

# **MAPI**

## **GUIDA COMPLETA**

**Utilizzo degli oggetti per la  
gestione della posta elettronica  
in Visual Basic**

**Visual Basic Italia**

## Indice

La struttura di riferimento.....	3
L'oggetto MAPISession.....	4
I metodi di MAPISession.....	6
L'oggetto MAPIMessages .....	8
Recuperare un messaggio di posta elettronica.....	7
Esempio pratico.....	12
Inoltrare un messaggio e rispondere al mittente.....	13
La gestione degli allegati.....	14

## La struttura di riferimento

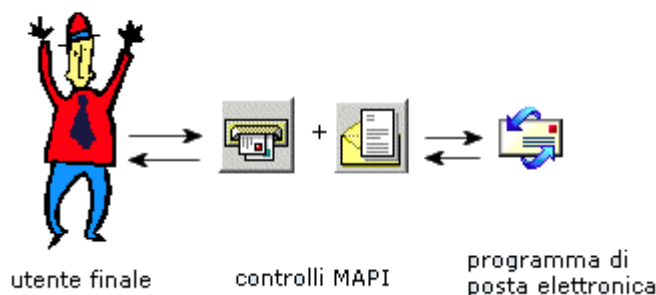
**MAPI** è l'acronimo di **M**essaging **A**pplication **P**rogram **I**nterface, ossia interfaccia per applicazioni di posta elettronica.

Come già il nome indica, i controlli MAPI che costituiscono l'interfaccia citata, permettono di interagire con i dati gestiti da un sistema di posta elettronica conforme MAPI ossia un sistema strutturato in modo da poter essere gestito anche esternamente attraverso i controlli MAPI che in seguito vedremo in dettaglio.

Due di questi sistemi di posta elettronica sono Microsoft Exchange e Microsoft Outlook Express.

Si è appena parlato di interfaccia nel senso di un gruppo di proprietà e metodi che forniscono uno strumento di interazione tra l'utente ed i messaggi di posta elettronica.

Probabilmente la figura sotto servirà a comprendere meglio l'utilità dei controlli MAPI:

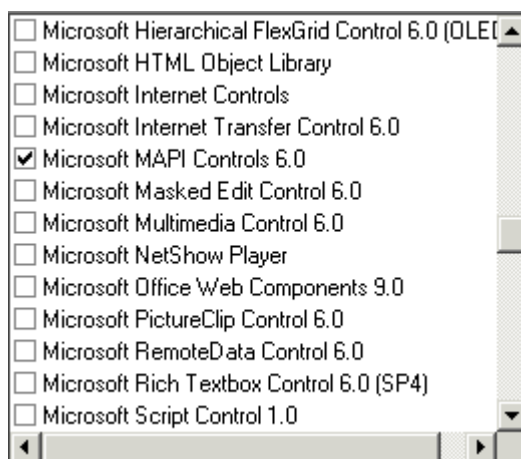


Prima di iniziare l'analisi vera e propria dei controlli MAPI, è necessario accertarsi della presenza del file **MSMAPI32.OCX** nel proprio hard disk (generalmente tale file è contenuto nella cartella "C:\WINDOWS\SYSTEM" per i sistemi Windows 9x e "C:\WINNT\SYSTEM" per i sistemi basati su NT ma per sicurezza è preferibile eseguire una ricerca su tutto il disco fisso).

Una volta trovato il file ed aperto un nuovo progetto EXE Standard in Visual Basic, sarà necessario importare i controlli.

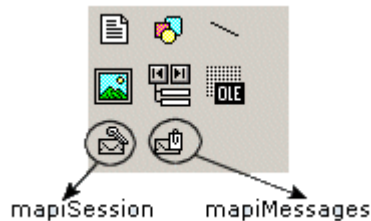
A tale scopo cliccare su Progetto e selezionare la voce componenti dal menu a discesa che apparirà.

Verrà quindi visualizzata la finestra 'Componenti' che dovremo scorrere fino a trovare la voce *Microsoft MAPI Controls x.0*, come in figura:



Selezionare col segno di spunta la casella corrispondente a tale voce e dare l'OK alla finestra.

Una volta importato il gruppo di controlli MAPI, si potrà visualizzare nella lista dei componenti gli oggetti **MapiSession** e **MapiMessages**, come mostrato dalla figura qui sotto:



La creazione di un'interfaccia MAPI può essere quindi suddivisa in due passaggi distinti, in relazione ai controlli MAPI che si andranno ad utilizzare.

Per chiarezza andiamo ad analizzare nel dettaglio il primo due controlli del gruppo MAPI: **MAPISession**.

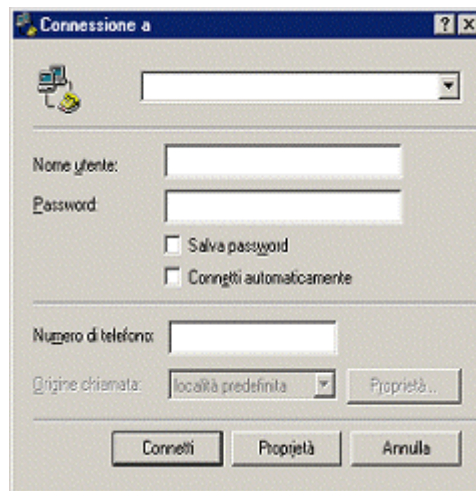
## L'oggetto MAPISession

MAPISession consente di aprire una nuova sessione di posta elettronica.

