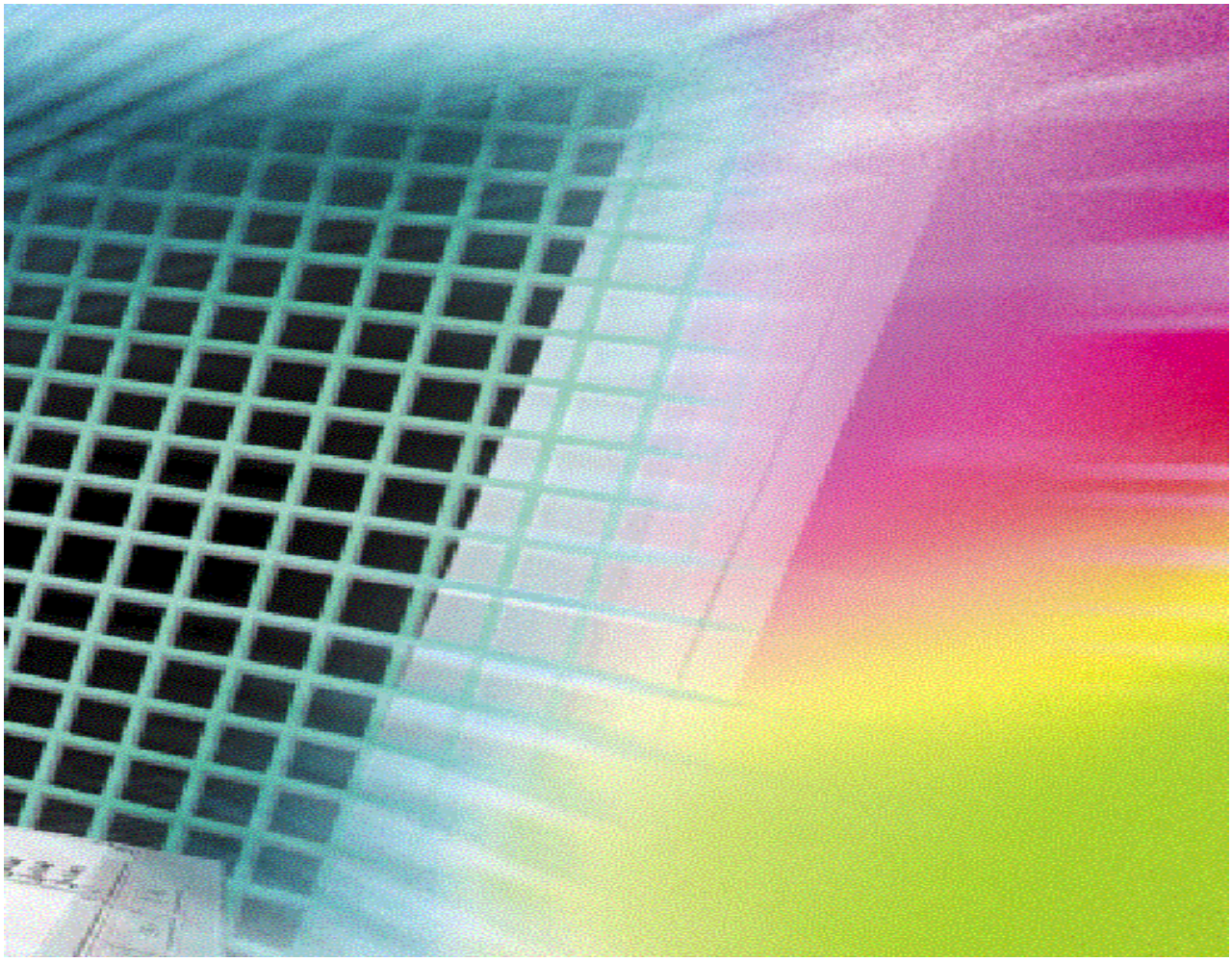
MASSIMO UBERTINI



VISUAL BASIC

WWW.UBERTINI.IT

CREAZIONI GUIDATE

INTRODUZIONE

La Creazione guidata applicazioni di Visual BASIC aiuta a produrre applicazioni senza scrivere codice o aggiungere controlli ai form, si limita a creare una **shell** o **struttura di applicazione** con alcune caratteristiche generali: consiste di una sequenza di finestre di dialogo in cui è possibile selezionare opzioni tipiche delle applicazioni Windows, come il numero di finestre del form, i menu e altri comuni elementi dell'interfaccia, di solito è di maggiore vantaggio per i programmatori Visual BASIC avanzati che non per i principianti. Per aggiungere codice e controlli alla shell di un'applicazione Windows occorre, infatti, riuscire a capire il contenuto della shell. Per generate la prima applicazione con la creazione guidata applicazioni si procede come segue.

1. **File/Nuovo progetto.**
2. Fare doppio clic sull'icona **Creazione guidata applicazioni VB**.
3. Leggere il testo nella prima finestra di dialogo della creazione guidata. Un profilo è un elenco salvato d'impostazioni, che è possibile specificare e registrare, le quali determineranno la natura dell'applicazione che si creerà. Passare oltre la finestra di dialogo, facendo clic su *Avanti* senza cambiare alcuna opzione.
4. Nella seconda finestra di dialogo della creazione guidata si deve specificare se l'applicazione avrà un'interfaccia a documenti multipli, **MDI** (*Multiple Document Interface*: un'applicazione MDI è un'applicazione Windows da cui è possibile aprire più finestre di dati, come un elaboratore di testi nel quale è possibile avere più documenti aperti per la modifica contemporaneamente), oppure a documento singolo, **SDI** (*Single Document Interface*: un'applicazione SDI è una singola finestra del form), e ancora se si tratterà di un'applicazione del tipo di Gestione risorse di Windows: mostra una struttura di opzioni di alto livello a sinistra sullo schermo, e informazioni più dettagliate a destra. Fare clic sui pulsanti di opzione al centro dello schermo, l'immagine visualizzata nella finestra di dialogo cambia per mostrare il tipo di applicazione selezionata. Fare clic sull'opzione Interfaccia a documento singolo (SDI). Immettere *Primo* come nome per il progetto e fate clic su *Avanti* per passare alla successiva finestra di dialogo della creazione guidata.
5. La finestra di dialogo *Menu*, permette di creare un menu a discesa tipico di Windows e aggiungerlo all'applicazione. Anche se è possibile aggiungere menu tramite la casella degli strumenti e la programmazione, la creazione guidata rappresenta il modo più semplice per farlo. Accettare tutte le impostazioni predefinite, in modo che la creazione guidata aggiunga i menu seguenti all'applicazione: *File, Modifica, Visualizza e ?,* insieme ai comandi di menu standard. Fare clic su *Avanti* per continuare.
6. La finestra di dialogo successiva permette di selezionare una barra degli strumenti Windows che apparirà lungo il lato superiore della finestra dell'applicazione sotto i menu. Accettare le impostazioni predefinite e fare clic su *Avanti*.
7. I *file di risorse* contengono immagini bitmap, testo e altri elementi di una tipica applicazione Windows. Anche se è possibile incorporare risorse direttamente nell'applicazione tramite vari metodi, posizionandole all'esterno di un'applicazione in un file di risorse, è possibile cambiarle più facilmente senza dover modificare l'applicazione. Usare un file di risorse per il testo dell'applicazione. Ogni valore di testo che appare sul form dell'applicazione può risiedere all'interno di un file di risorse. Se quindi si volesse vendere l'applicazione in un altro paese, basta tradurre e fornire un nuovo file di risorse senza cambiare nessuna riga di codice. Se invece s'incorpora il testo nel codice, per tradurlo si deve trovarlo, modificarlo e ricompilare ogni file che includa qualsiasi testo. Non selezionare alcun file di risorse. Fare clic su *Avanti* per passare alla successiva finestra di dialogo.

