear
24.         For lngLoop = 0 To lbRecipients.ListCount - 1
25.             strRecipient = lbRecipients.List(lngLoop)
26.             Select Case Left$(strRecipient, InStr(1, strRecipient, ":") - 1)
27.                 Case "To": Call .AddTo("", Mid$(strRecipient, 5))
28.                 Case "Cc": Call .AddCC("", Mid$(strRecipient, 5))
29.                 Case "Bcc": Call .AddBCC("", Mid$(strRecipient, 6))
30.             End Select
31.         Next lngLoop
```

La pressione del pulsante **cmdSendMail** dovrà raccogliere le informazioni da tutti i controlli e formare con esse un nuovo messaggio nell'istanza SendMail allocata alla riga 21.

Il messaggio verrà inizialmente svuotato mediante il metodo **Clear** (anche se non è strettamente necessario) ed in seguito (righe 24-31) dovranno essere recuperati tutti i destinatari, suddividendoli per tipo di recipient, dalla ListBox. Leggendo infatti il prefisso ***To***, ***Cc*** o ***Bcc*** dall'elenco è possibile assegnare l'indirizzo al gruppo corretto, mediante i metodi **AddTo**, **AddCC** e **AddBCC** (righe 27-29).

```
32.         .Subject = txtSubject.Text
33.         .Body = txtBody.Text
34.         For lngLoop = 0 To lbAttachments.ListCount - 1
35.             Call .FileAdd(lbAttachments.List(lngLoop))
36.         Next lngLoop
37.         .SendImmediately = (chkSendNow.Value = vbChecked)
38.         lngResult = .Send
39.         If lngResult > 0 Then
40.             MsgBox .MAPIError(lngResult), vbCritical Or vbOKOnly, "Errore
    nell'invio del messaggio"
41.         Else
```

```
42.             MsgBox "Messaggio inviato con successo", vbInformation Or vbOKOnly,
    "Invio del messaggio riuscito"
43.         End If
44.     End With
45.     Set SendMail = Nothing
46. End Sub
```

Alle righe 32 e 33 sono riempite le proprietà **Subject** e **Body** del messaggio; successivamente saranno recuperati tutti i files dall'elenco **lbAttachments** ed aggiunti al messaggio mediante il metodo **FileAdd**. La CheckBox **chkSendNow** determinerà il valore della proprietà **SendImmediately**.

Raccolte tutte queste informazioni, sarà possibile inviare il messaggio con il metodo **Send**. Il valore restituito sarà quindi discriminante del messaggio mostrato a seguito dell'invio (righe 39-43). Completata l'operazione sarà possibile deallocare la variabile **SendMail**.

L'esecuzione del progetto con la compilazione di tutti i campi, al momento della pressione del pulsante ***Invia email*** richiamerà il client MAPI di posta predefinito (spesso assegnato ad *Outlook Express*).

Nell'esempio mostrato a fianco i destinatari del messaggio sono 3, uno per ciascun tipo di recipient (Diiretto a, Copia carbone e Copia carbone nascosta). Gli stessi sono mostrati sulla destra con i loro rispettivi prefissi To, Cc e Bcc.

In fondo sono mostrati i nomi di due files allegati di cui uno eseguibile (***WC2.EXE***). Il client di posta si occuperà infatti di codificare ove necessario i files in Base64.

La spunta sul flag ***Immediatamente*** farà si che il messaggio venga inviato direttamente senza una necessaria revisione dello stesso. Senza la spunta invece il messaggio apparirà a video e sarà richiesto l'intervento dell'utente (come per l'articolo precedente).

IMPORTANTE!
Il messaggio che apparirà in seguito all'invio del messaggio si riferisce al recapito verso il programma di posta elettronica, che si occuperà per i fatti suoi di inviare il messaggio. Il messaggio restituito infatti non può essere considerato come una conferma dell'avvenuto invio del messaggio, ma piuttosto della consegna a buon fine verso il programma di posta.

Figura 3

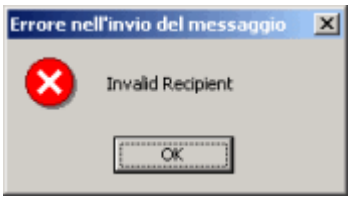

Figura 4

Alcune versioni recenti di Outlook Express

presentano una finestra di conferma simile a quella mostrata a fianco.

Ciò rende più sicuro l'uso del programma client, in modo da evitare che lo stesso venga usato a sproposito e senza che l'utente ne venga a conoscenza.

Sia che l'utente risponda ***Invia*** o ***Non inviare***, la funzione **Send** restituirà comunque il risultato 0 e nessun errore verrà riportato al nostro programma.

Come la precedente, anche questa soluzione necessita di un account di posta correttamente configurato presso il client di posta; inoltre lo stesso client deve rispondere alle specifiche MAPI.

Nonostante le numerose limitazioni di questa soluzione, viste le svariate richieste sull'argomento, abbiamo comunque preferito riportarla ed evidenziarne i punti deboli.

Fibia FBI e Andrea Barchetti
25 Aprile 2003

Torna all'indice degli HowTo