Con l'apertura di una nuova sessione in realtà non accade niente di così utile per l'utente: si imposta infatti il collegamento col server della posta e si impostano alcuni parametri importanti per definire le regole della sessione: download automatico della posta, username e password dell'account di posta elettronica, apertura di sessioni di posta multiple e conseguente gestione separata di ognuna di esse e così via.

Dunque è necessario prima di tutto impostare l'oggetto sessione MAPI fornendo un nome utente registrato e la corrispondente password validi per accedere all'account di posta.

I controlli MAPI in ogni caso forniscono due vie per eseguire tali operazioni. La prima implica l'interazione diretta tra utente ed applicazione: l'oggetto MAPISession chiederà all'utente di impostare i parametri per la connessione all'account attraverso la tipica finestra mostrata qui sotto:



Sarà quindi l'utente dell'applicazione ad immettere manualmente i dati relativi a username e password. Tale sistema si traduce nel codice che segue:

```
Private Sub Form_Load()
    MAPISession1.SignOn
    ...
    MAPISession1.SignOff
End Sub
```

I metodi SignOn e SignOff verranno analizzati nel prossimo capitolo.

E' ora importante capire che accanto al sistema appena visto (utile nel caso in cui non si conoscano i dati dell'account dell'utente), possiamo trovare il secondo metodo di cui si parlava, ossia l'impostazione dei dati da codice.

Username e password possono così essere inseriti tramite le omonime proprietà UserName e Password del controllo MAPISession.

In questo caso il codice equivalente al blocco appena visto è il seguente:

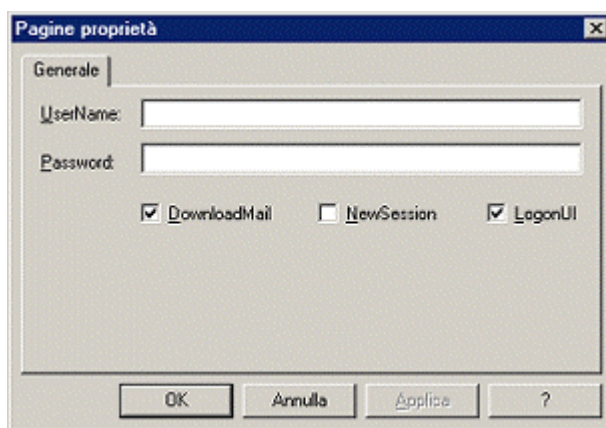
```
Private Sub Form_Load()  
MAPISession1.UserName = "NomeUtente"  
MAPISession1.Password = "Password"  
MAPISession1.SignOn  
...  
MAPISession1.SignOff  
End Sub
```

con l'unica differenza poco fa accennata della necessità o di conoscere direttamente i dati dell'account o di passarli alle proprietà UserName e Password attraverso vie indirette (come le TextBox o altri controlli eventualmente previsti dall'applicazione) o comunque senza il supporto della finestra di connessione mostrata nella figura sopra.

Esiste in realtà una seconda via per l'inizializzazione del controllo MAPISession durante la fase di progettazione, che consiste nell'inserimento dei dati relativi all'account di posta elettronica nella finestra Proprietà del controllo MAPISession.

Per far ciò basta cliccare sul controllo in questione col pulsante destro del mouse e scegliere la voce Proprietà dal popup-menu che compare.

La finestra Proprietà è simile a quella mostrata in figura:



dove Username e Password sono i campi che corrispondono alle proprietà Username e Password dell'oggetto MAPISession proprio come abbiamo visto in precedenza.

La proprietà LogonUI se contrassegnata col segno di spunta consente di visualizzare in ogni caso la finestra di connessione per la richiesta dei dati relativi all'account di posta elettronica.

E' possibile impostare tale proprietà da codice, assegnando a LogonUI il valore True o False a seconda che si desideri visualizzare la finestra o meno:

```
MAPISession1.LogonUI = True(False)
```

Continuando ad analizzare le caratteristiche della finestra mostrata sopra, la proprietà NewSession specifica se deve essere stabilita una nuova sessione di posta elettronica.

Se è già stata stabilita una sessione valida, l'impostazione di questa proprietà consente di eseguire due sessioni contemporaneamente.

Anche per NewSession la proprietà può essere impostata da codice in questo modo:

```
MAPISession1.NewSession = True(False)
```

Infine la proprietà DownloadMail, se spuntata nella casella apposita, consente di scaricare automaticamente i nuovi messaggi all'apertura di una nuova sessione MAPI ossia all'inizializzazione del controllo MAPISession.