8. Nella finestra di dialogo *Connessione ad Internet* fate clic su *Sì*. La creazione guidata applicazioni fornisce l'accesso ad Internet. Gli utenti devono già avere una connessione per poter accedere al web con l'applicazione. Inoltre nel PC deve essere installato Internet Explorer perché contiene codice condiviso che la creazione guidata utilizza per aggiungere il codice appropriato. Quando un utente attiva il browser all'interno dell'applicazione, il browser si connette alla rete tramite l'ISP e accede all'**URL** che si è specificato nella creazione guidata. La creazione guidata fornisce un URL predefinito, ovvero la home page di Microsoft, quindi si deve cambiare quest'impostazione se si desidera che gli utenti vedano qualcos'altro. Fare sempre iniziare l'URL con **http:*//***.
9. Fare clic sulle opzioni appropriate della creazione guidata per aggiungere gli elementi desiderati all'applicazione: una *schermata iniziale* (una finestra che appare quando l'applicazione si avvia) e una *finestra Informazioni su* (una finestra di dialogo che elenca i dettagli e le informazioni di contatto per l'applicazione, cui si può accedere dal menu *?*). Fare clic su *Avanti* per continuare nella creazione guidata.
10. Fare clic su *Crea nuovo form*…, crea in modo semplice form legati ai dati che forniscono all'utente un accesso completo ai dati del database dell'applicazione, funziona solo con il controllo Data Control **ADO** *(ActiveX Data Objects)*.
11. La finestra di dialogo visualizzata, *Introduzione*, permette di selezionare un profilo. Fare clic su *Avanti* per continuare.
12. Selezionare il tipo di database: Access o Remote **ODBC** *(Open DataBase Connectivity).* Selezionare Access e fare clic su *Avanti* per continuare.
13. Specificare la posizione fisica del database (le informazioni di connessione per ODBC, per esempio il DSN, UID, User ID). Fare clic su *Avanti* per continuare.
14. Nella finestra di dialogo successiva si connette al database specificato e lo apre. Fare clic su *Avanti* per continuare.
15. La finestra di dialogo *Form* permette di specificare il nome del nuovo form, il suo layout (ogni layout richiede informazioni differenti, i passi dipendono dal tipo di layout scelto. per esempio, scegliere *Record singolo*) e il tipo di associazione Controllo ADO Data. Fare clic su *Avanti* per continuare.
16. La finestra di dialogo richiede le informazioni sulla query principale. Dalla lista a discesa selezionare la tabella, poi selezionare i campi che dovranno essere visualizzati sul form e impostare l'ordinamento sulla colonna. Fare clic su *Avanti* per continuare.
17. La finestra di dialogo, *Seleziona controllo*, permette di selezionare i controlli di navigazione e manipolazione dei dati che dovranno apparire sul form. Pulsante *Inserisci*: permette l'aggiunta di nuovi record nel database. Pulsante *Aggiorna*: per aggiornare i dati del database in base a quelli contenuti nel form. Pulsante *Elimina*: permette di cancellare record del database attraverso il form. Pulsante *Rivisualizza*: permette di aggiornare i dati del form leggendoli nuovamente dal database. Pulsante *Chiudi*: permette di chiudere il form. Pulsante *Mostra controllo data*: visualizza il controllo all'interno del form. Fare clic su *Avanti* per continuare.
18. A questo punto si possono salvare tutte le impostazioni effettuate in un profilo. Fare clic su *Fine* per creare il form.
19. Se si desidera creare un altro form di dati fare clic su *Sì*, altrimenti su *No*.
20. Fare clic su *Fine* per creare l'applicazione.
21. Chiudere la finestra di dialogo che avvisa dell'avvenuta creazione dell'applicazione, facendo clic su *OK*. La creazione guidata ha lasciato l'ambiente di sviluppo sgombro da qualsiasi elemento, ma è possibile fare doppio clic su qualunque oggetto nella finestra Progetto per vedere i form e i moduli di codice. Premere F5 per compilare ed eseguire l'applicazione.

Ricordare che il lavoro della creazione guidata è generare un'applicazione completamente funzionante, ma generica, in grado di eseguire operazioni basilari di avvio, come visualizzare una finestra, impostare una struttura di menu e aggiungere elementi speciali come una barra degli strumenti e la finestra Informazioni su.

I controlli Internet

Se si utilizza la versione Professional o Enterprise di Visual Basic, è possibile impiegare molti controlli Internet avanzati (e ActiveX) per aggiungere e controllare l'accesso ad Internet nelle applicazioni.

Vari controlli Internet appaiono quando si sceglie nel menu **Progetto/ Componenti**.

1. InternetTransfer incapsula i tre più diffusi protocolli Internet, ovvero HTTP, FTP e Gopher. È possibile scaricare i file direttamente dall'interno delle applicazioni Visual BASIC utilizzando FTP.
2. WebBrowser incapsula un browser Web direttamente nell'applicazione.
3. WinSock offre un comune controllo Windows per la connessione e lo scambio dei dati che fornisce due protocolli, ovvero UDP e TCP.