Quindi se la proprietà viene impostata su True, tutti i nuovi messaggi verranno inviati automaticamente alla cartella Posta in arrivo dell'utente, in caso contrario cioè all'assegnazione del valore False alla proprietà,

l'utente potrà decidere di scaricare la posta in un secondo momento e di impostare la frequenza con cui la posta viene scaricata dal server della posta.

Anche in questo caso la proprietà è impostabile da codice in questo modo:

```
MAPISession1.DownloadMail = True(False)
```

## I metodi di MapiSession

Come premesso in questo capitolo si andrà ad analizzare i metodi supportati dal controllo MapiSession, metodi che sono stati già introdotti negli esempi del capitolo precedente.

I metodi in questione sono rispettivamente SignOn e SignOff ed hanno lo scopo specifico di attivare e disattivare la sessione MAPI.

Quindi il procedimento per l'utilizzo dei controlli MAPI, in relazione all'oggetto di base ossia MapiSession può essere descritta dalla figura sotto:



Vediamoli uno per uno: il metodo SignOn come già detto, apre la sessione MAPI dopo aver recuperato tutte le informazioni sulla connessione all'account di posta con uno dei metodi che abbiamo già visto in precedenza (inserimento dei parametri come username e password da parte dell'utente, inserimento nel codice, inserimento nella finestra Proprietà dell'oggetto MapiSession).

Come già è stato accennato la sintassi relativa all'attivazione della sessione MAPI è la seguente:

```
MAPISession1.SignOn
```

dove MapiSession1 è il nome assegnato come default al controllo MapiSession.

Cosa accade quando viene aperta una nuova sessione MAPI? Fondamentalmente quello che succede quando viene creata una nuova finestra: il sistema operativo assegna ad essa un indicatore univoco che risulterà necessario per individuarla tra più sessioni aperte.

L'identificatore che corrisponde ad un valore numerico, viene quindi memorizzato nella proprietà SessionID, che dovrà essere richiamata nel caso esistano più sessioni aperte e si desidera passare da una all'altra.

Il valore predefinito di SessionID corrisponde a 0.

Naturalmente questo rappresenta un valore di sola lettura: non è possibile impostare manualmente l'identificatore di una sessione MAPI ne in fase di progettazione ne in fase di avvio perchè questo è un lavoro che spetta al sistema operativo.

Una volta visto il metodo SignOn preoccupiamoci del metodo SignOff che come accennato chiude una connessione MAPI.

La sintassi è la seguente:

```
MAPISession1.SignOff
```

Chiudendo la sessione MAPI non è più possibile accedere al controllo MAPIMessages allo stesso modo in cui, chiudendo il programma Excel, non è più possibile accedere ai singoli fogli di calcolo.

Chiusa la sessione anche la proprietà SessionID ed i vari parametri di inizializzazione come LogonUI, Password, DownloadMail vengono svuotati dei valori precedentemente assegnatole. Sarà quindi necessario reimpostare tutta la serie di informazioni da passare ad un'eventuale nuova sessione.

## L'oggetto MAPIMessages

Adesso che è stata terminata la descrizione generale del controllo MAPISession si può analizzare in dettaglio il controllo successivo ossia **MAPIMessages** che è in poche parole il cuore di ogni sessione MAPI. Senza MAPIMessages infatti non sarebbe possibile intervenire e gestire la posta elettronica e quindi anche l'apertura di una nuova sessione risulterebbe del tutto inutile.

Il controllo in questione infatti mette a disposizione una serie di metodi e proprietà utili per creare il proprio programma di posta elettronica con tutte le funzioni di invio, ricezione e lettura dei messaggi.

La prima cosa da fare per inizializzare un controllo MAPIMessages è quella di associare il controllo ad una sessione MAPI esistente.

Questo perchè basta un unico controllo *MAPIMessages* (proprio com'è sufficiente un solo controllo MAPISession) nell'applicazione per gestire più sessioni MAPI.

Per questa ragione però bisogna indicare chiaramente su quale sessione far lavorare il controllo MAPIMessages.

Torna quindi utile la proprietà SessionID che identifica ognuna delle sessioni aperte. Per associare il controllo MAPIMessages ad una sessione:

```
MAPIMessages1.SessionID = MAPISession1.SessionID
```

Così per passare ad una nuova sessione MAPI e assegnarla al controllo MAPIMessages:

```
MAPISession1.NewSession = True  
MAPIMessages1.SessionID = MAPISession1.SessionID
```

Come fare per accedere ai singoli messaggi di posta elettronica?

Innanzitutto quando si connette il controllo MAPIMessages ad una connessione MAPI si ha accesso ad una partizione della memoria che archivia i messaggi ricevuti.

L'interno di questo archivio viene assegnato a ciascun messaggio un indice che lo identifica in maniera univoca.

Così il primo messaggio della lista ha indice 0, il secondo 1 e così via fino all'ennesimo messaggio che è dotato di indice n-1. Si può pensare a quest'archivio come una ListBox, siccome anch'essa ha una struttura simile in fatto di indicizzazione dei singoli elementi. Così come in una ListBox l'indice che individua ogni singolo membro è dato dalla proprietà ListIndex, per quanto riguarda il controllo MAPISession l'indice è dato dalla proprietà MsgIndex. Si vedrà tra poco come la proprietà risulterà molto utile.