Questi controlli iniziano con l'abbreviazione IE.

Nome componente	**Descrizione**
IE AnimateButton	Visualizzazione animata che mostra la connessione di IE
IE PopupMenu	Un controllo menu che appare sulla pagina Web
IE PopupWindow	Una finestra a schede che apre una nuova finestra di connessione
IE Preloader	Precarica un sito prima che inizi l'accesso visibile ad Internet
IE SuperLabel	Un'etichetta di pagina Web
IE Timer	Fornisce operazioni di temporizzazione per servizi Internet
Microsoft Internet Controls	Controllo browser Web
Microsoft Internet Transfer Control	Controllo protocollo di trasferimento
Microsoft WinSock Control	Connessione Windows a protocolli Internet comuni

AUTOCOMPOSIZIONE INSTALLAZIONE

INTRODUZIONE

In questo passaggio è possibile scegliere il file del progetto Visual BASIC (.VBP) o il file del modello dell'Autocomposizione Installazione (.SWT) per l'applicazione da distribuire. È inoltre possibile impostare la creazione di un‘applicazione d’installazione con o senza un file di dipendenze, di un programma d’installazione specifico per lo scaricamento da Internet oppure solo di un file di dipendenze. Nel file .VBP e nei file di origine sono individuate le informazioni sugli oggetti, i riferimenti, le risorse, le librerie a collegamento dinamico e così via da includere nei file di distribuzione, nonché informazioni sulla posizione del file .EXE o .DLL.

È buona norma di programmazione fornire sempre un file di dipendenze (.DEP) insieme all'applicazione se si tratta di un componente condiviso, quale un controllo ActiveX o un componente ActiveX, anche se il file indica semplicemente che, oltre ai file runtime di Microsoft Visual BASIC, non esistono dipendenze per il componente.

Opzioni dell'autocomposizione

Posizione del progetto consente di specificare il nome del file di progetto che si desidera distribuire e la posizione nel PC. Se il file eseguibile .EXE, .DLL o .OCX non esiste, sarà compilato automaticamente.

Sfoglia Visualizza la finestra di dialogo comune Apri in cui è possibile individuare e scegliere il file di progetto dell'applicazione che si desidera creare. Il nome del file è inserito automaticamente nel campo relativo alla posizione del progetto.

Ricompila il progetto se questa opzione è selezionata, il file di progetto .EXE, .DLL o .OCX è ricompilato indipendentemente dal fatto che sia stato o meno individuato un file .EXE, .DLL o .OCX per il progetto. Selezionare questa opzione se si desidera eseguire la ricompilazione prima di distribuire l'applicazione.

Prima di utilizzare l'Autocomposizione Installazione, è necessario compilare e salvare il file di progetto almeno una volta. In caso contrario, l'Autocomposizione Installazione e la distribuzione dell’applicazione d’installazione potrebbero non funzionare correttamente.

Opzioni

Crea programma d’installazione

Crea file dipendenze: crea un file contenente informazioni sulle dipendenze da includere nel programma di installazione. I file di dipendenze sono utilizzati per qualsiasi oggetto di un file .EXE, .DLL o .OCX, componente ActiveX o progetto utilizzabile come componente in altri progetti.

La posizione d’installazione predefinita per i file .DEP corrisponde alla directory di installazione del file .OCX. È tuttavia possibile modificarla per il file .DEP nella finestra di dialogo Dettagli file cui si accede dal passaggio Riepilogo file.

Se si modifica la posizione del file .DEP, è possibile che il file non sia individuato in una sessione successiva dell'Autocomposizione Installazione.

Non è possibile assegnare lo stesso nome di base a due file che richiedono file di dipendenze. A ciascun file deve essere associato un file di dipendenze distinto il cui nome, per impostazione predefinita, corrisponde al nome del file con estensione .DEP.

Crea installazione scaricamento Internet

Crea un programma di installazione scaricabile da Internet solo per i progetti di controlli ActiveX, EXE ActiveX e DLL ActiveX cui sono associate classi pubbliche, compresi i progetti che includono oggetti UserDocuments.

Quando si seleziona questa opzione, è disponibile il pulsante Novità che consente di visualizzare una pagina Web Microsoft che contiene informazioni per lo scaricamento di dati da Internet.

Crea solo file dipendenze

Crea un file di dipendenze avente lo stesso nome del progetto con estensione .DEP e lo inserisce nella stessa directory del progetto.
Per creare il file di dipendenze in modo corretto, è necessario fornire informazioni accurate in tutti gli altri passaggi dell'Autocomposizione Installazione, esattamente come se si dovesse creare un programma d'installazione per l'applicazione. Il file di dipendenze includerà informazioni sulle dipendenze, ad esempio i file dipendenti, la corrispondente destinazione e versione, necessarie per installare il progetto come parte di un'altra applicazione.
Novità Consente di visitare il sito Web Microsoft in cui è possibile trovare informazioni aggiuntive sullo scaricamento di dati da Internet.
Questo pulsante è disponibile solo quando si sceglie l'opzione Crea installazione scaricamento Internet.
? Visualizza l'argomento della Guida relativo a questo passaggio. In alternativa, è possibile premere F1.
Annulla Annulla le azioni precedenti e chiude l'Autocomposizione Installazione.
Indietro Torna al passaggio precedente.
Avanti Visualizza il passaggio successivo.
Fine Crea un'applicazione d'installazione in base alle impostazioni specificate. È inoltre visualizzata una finestra di dialogo di conferma. È disponibile solo quando sono state fornite informazioni sufficienti per la creazione di un programma di installazione.