Passando adesso a qualcosa di immediatamente utile, possiamo analizzare la proprietà MsgCount che come la proprietà ListCount del controllo ListBox, indica quanti messaggi sono presenti nel gruppo della posta in arrivo (quindi messaggi nuovi più messaggi già ricevuti).

Così risulta semplice, ad esempio all'avvio di un'applicazione, andare a recuperare il numero dei messaggi di posta elettronica presenti nella casella. Proviamo a visualizzare tale numero in una finestra di messaggio.

```
Private Sub Form_Load()  
MAPISession1.DownloadMail = True  
MAPISession1.Fetch  
MAPISession1.SignOn  
MAPIMessages1.SessionID = MAPISession1.SessionID  
MsgBox MAPIMessages1.MsgCount & " messaggi"  
MAPISession1.SignOff  
End Sub
```

E' da notare che la prima linea del codice, dopo l'indicazione della sottoprocedura Form\_Load indica, come abbiamo visto nel capitolo precedente, che i messaggi verranno scaricati all'avvio dell'applicazione.

Nulla però vieta di eliminare questo passaggio indicando in questo caso:



```
MAPISession1.DownloadMail = False
```

La seconda riga invece fa uso della proprietà *Fetch*. Tale proprietà ha lo scopo di creare gruppi di messaggi utilizzando i messaggi selezionati nella cartella della posta in arrivo.

Ognuno di questi gruppi di messaggi è creato in base a proprietà comuni indicate da *FetchMsgType*.

Non aver indicato la proprietà *FetchMsgType* prima di includere *Fetch* nel codice significa aver creato un unico gruppo di messaggi che corrisponde a quello che sui server di posta elettronica viene definita la casella "InBox" oppure "Posta in arrivo".

Inoltre, siccome nel codice non c'interessa gestire messaggi diversi da quelli standard, abbiamo potuto sorvolare sulla proprietà *FetchMsgType*.

Adesso che è più o meno stata chiarita la struttura del controllo *MAPIMessages* e le regole per la gestione della posta in arrivo, possiamo dedicarci nel corso del prossimo capitolo al recupero delle informazioni contenute nei singoli messaggi, come oggetto, corpo del messaggio, destinatario, mittente, data etc, utilizzando la proprietà *MsgIndex* di cui si parlava prima.

## Recuperare un messaggio di posta elettronica

Adesso che sono stati recuperati tutti i messaggi della posta in arrivo, sappiamo come estrarre un messaggio particolare: con la proprietà *MsgIndex*.

Una volta recuperato un messaggio particolare (ad esempio il 50° su 100 messaggi nella casella della posta in arrivo), sono possibili una serie di operazioni per dare una forma allo stesso.

La prima cosa è recuperarne l'oggetto. Questo consente di trattare ogni singolo messaggio non più come un semplice numero (l'indice del messaggio rappresentato dalla proprietà *MsgIndex*, ma in base alle informazioni in esso contenuto.

Possiamo pensare infatti di includere nel progetto un controllo *ListBox* nel quale inserire tutti gli oggetti dei messaggi.

Per fare ciò dovremo utilizzare la proprietà *MsgSubject* del controllo *MAPIMessages* che individua appunto l'oggetto del messaggio.

Quindi, per ottenere il risultato di cui si parlava poco sopra:

```
For i = 0 To MAPIMessages1.MsgCount - 1
    MAPIMessages1.MsgIndex = i
    List1.AddItem MAPIMessages1.MsgSubject
Next I
```

E' importante comprendere che l'elemento che varia è l'indice del messaggio. In base ad esso, cioè al messaggio correntemente considerato, l'oggetto sarà diverso.

Altra cosa da considerare, ma che in parte avevamo già trattato nei capitoli precedenti è la corrispondenza degli indici del controllo *MAPIMessages* e *ListBox*.

Il primo elemento elencato in essi ha indice 0, quindi è comprensibile che l'n-esimo elemento abbia un indice pari a n-1.

Ciò che abbiamo visto per l'oggetto dei messaggi può essere facilmente applicato anche ai mittenti dei messaggi della posta.

Se vogliamo elencare il mittente di ogni messaggio nella *ListBox*, basterà sostituire alla proprietà *MsgSubject* la proprietà *MsgOrigDisplayName*, nel seguente modo:

```
For i = 0 To MAPIMessages1.MsgCount - 1
    MAPIMessages1.MsgIndex = i
    List1.AddItem MAPIMessages1.MsgOrigDisplayName
Next I
```

Naturalmente i blocchi di codice visti sopra hanno un significato (e quindi non restituiscono un messaggio d'errore) solamente se i messaggi nella casella della posta in arrivo sono uno o più.

Per questo si può ipotizzare la necessità di un controllo preliminare della presenza di almeno un messaggio. Il codice qua sotto ha proprio tale scopo:

```
If MAPIMessages1.MsgCount > 0 Then
    ' ...
End If
```



Viene cioè utilizzata la proprietà `MsgCount` per il conto totale dei messaggi. Tale proprietà era già stata analizzata nel capitolo precedente.