Autocomposizione Installazione: Dipendenze mancanti

La visualizzazione di questo passaggio indica che durante l'esecuzione dell'Autocomposizione Installazione non sono state trovate le informazioni sulle dipendenze dei file inclusi nell'elenco. La ricerca di tali informazioni è eseguita nel file VB5DEF.INI e/o nei file di dipendenze (.DEP) che dovrebbero essere disponibili per ciascun file .EXE, .OCX e .DLL dell'applicazione.
Nel caso di file di altri produttori, richiedere le informazioni sulle dipendenze o i file di dipendenze necessari al rispettivo produttore.
Se uno dei file elencati è un progetto creato in Visual BASIC, è possibile utilizzare l'Autocomposizione Installazione per creare un file di dipendenze per il progetto. Riavviare l'Autocomposizione, selezionare il file di progetto (.VBP) del componente e scegliere l'opzione Crea solo file dipendenze. Eseguire quindi tutti i passaggi in modo da creare il file di dipendenze mancante, quindi riavviare l'Autocomposizione Installazione e provare di nuovo con il progetto corrente.
Se per un file indicato nell'elenco non sono necessarie informazioni aggiuntive, selezionarlo in modo che sia visualizzato un segno di spunta a sinistra del nome del file. Ciò consente d'indicare che al file non è associata alcuna dipendenza. In futuro il file non sarà visualizzato in questa finestra di dialogo.
Se alcuni file non sono selezionati, sarà visualizzata una finestra di dialogo in cui si chiede se le informazioni sulle dipendenze mancanti devono essere ignorate nella sessione corrente. È possibile scegliere Sì e continuare l'esecuzione dell'Autocomposizione oppure disattivare la successiva visualizzazione della finestra di dialogo per eventuali altri file di dipendenze mancanti.
Opzioni dell'autocomposizione
Dipendenze mancanti Visualizza un elenco dei file per cui non sono stati trovati i file di dipendenze.
? Visualizza l'argomento della Guida relativo a questo passaggio. In alternativa, è possibile premere F1.
Annulla Annulla le azioni precedenti e chiude l'Autocomposizione Installazione.
Indietro Torna al passaggio precedente.
Avanti Visualizza il passaggio successivo.
Fine Crea un programma d'installazione in base alle impostazioni specificate. È inoltre visualizzata una finestra di dialogo di conferma. È disponibile solo quando sono state fornite informazioni sufficienti per la creazione di un programma di installazione.

Autocomposizione Installazione: Metodo di distribuzione
In questo passaggio è possibile scegliere la modalità di distribuzione dell'applicazione.
Opzioni dell'autocomposizione
Disco floppy Copia il file Setup.exe e tutti gli altri file di installazione del progetto in dischi floppy.
Directory singola Copia il file Setup.exe e tutti gli altri file di installazione del progetto in una singola directory da cui l'utente può eseguire l'installazione.
Directory disco Crea più directory, quali \Disk1, \Disk2 e \Disk3, e copia i file nelle directory appropriate come se ciascuna directory rappresentasse un disco floppy. L'utente potrà eseguire l'installazione da \Disk1 o creare una copia su floppy dalle immagini disco.
Prima di continuare, è necessario eliminare tutte le sottodirectory e i file inclusi nella directory di destinazione da Gestione risorse, in quanto questa operazione non è eseguita automaticamente.
Non è possibile scegliere l'unità principale di rete o l'unità locale, a meno che non si tratti di un'unità disco floppy.
? Visualizza l'argomento della Guida relativo a questo passaggio. In alternativa, è possibile premere F1.
Annulla Annulla le azioni precedenti e chiude l'Autocomposizione Installazione.
Indietro Torna al passaggio precedente.
Avanti Visualizza il passaggio successivo.
Fine Crea un programma di installazione in base alle impostazioni specificate. È inoltre visualizzata una finestra di dialogo di conferma. È disponibile solo quando sono state fornite informazioni sufficienti per la creazione di un programma di installazione.
Autocomposizione Installazione: Componenti server ActiveX
In questo passaggio è possibile includere nel progetto componenti ActiveX aggiuntivi.
È necessario deselezionare la casella di controllo dei file già installati nel PC dell'utente e dei file per i quali non è disponibile alcuna autorizzazione di distribuzione. Tutti questi file devono essere indicati nelle note di rilascio in modo che gli utenti possano verificare di avere tutti i file necessari.
Opzioni dell'autocomposizione
Elenco dei componenti ActiveX Elenco dei componenti ActiveX creati quando, durante l'esecuzione dell'Autocomposizione Installazione, sono stati cercati i file di progetto (.VBP) e trovati riferimenti ai componenti server ActiveX utilizzati nell'applicazione.
Aggiungi locale Visualizza la finestra di dialogo Aggiungi locale in cui è possibile individuare e aggiungere i componenti ActiveX (*.DLL, *.EXE, *.OCX) nell'elenco dei file da distribuire con l'applicazione. La finestra di dialogo Aggiungi locale è una finestra di dialogo comune Apri.
Aggiungi remoto Visualizza la finestra di dialogo Aggiungi remoto in cui è possibile individuare e aggiungere i componenti ActiveX remoti (*.VBR) da aggiungere nell'elenco dei file da distribuire con l'applicazione. La finestra di dialogo Aggiungi remoto è una finestra di dialogo comune Apri.
Dettagli remoto Visualizza la finestra di dialogo Dettagli connessione remota in cui è possibile selezionare il modello **DCOM** (*Distributed Component Object Model*) o l'automazione remota e aggiungere l'indirizzo di rete, il protocollo di rete e le informazioni di autenticazione necessarie per il funzionamento corretto del componente ActiveX remoto.
Dettagli file Visualizza la finestra di dialogo Dettagli file in cui sono visualizzate informazioni sul file.
? Visualizza l'argomento della Guida relativo a questo passaggio. In alternativa, è possibile premere F1.
Annulla Annulla le azioni precedenti e chiude l'Autocomposizione Installazione.
Indietro Torna al passaggio precedente.
Avanti Visualizza il passaggio successivo.
Fine Crea un programma di installazione in base alle impostazioni specificate. È inoltre

visualizzata una finestra di dialogo di conferma. È disponibile solo quando sono state fornite informazioni sufficienti per la creazione di un programma d'installazione.

Autocomposizione Installazione: Conferma dipendenze

Questo passaggio è possibile confermare eventuali file di dipendenze aggiuntivi individuati dall'Autocomposizione Installazione, ad esempio i controlli ActiveX utilizzati nel progetto o i file cui è fatto riferimento nel progetto.

È necessario deselezionare la casella di controllo dei file già installati nel PC dell'utente e dei file per i quali non è disponibile l'autorizzazione di distribuzione. Tutti questi file devono essere indicati nelle note di rilascio in modo che gli utenti possano verificare di avere tutti i file necessari.

Quando si crea un nuovo progetto in Visual BASIC, sono aggiunti automaticamente determinati controlli nella casella degli strumenti e un riferimento agli oggetti di accesso ai dati **DAO** (*Data Access Objects)*. Durante l'esecuzione dell'Autocomposizione Installazione questi controlli e i riferimenti sono sempre riconosciuti indipendentemente dal fatto che siano o meno utilizzati nell'applicazione. Se nel passaggio Conferma dipendenze sono indicati controlli non utilizzati nell'applicazione oppure è visualizzato il passaggio Accesso ai dati anche se non sono stati aggiunti riferimenti agli oggetti di accesso ai dati, sarà necessario deselezionare i file non utilizzati rimuovendo gli strumenti e i riferimenti dai file di origine in modo manuale o deselezionando la corrispondente casella di controllo nell'elenco dei file.