Unitamente alle proprietà finora viste, nel campo della gestione dei messaggi in lettura, esistono una serie di altre proprietà utili che permettono in pochi semplici passaggi di ricreare un'applicazione di gestione della posta elettronica.

La prima proprietà degna di nota è `MsgDateReceived` che non va molto oltre quelle appena analizzate: semplicemente recupera le informazioni relative alla data di arrivo del messaggio.

La proprietà è richiamabile nel seguente modo:

```
MAPIMessages1.MsgDateReceived
```

Altra proprietà è `MsgOrigDisplayName` che recupera invece le informazioni relative al mittente che ha inviato la lettera.

Per richiamare tale proprietà è sufficiente la seguente linea di codice:

```
MAPIMessages1.MsgOrigDisplayName
```

Andando oltre troviamo la possibilità di determinare se in precedenza il messaggio è già stato letto o meno, attraverso la proprietà `MsgRead` che restituisce un valore di tipo booleano: `True` in caso affermativo e `False` se il messaggio non è mai stato aperto.

Ecco come utilizzare la proprietà:

```
MAPIMessages1.MsgRead
```

Ogni utente di un account di posta elettronica ha inoltre la facoltà di richiedere una ricevuta di ritorno del messaggio inviato.

Per determinare se il mittente si è avvalso di tale opportunità si usa la proprietà `MsgReceiptRequested`. In pratica la proprietà si richiama in questo modo:

```
MAPIMessages1.MsgReceiptRequested
```

Inoltre esiste la proprietà `MsgOrigAddress` che determina l'indirizzo di provenienza del messaggio ossia l'indirizzo del mittente:

```
MAPIMessages1.MsgOrigAddress
```

e la possibilità di recuperare la stringa d'identificazione assegnata al messaggio correntemente considerato. Questo è possibile attraverso la proprietà `MsgID`:

```
MAPIMessages1.MsgID
```

In più si può determinare il valore di identificazione del thread di conversazione relativo al messaggio indicizzato corrente attraverso la proprietà `MsgConversationID`:

```
MAPIMessages1.MsgConversationID
```

Infine le ultime due proprietà: la prima, `MsgSent`, consente di determinare se il messaggio indicizzato corrente è già stato inviato al server della posta:

```
MAPIMessages1.MsgSent
```

mentre `MsgType` specifica il tipo di messaggio indicizzato:

```
MAPIMessages1.MsgType
```

Per quanto riguarda i tipi di messaggi, nel capitolo precedente abbiamo già visto che non è particolarmente frequente la gestione di messaggi diversi da quelli standard, motivo per il quale avevamo potuto sorvolare sulla proprietà `FetchMsgType`.

Le proprietà che caratterizzano i controlli MAPI sono certo più di quelle elencate nel capitolo precedente. In realtà però per creare una semplice applicazione di invio e ricezione della posta quelle già analizzate sono più che sufficienti.

E dunque all'interno di questo capitolo se ne vedranno poche altre, insieme a tutti i metodi necessari per la gestione della posta in uscita.

Il che vuol dire composizione dei messaggi ed invio degli stessi ossia il fine ultimo dell'utilizzo del controllo MAPIMessages.

Prima di scrivere un messaggio però bisogna tenere conto di una scelta preliminare: si intende lavorare su un messaggio esistente (ossia rispondere al messaggio, copiarne il testo o l'allegato e così via) oppure si desidera comporne uno nuovo?

La differenza è sostanziale perchè nel primo caso è necessario chiamare nuovamente in causa la proprietà `MsgIndex` che come già abbiamo visto rivela l'indice di ogni singolo messaggio ed ha quindi la funzione di registro di tutti i messaggi. Nel secondo caso al contrario, l'apertura di un nuovo messaggio non comporterà alcuna scelta di messaggi.

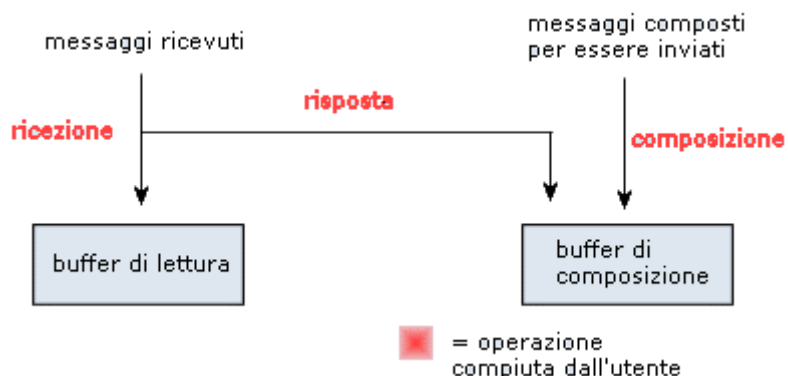
In seguito si vedrà perchè.

Cominciamo a studiare il secondo caso.

Come già visto esistono due partizioni della memoria: il buffer di lettura, dove sono contenuti tutti i messaggi in lettura ed il buffer di composizione dove vengono raccolti tutti i messaggi creati per essere inviati.