Opzioni dell'autocomposizione

Elenco dei file con dipendenze Elenco delle dipendenze individuate per l'applicazione.

Dettagli file Visualizza la finestra di dialogo Dettagli file.

? Visualizza l'argomento della Guida relativo a questo passaggio. In alternativa, è possibile premere F1.

Annulla Annulla le azioni precedenti e chiude l'Autocomposizione Installazione.

Indietro Torna al passaggio precedente.

Avanti Visualizza il passaggio successivo.

Fine Crea un programma di installazione in base alle impostazioni specificate. È inoltre visualizzata una finestra di dialogo di conferma. È disponibile solo quando sono state fornite informazioni sufficienti per la creazione di un programma di installazione.

Autocomposizione Installazione: Riepilogo file

In questo passaggio è visualizzato un elenco dei file che è necessario distribuire insieme all'applicazione.

È necessario deselezionare la casella di controllo dei file per i quali non è disponibile l'autorizzazione di distribuzione. Tali file devono essere indicati nelle note di rilascio in modo che gli utenti possano verificare di avere tutti i file necessari.

Opzioni dell'autocomposizione

Elenco dei file Elenco dei file necessari per il corretto funzionamento dell'applicazione.

Aggiungi Aggiunge file nella casella di riepilogo dei file. È possibile selezionare più file nella finestra di dialogo Aggiungi file.

Nella lista di distribuzione è possibile aggiungere anche file non richiesti dall'applicazione. Nell'elenco tuttavia non è possibile includere nomi di file uguali anche se inclusi in directory diverse. Se si aggiunge un file avente lo stesso nome di un file già incluso nell'elenco, sarà visualizzato un messaggio di errore.

Se durante l'esecuzione dell'Autocomposizione Installazione non sono trovate le informazioni sulle dipendenze nel file Vb5dep.ini o in un file .DEP associato al file da aggiungere, sarà visualizzato un messaggio in cui è chiesto se si desidera includere comunque il file. Se si aggiunge il file, le informazioni sulle dipendenze richieste non saranno incluse nel pacchetto d'installazione e di conseguenza sia il programma di installazione che l'applicazione potrebbero non funzionare correttamente. È inoltre possibile disattivare la visualizzazione del messaggio per questo file.

Se si è certi che non esista alcun file di dipendenze, è possibile continuare o tornare al passaggio Seleziona progetti e opzioni e creare un file di dipendenze per il file.

Dettagli file Visualizza le dimensioni, la data, l'ora, nonché la posizione corrente e di destinazione del file selezionato nelle caselle di riepilogo. Se disponibili, sono inoltre visualizzate le informazioni sulla versione.
È possibile modificare la posizione d’installazione del file selezionato. Se si esegue questa operazione per un file condiviso o un server ActiveX remoto, si possono avere effetti negativi sulla funzionalità.
Se si crea un programma d’installazione che potrà essere scaricato da Internet, l'opzione Directory di destinazione non sarà disponibile. Inoltre, sarà possibile scaricare i file .VBL da Internet, ma non registrarli. La registrazione deve pertanto essere eseguita dall'utente utilizzando RegEdit.
La modifica della directory di destinazione o dello stato di un file condiviso o server ActiveX remoto può avere effetti negativi sulle capacità del componente e impedire la successiva rimozione del componente stesso. Se, ad esempio, un componente ActiveX condiviso è installato in due directory diverse, con la rimozione di uno dei file è possibile che siano rimosse le informazioni di registrazione per l'accesso ai due file, rendendo quindi impossibile l'accesso all'altro componente.
Informazioni di riepilogo Visualizza una finestra di dialogo in cui è indicato il numero dei file di programma e d’installazione, il numero totale dei file, il numero di byte espansi di ciascun file di programma e di installazione, il numero totale di byte espansi e il percorso della posizione di installazione.
Dipendenza Indica il file di dipendenza richiesto per il file selezionato.
? Visualizza l'argomento della Guida relativo a questo passaggio. In alternativa, è possibile premere F1.
Annulla Annulla le azioni precedenti e chiude l'Autocomposizione Installazione.
Indietro Torna al passaggio precedente.
Avanti Visualizza il passaggio successivo.
Fine Crea un programma di installazione in base alle impostazioni specificate. È inoltre visualizzata una finestra di dialogo di conferma. È disponibile solo quando sono state fornite informazioni sufficienti per la creazione di un programma di installazione.

APPLICAZIONI INTERNET/INTRANET

INTRODUZIONE

Le tecnologie per Internet introdotte in Visual BASIC legano il codice all'interfaccia realizzata tramite pagine HTML. Si tratta di un potente meccanismo per realizzare complesse operazioni di accesso ai dati, senza le difficoltà tipiche degli script o delle procedure CGI. Inoltre si può utilizzare un ambiente di sviluppo, che include moduli di classe, controlli, finestre di progettazione ed un sofisticato debugger.
Esistono fondamentalmente due tipi di applicazioni per Internet:

1. **applicazioni DHTML**: si tratta di applicazioni scaricate ed eseguite in un browser su un PC client, oppure basate su HTML che girano all'interno di un controllo WebBrowser. In ogni caso, un'applicazione DHTML consente di gestire gli eventi della pagina HTML direttamente sul client, sebbene sia possibile effettuare chiamate al server quando necessario;
2. **applicazioni IIS** (*Internet Information Server*): sono applicazioni che risiedono su un server web, processano le richieste provenienti dai client ed eseguono codice Visual BASIC per rispondere all'utente. Il processo avviene interamente sul server.

Le applicazioni DHTML richiedono Internet Explorer 4.01 con SP1 o successivo, mentre le applicazioni IIS sono indipendenti dal browser e dalla piattaforma. Per questi motivi le seconde sono indicate per siti Internet pubblici, quando non è possibile prevedere le caratteristiche (browser e sistema operativo) dei client, mentre le prime sono adatte per siti Intranet, dove risulta più facile standardizzare il tipo di browser utilizzato.