In realtà anche i messaggi in lettura possono andare nel buffer di composizione nel momento in cui si risponde (composizione) ad un messaggio ricevuto (lettura).

Ma questo rappresenta il primo caso dei due visti sopra perciò verrà trattato poco più avanti.



Quando si crea un nuovo messaggio per la composizione la proprietà `MsgIndex` che è un valore di tipo Long di intervallo compreso tra -1 e `MsgCount-1`, passa automaticamente al valore -1.

Tale valore indica che quello indicizzato (ossia quello correntemente considerato) è un messaggio in composizione per essere inviato.

Quindi, il metodo principale per creare un nuovo messaggio è `Compose` che viene richiamato nel seguente modo:

```
MAPIMessages1.Compose
```

Perciò se volessimo riassumere quanto visto finora in un codice che apra una nuova sessione MAPI, controlli la posta in arrivo ed apra un nuovo messaggio:

```
Private Sub Form_Load()  
MAPISession1.DownLoadMail = True  
MAPIMessages1.Fetch  
MAPISession1.SignOn  
With MAPIMessages1  
    .SessionID = MAPISession1.SessionID  
    .Compose  
    MsgBox .MsgIndex  
End With
```

```
MAPISession1.SignOff  
End Sub
```

come si può notare è stata inserita anche una riga di codice che, attraverso una finestra di messaggio, dimostra che l'indice del nuovo messaggio in composizione è proprio -1.

L'invio del nuovo messaggio è però un procedimento che necessita di alcuni passaggi ulteriori. Il primo consiste nell'impostare l'indirizzo del destinatario.

Questo può essere fatto agevolmente attraverso la proprietà RecipAddress, in questo modo:

```
MAPIMessages1.RecipAddress
```

Inoltre è possibile inserire anche il nome del destinatario, che si sovrappone all'indirizzo di posta elettronica.

Ad esempio se si considera l'indirizzo mariorossi@provider.it ed il nome Mario Rossi, è possibile sovrapporre nome ed indirizzo in questo modo: MarioRossi nel formato MarioRossi

Il nome che corrisponde all'indirizzo indicato è impostabile attraverso la proprietà RecipDisplayName:

```
MAPIMessages1.RecipDisplayName
```

quindi riassumendo per comporre un nuovo messaggio da scrivere al signor Mario Rossi:

```
MAPIMessages1.RecipDisplayName = "Mario Rossi"  
MAPIMessages1.RecipAddress = mariorossi@provider.it
```

Poco prima dell'invio del messaggio è però possibile eseguire un controllo sulla validità dell'indirizzo.

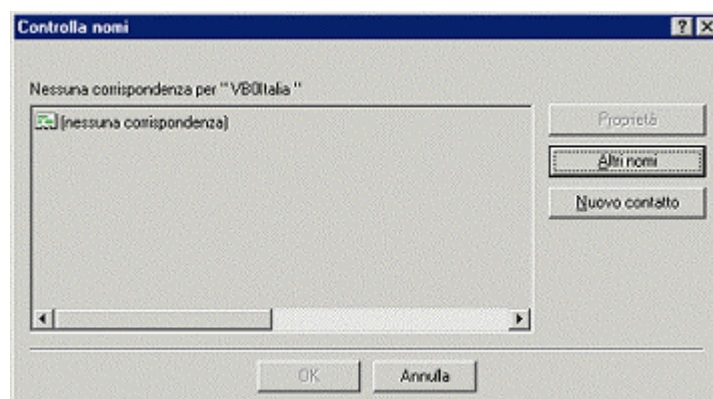
Questa operazione viene gestita dall'applicazione di posta elettronica attraverso il metodo ResolveName che cerca nella rubrica se esiste in memoria il nome indicato.

Parallelamente, attraverso la proprietà AddressResolveUI si può mostrare una finestra di messaggio che mostri i dettagli dell'operazione (se la proprietà è impostata su True) e dove viene suggerito l'eventuale indirizzo alternativo simile a quello specificato.

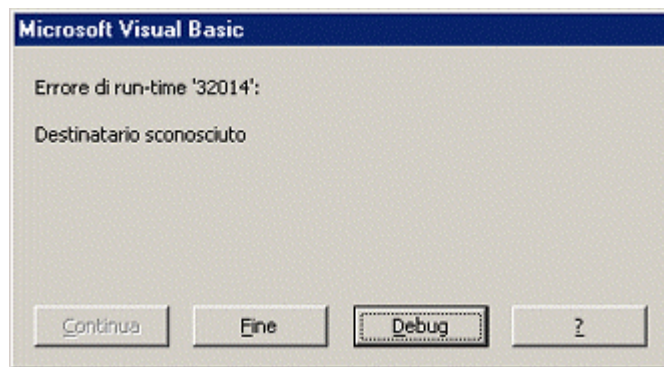
Nel caso in cui AddressResolveUI sia False o non sia contemplata nel codice, viene visualizzato un messaggio di errore.

Naturalmente se si desidera attivare la finestra dei dettagli bisogna richiamare la proprietà AddressResolveUI prima del metodo ResolveName.