LE APPLICAZIONI DHTML

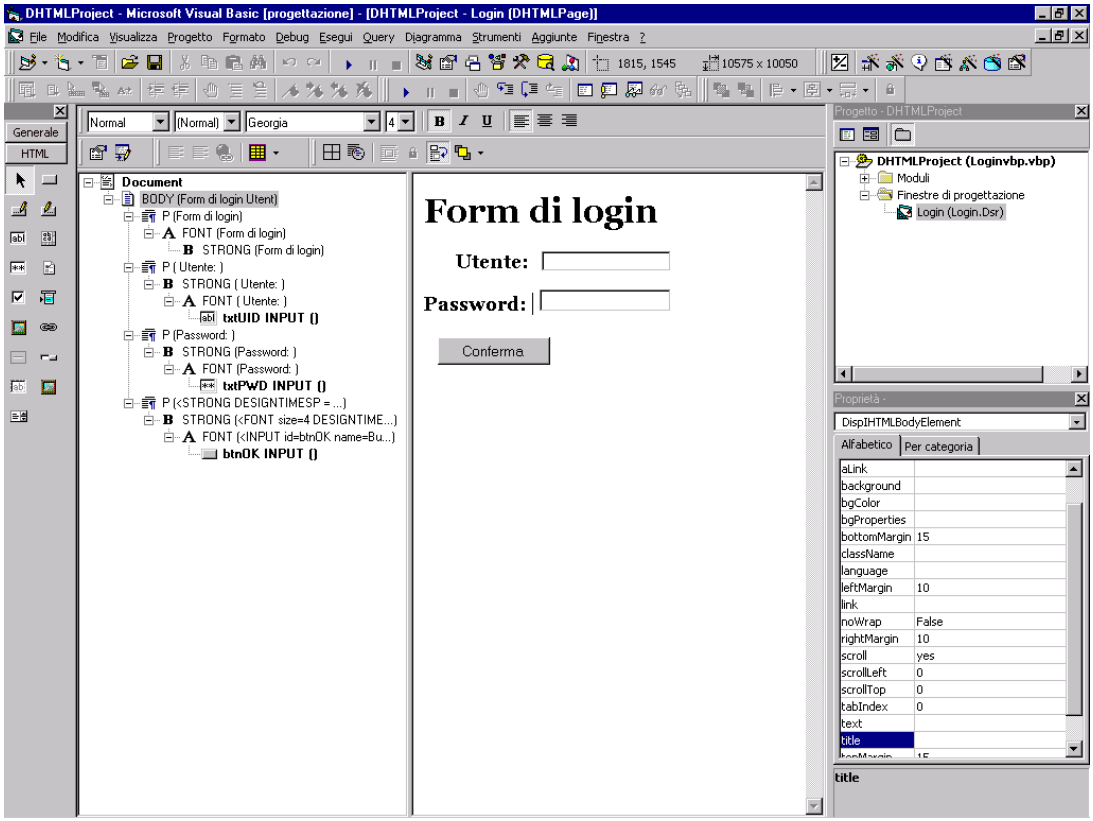
Si possono individuare diverse situazioni per le quali una soluzione basata unicamente sul server non è indicata, l'esempio classico è quando la rete non è sempre disponibile. Un'applicazione DHTML è un'applicazione che utilizza una combinazione di HTML dinamico e codice Visual BASIC compilato e risiede sulla macchina che ospita il browser, dove interpreta e risponde agli eventi scatenati nel browser. Sebbene possano esservi delle chiamate al server, la maggior parte dell'elaborazione avviene sul client, fatto che aumenta i tempi di risposta. Le applicazioni DHTML sono strutturate in maniera diversa rispetto alle tradizionali applicazioni Visual BASIC. L'interfaccia consiste non di form, ma di una serie di pagine HTML, memorizzate in file .HTM o .HTML, che contengono gli elementi visuali l'analogo dei controlli Visual BASIC. Queste pagine possono esser create con il disegnatore integrato in Visual BASIC oppure con un opportuno editor di pagine HTML. In definitiva un'applicazione DHTML consiste di:

- una o più pagine HTML
- codice Visual BASIC che gestisce gli eventi generati dalle pagine HTML;
- un run-time che ospita la pagina in un browser o in un controllo WebBrowser;
- una DLL che contiene il codice Visual BASIC ed alla quale il run-time accede.

Per realizzare agevolmente queste applicazioni, si usa una finestra di progettazione (designer) di pagine DHTML, un potente strumento che offre una superficie sulla quale posizionare gli elementi d'interfaccia, trascinati dalla casella degli strumenti.
Si noti la visualizzazione gerarchica, sulla sinistra, e la finestra di dettaglio, sulla destra, dove si trascinano i controlli dalla casella degli strumenti. L'impostazione delle proprietà per gli oggetti avviene seguendo le modalità tipiche di Visual BASIC. Gli oggetti evidenziati in grassetto nel treeview sono quelli programmabili, per i quali è stata impostata la proprietà ID. Visual BASIC, fornisce una vista gerarchica degli elementi all'interno della pagina DHTML ai quali è possibile associare codice.
I principali vantaggi dell'utilizzo di applicazioni DHTML sono:

- accesso al modello ad oggetti di DHTML;

- riduzione del carico di lavoro sul server, perché non tutte le richieste sono indirizzate a quest'ultimo;
- risposte più rapide ed un numero inferiore di refresh, perché è il client a elaborare e rispondere alle azioni dell'utente;
- interazione dinamica, grazie alla manipolazione diretta degli elementi sulla pagina;
- migliore gestione dello stato, grazie alla memorizzazione, lato client, delle proprietà critiche di una pagina. Si superano così le limitazioni legate alla natura stateless del protocollo HTTP e le applicazioni multipagina non necessitano di cookies o altri interventi da parte del server;
- capacità di elaborazione disconnessa, grazie alla cache del browser;
- sicurezza del codice, dato che le applicazioni sono inviate al client sotto forma di DLL. Nessuno può accedere alla pagina e leggere o modificare gli script.



I principali oggetti di un'applicazione DHTML sono:

- **BaseWindow**: l'oggetto che rappresenta l'istanza del browser ed è utilizzato per visualizzare i documenti;
- **Document**: l'oggetto che rappresenta la pagina HTML, fornisce l'accesso al modello ad oggetti di DHTML e gestisce le azioni dell'utente;
- **DHTMLPage**: l'oggetto, contenuto all'interno dell'oggetto Document, che consente il legame tra il codice Visual BASIC e la pagina in fase di esecuzione. È questo oggetto che fornisce gli eventi Load, Unload, Initialize e Terminate.