La finestra visualizzata cambia naturalmente in base all'applicazione che si utilizza. Ecco un esempio:



insieme a tale finestra, nel caso in cui si sia tralasciata la gestione degli eventuali errori si otterrà un errore di esecuzione della seguente tipologia:



```
MAPIMessages1.MsgOrigDisplayName
```

Adesso che è stato correttamente impostato l'indirizzo del destinatario, si può pensare finalmente al corpo del messaggio e all'oggetto.

Quest'ultimo viene definito molto semplicemente dalla proprietà MsgSubject che però è limitata a 64 caratteri.

Eccone un possibile utilizzo:

```
MAPIMessages1.MsgSubject = "Oggetto della lettera da inviare"
```

Per quanto riguarda invece il testo vero e proprio del messaggio è necessario utilizzare la proprietà MsgNoteText.

Ad esempio:

```
MAPIMessages1.MsgNoteText = "Testo della lettera da inviare." & vbCrLf & _  
"Per le regole di composizione del testo riferirsi" & vbCrLf & _  
"alle normali regole di inserimento di un testo in una " & vbCrLf & _  
"casella di testo."
```

Infine, terminate queste operazioni preliminari, si può inviare il messaggio.

Il metodo da utilizzare è in questo caso Send, che necessita di un parametro di tipo booleano: True se l'invio necessita dell'intervento dell'utente (viene infatti visualizzata una finestra di dialogo di composizione dei messaggi del sistema di posta elettronica) e False se l'invio è automatico.

Esempio:

```
MAPIMessages1.Send
```

## Esempio pratico

Nell'ultimo capitolo abbiamo visto come comporre un messaggio.

Ricapitoliamo tutti i concetti in un codice che invii una lettera ad un destinatario ipotetico (che per avere un ritorno e per evitare di infastidire altri utenti) potrebbe essere il proprio indirizzo di posta elettronica ossia in pratica quello stesso del mittente.

```
Private Sub Form_Load()  
MAPISession1.DownLoadMail = False  
MAPISession1.SignOn  
With MAPIMessages1  
.SessionID = MAPISession1.SessionID  
.Compose
```

```

.RecipAddress = "destinatario@email.it"
.RecipDisplayName = "Nome_Destinatario"
.MsgSubject = "Oggetto della lettera da inviare"
.MsgNoteText = "Testo della lettera da inviare." & vbCrLf & _
"Per le regole di composizione del testo riferirsi" & vbCrLf & _
"alle normali regole di inserimento di un testo in una " & vbCrLf & _
"casella di testo."
.Send
End With
MAPISession1.SignOff
End Sub

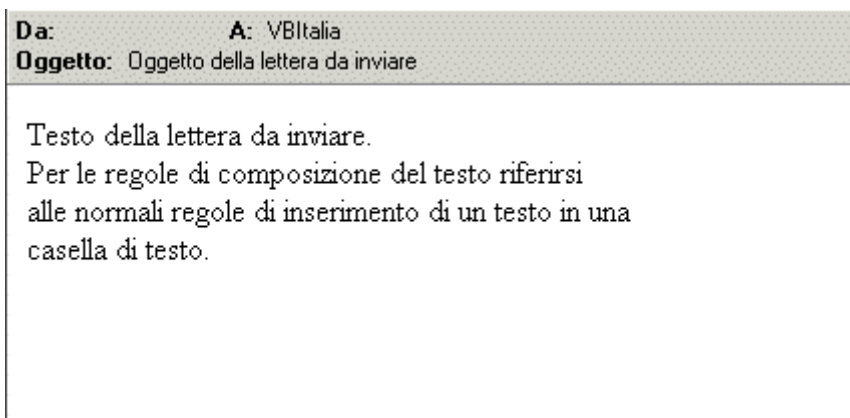
```

Dal codice visto qui sopra si ricavano alcune semplici considerazioni. Prima di tutto che il progetto funziona correttamente.

Basterà aprire il proprio programma di posta elettronica per ricevere immediatamente la lettera (ammesso che il destinatario non sia un terzo, cosa per la verità molto sconsigliata).

Seconda considerazione, non meno importante, che ricaviamo è la determinazione dell'indirizzo del mittente. Non è stato impostato infatti nel codice l'indirizzo dal quale mandare la lettera ma solo quello del destinatario.

In realtà questo dipende dall'indirizzo predefinito che si è impostato nel programma di posta elettronica utilizzato. Se c'è la possibilità di impostare account multipli quindi quello predefinito sarà l'indirizzo mittente. Ecco la dimostrazione della correttezza del codice: viene riportata sotto la lettera ricevuta all'indirizzo mariorossi@provider.it:



Notare che il campo 'Da:' è vuoto. In realtà l'indirizzo è presente nel messaggio originale ma essendo stato inviato da un account personale è stato cancellato.

Arrivati a questo punto della trattazione, la maggior parte del lavoro è stata fatta: siamo riusciti a recuperare i messaggi nella posta in arrivo e ad inviarne dei nuovi.

Adesso rimangono solamente alcuni particolari, di minore rilievo rispetto ai due appena elencati ma che rendono gli oggetti MAPI ancora più completi.