Il tipico ciclo di vita di un oggetto DHTMLPage è il seguente:

- L'evento **Initialize** si verifica per primo durante il processo di caricamento, ogniqualvolta la DLL per l'applicazione è creata dal sistema. È sempre il primo evento nel ciclo di vita di un'applicazione DHTML. Quando l'evento Initialize è scatenato, gli oggetti sulla pagina non sono ancora stati creati e quindi i riferimenti a questi non sono validi e non è possibile impostare le proprietà degli oggetti.
- L'evento **Load** si verifica subito dopo l'evento Initialize. Quando la pagina è caricata in maniera asincrona, questo evento si verifica subito dopo il caricamento del primo elemento. In caso contrario è scatenato dopo il caricamento dell'ultimo elemento.
- L'evento **Unload** si verifica quando l'utente si sposta su un'altra pagina o chiude

l'applicazione.
- L'evento **Terminate** si verifica quando la pagina HTML sta per essere distrutta. Durante questo evento gli oggetti della pagina non esistono più e non è possibile referenziarli.

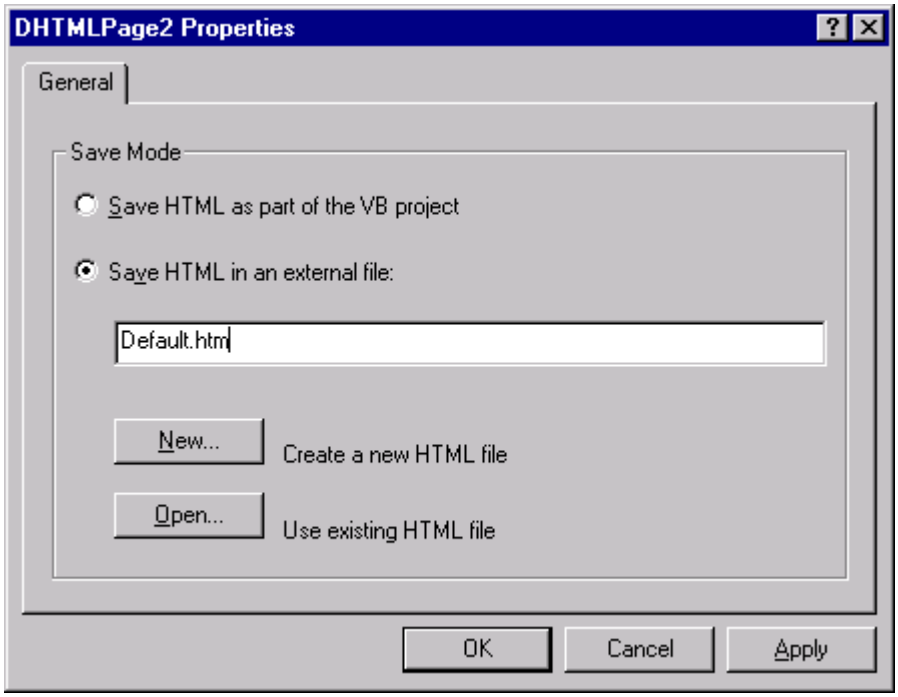
Gli elementi di una pagina possono includere intestazioni, paragrafi, immagini, campi, tabelle. Nelle versioni precedenti dell'HTML non era possibile accedere alle proprietà ed agli eventi associati a questi oggetti, perché soltanto un numero ristretto di questi poteva essere programmabile. Con l'HTML dinamico è possibile accedere e manipolare tutti gli elementi di una pagina alla stregua delle form Visual BASIC consente, tra l'altro, di:

- utilizzare nuove unità di misura per un controllo più fine del posizionamento degli oggetti;
- creare elementi che ne racchiudono altri, che sono contenuti in altri o che agiscono come un unico blocco;
- gestire meglio le tabelle e la grafica;
- utilizzare stili e contenuto dinamici. L'obiettivo iniziale dell'HTML – separare il contenuto di una pagina dalla sua presentazione – è finalmente raggiunto. L'utilizzo dei fogli di stile **CSS** (*Cascading Style Sheet*) consente di memorizzare il formato di presentazione delle informazioni in un file esterno alla pagina HTML che le contiene. Un foglio di stile è un'insieme di proprietà che controllano l'aspetto degli elementi su una pagina.

A differenza della programmazione Visual BASIC tradizionale, dove ogni oggetto dispone di un proprio gestore degli eventi, gli elementi dell'HTML dinamico possono condividere un gestore di evento. Quando un evento si verifica su un oggetto figlio, questo risale la gerarchia degli elementi della pagina sino ad incontrare un gestore dell'evento. Questo processo è chiamato **bubbling** degli eventi. La gerarchia da risalire è determinata dalla posizione degli elementi all'interno dello stream HTML e non corrisponde necessariamente all'ordine di visualizzazione. Il bubbling degli eventi aiuta a ridurre la quantità di codice da scrivere ed è disattivabile a discrezione del programmatore. Il processo di creazione di una tipica applicazione DHTML è simile a quello di una normale applicazione Visual BASIC. Occorre:

- iniziare un nuovo progetto Visual BASIC e selezionare il modello Applicazione DHTML;
- dalla finestra **Progetto** selezionare **Finestre di progettazione** ed aprire il designer DHTMLPage1, creato automaticamente;
- aggiungere elementi e controlli e disporli a proprio piacimento;
- utilizzare la finestra di dialogo **Proprietà** per referenziare un file HTML esterno, se si desidera modificare una pagina esistente;
- aggiungere il codice per gli elementi di cui si vogliono gestire gli eventi;
- se necessario, aggiungere altre pagine al progetto. In questo caso occorre specificare la modalità di salvataggio delle pagine. La pagina di Proprietà offre due possibilità:
- salvare la pagina nel disegnatore: per condividere agevolmente le pagine tra sviluppatori o muovere l'applicazione su una diversa macchina di sviluppo senza problemi
- salvare la pagina in un file esterno per poter utilizzare un editor esterno;
- eseguire il debug dell'applicazione lanciando il progetto e visualizzando la pagina in Internet Explorer;
- compilare il progetto;
- distribuire l'applicazione.

Un'applicazione DHTML è in realtà un progetto di tipo DLL ActiveX che automaticamente imposta i riferimenti necessari per accedere al designer ed ai controlli HTML necessari. Ogni finestra di progettazione corrisponde ad una pagina e l'applicazione gira in-process con il browser. Il designer consiste di due parti: la finestra di dettaglio, sulla destra, e quella di visualizzazione gerarchica, sulla sinistra.