## Inoltrare un messaggio e rispondere al mittente

Ad esempio la possibilità di inoltrare il messaggio ricevuto ad un altro indirizzo di posta elettronica, sostituendo quindi al metodo Compose il metodo Forward.

Il metodo aggiungerà una "I:" alla riga dell'oggetto per cui nell'esempio visto prima si riceverà un messaggio con oggetto "I: Oggetto della lettera da inviare".

Simile al metodo Forward c'è il metodo Reply ed il metodo ReplyAll (l'unica differenza è che il secondo permette di rispondere a tutti i destinatari indicati nel messaggio) che consente di rispondere ad un messaggio ricevuto e quindi contenuto nella cartella 'Posta in arrivo'.

Questa volta però alla riga dell'oggetto verrà aggiunto "R:".

Naturalmente l'uso di questi due metodi necessita in ogni caso di inviare il messaggio, proprio come si fa per un messaggio di nuova creazione.

Per questo è necessario usare anche in questi casi il metodo Send dopo aver impostato il testo del messaggio e qualsiasi altro eventuale parametro (indirizzi dei destinatari, oggetto e così via).

Save invece salva il messaggio correntemente indicizzato nel buffer di scrittura, mentre Copycopia il messaggio corrente contenuto nel buffer di scrittura.

Terminati questi metodi possiamo passare all'ultimo argomento di questo speciale dedicato agli oggetti MAPI analizzando la gestione degli allegati.

## La gestione degli allegati

Concludiamo questa panoramica sugli oggetti MAPI con la gestione degli allegati ossia come inserire file di qualsiasi genere a messaggi in uscita e come ricevere e archiviare allegati dai messaggi nella casella posta in arrivo.

Per quanto riguarda questo secondo punto è innanzitutto necessario indicare l'indice del messaggio del quale si desidera ottenere l'allegato.

Ad esempio per ottenere il nome di tutti gli allegati di tutti i messaggi arrivati bisogna indicare la proprietà AttachmentCount che conta il totale di allegati per messaggio e la proprietà AttachmentName che ottiene invece il nome dell'allegato. Estendendo questo processo a tutti i messaggi della posta in arrivo:

```
For i = 0 To MAPIMessages1.MsgCount - 1
MAPIMessages1.MsgIndex = i
For ii = 0 To MAPIMessages1.AttachmentCount - 1
List1.AddItem MAPIMessages1.AttachmentName
Next ii
Next I
```

E' inoltre possibile determinare un singolo allegato attraverso la proprietà AttachmentIndex. Naturalmente tale proprietà dev'essere compresa in un intervallo con valore minimo 0 e valore massimo AttachmentCount - 1. Ad esempio se volessimo visualizzare in un controllo ListBox il nome del secondo allegato del quindicesimo messaggio di posta in arrivo:

```
MAPIMessages1.MsgIndex          = 15
MAPIMessages1.AttachmentIndex    = 2
List1.AddItem MAPIMessages1.AttachmentName
```

Si può inoltre utilizzare la proprietà AttachmentPathName per determinare la provenienza dell'allegato ossia il percorso sul disco fisso dal quale è stato prelevato il file. Naturalmente tale proprietà è disponibile solamente nel processo di inserimento di allegati nelle lettere da comporre. Attraverso la proprietà AttachmentPathName quindi è materialmente possibile inserire un allegato. Nell'esempio di composizione di lettera fatto nel capitolo precedente possiamo inserire un allegato preso dal percorso "C:\Documenti\allegato.txt":

```
Private Sub Form_Load()
MAPISession1.DownLoadMail = False
MAPISession1.SignOn
With MAPIMessages1
.SessionID = MAPISession1.SessionID
.Compose
.RecipAddress = "destinatario@email.it"
.RecipDisplayName = "Nome_Destinatario"
.MsgSubject = "Oggetto della lettera da inviare"
.MsgNoteText = "Testo della lettera da inviare." & vbCrLf & _
"Per le regole di composizione del testo riferirsi" & vbCrLf & _
"alle normali regole di inserimento di un testo in una " & vbCrLf & _
"casella di testo."
.AttachmentPathName = "C:\Documenti\allegato.txt"
.AttachmentName = "allegato"
```

```
.Send  
End With  
MAPISession1.SignOff  
End Sub
```

E' inoltre possibile indicare il nome dell'allegato come abbiamo fatto nell'esempio, nel caso in cui si desideri indicare un nome differente da quello originale del file.

Nel caso non sia specificato alcun nome, l'allegato sarà denominato con il nome originale del file.  
Infine l'ultimo elemento da considerare è la posizione dell'allegato nel corpo del messaggio. Questa posizione si può determinare attraverso la proprietà AttachmentPosition. Con il valore predefinito 0 l'allegato viene posizionato all'inizio del messaggio mentre per posizionarlo alla fine, è necessario contare il numero di caratteri del corpo del messaggio. Se ad esempio il messaggio è composto da cinque caratteri, è necessario impostare il valore su 4 in modo da indicare che il corpo del messaggio occupa le posizioni di carattere da 0 a 4.