La finestra di dettaglio mostra la pagina come appare nel browser ed è l'area di posizionamento dei controlli. La visualizzazione gerarchica è utilizzata per mostrare gli elementi e le loro relazioni con il modello ad oggetti. Gli elementi visualizzati sono pulsanti, testo, immagini, checkbox e tag come quelli di formattazione del testo (grassetto, italico) e di organizzazione dei controlli. Sebbene tutti gli elementi presenti sulla pagina siano visualizzati, è possibile scrivere codice soltanto per quelli per i quali è stata impostata la proprietà *ID*. Questa attributo fornisce un identificatore univoco per l'elemento: è l'analogo della proprietà *Name* dei controlli Visual BASIC. Anche i tag HTML dispongono di un attributo *Name* che non riveste un ruolo fondamentale nella programmazione DHTML. Mentre l'*ID* deve sempre essere univoco, la stessa cosa non può dirsi per l'attributo *Name*. Il modello ad oggetti di DHTML espone come eventi le azioni dell'utente. È possibile intercettare e processare gli eventi tramite codice. Diversi aspetti della programmazione con DHTML risultano inconsueti per un programmatore Visual BASIC. Molti eventi sono simili a quelli di Visual BASIC, anche se i nomi sono leggermente diversi: tutti gli eventi di DHTML sono preceduti dal prefisso "on". Gli eventi di tastiera (**onkeydown**, **onkeyup**, **onkeypress**) e di mouse (**onmousedown**, **onmouseup**, **onmousemove**) corrispondono molto da vicino a quelli di Visual BASIC. Gli eventi **onmouseout** e **onmouseover** consentono di specificare gli elementi dai quali o verso i quali il puntatore si muove. Gli eventi di selezione e di gestione del fuoco differiscono leggermente. In particolare gli eventi di gestione del fuoco sono scatenati soltanto da alcuni elementi, mentre gli eventi di gestione della selezione sono gestiti in maniera differente. Gli eventi **onfocus** e **onblur** sono scatenati sempre sia che ci si sposti tra gli elementi di una pagina, sia quando l'utente si porta in un'altra applicazione. Esistono, infine, alcuni eventi (**onabort**, **onreset** e **onsubmit**) che sono specifici di DHTML. Molti elementi delle pagina HTML hanno un evento predefinito, un'azione tipicamente associata al loro utilizzo. In Visual BASIC è sufficiente scrivere del codice all'interno della procedura di gestione dell'evento per specificare l'azione da intraprendere in risposta a quell'evento. In Internet Explorer, quando si collega un evento ad un oggetto, il sistema ritorna sempre True o False quando l'evento è innescato. Il valore True permette l'esecuzione dell'azione, il valore False la impedisce. L'impostazione e la lettura delle proprietà di un elemento HTML funzionano differentemente in Visual BASIC rispetto a DHTML. Nelle form è possibile accedere alle proprietà di un controllo semplicemente referenziandolo e separando con un punto il nome della proprietà. Per impostare una proprietà di un elemento occorre fare accesso alla sua collezione **Style** e da questa referenziare la proprietà voluta. Ciò non è vero per l'oggetto Document, alle cui proprietà si può accedere direttamente.

Button1.Style.backgroundcolor = "red"
Document.bgcolor = "white"
I dati contenuti negli elementi di una pagina HTML possono essere letti e impostati tramite la proprietà Value. *Me.author.value*
La navigazione si effettua tramite l'oggetto Hyperlink oppure da codice utilizzando il metodo **navigate** dell'oggetto **BaseWindow**.
BaseWindow.Navigate "Project1.DHTMLPage2.html"
È possibile creare elementi HTML ed inserirli in una pagina a run-time, utilizzando il metodo **insertAdjacentHTML**.
Me.Document.body.insertAdjacentHTML "BeforeEnd", _
"<DIV><HR SIZE=2></DIV>"
Me.Document.body.insertAdjacentHTML "BeforeEnd", _
"<DIV CLASS=custhead>Customer Information</DIV>
Me.Document.body.insertAdjacentHTML "BeforeEnd", _
"<DIV id=custinfo>"
Per quanto riguarda la gestione dello stato, necessaria a causa della natura stateless del protocollo HTTP, è possibile utilizzare le funzioni **PutProperty** e **GetProperty** per memorizzare e accedere ai dati. Queste funzioni memorizzano i dati all'interno di un oggetto *property bag* fintantoché la finestra del browser è aperta.
PutProperty "Cognome", txtCognome.value
MsgBox "Cognome: " & GetProperty("Cognome")

LE APPLICAZIONI IIS E L'OGGETTO WEBCLASS

Visual BASIC offre ai programmatori la possibilità di scrivere codice per gestire le risposte di un server web. L'applicazione risultante è accessibile da qualunque browser su qualunque piattaforma. Il problema per gli sviluppatori che vogliono programmare un server web, è sempre stato quello di familiarizzare con la sintassi di linguaggi nuovi e più restrittivi, che non consentono il riutilizzo delle conoscenze acquisite. L'oggetto WebClass è stato ideato per risolvere questo problema. Grazie ad esso, i programmatori dispongono di un ambiente familiare, basato su componenti, per la realizzazione di applicazioni web che sfruttano il server. Le WebClass sono componenti creati con Visual BASIC, il cui ruolo è di operare come strato intermedio tra il client ed il server, per intercettare e gestire le richieste HTTP. La creazione di un'applicazione tradizionale segue un processo facile da capire. Per prima cosa si crea l'interfaccia, combinando opportunamente i controlli su una form. Quindi si scrive il codice per rispondere agli eventi e, dopo il debug, si distribuisce l'applicazione. Le WebClass estendono questo modello di programmazione alle applicazioni web lato server. Si crea una pagina Web utilizzando FrontPage o Visual InterDev e la si carica nel designer. Si scrive il codice per rispondere agli eventi e, lanciando l'applicazione dalla modalità Disegno, se ne può eseguire il debug prima di distribuirla. Le differenza tra un'applicazione WebClass ed un'applicazione Visual BASIC tradizionale sono, però, notevoli. Innanzitutto, l'interfaccia utente è costituita da pagine HTML o DHTML, in secondo luogo l'utente interagisce con l'applicazione tramite un link o un URL nel browser. Infine, la logica applicativa risiede sul server e non è necessaria alcuna installazione lato client. Si può assimilare un'applicazione WebClass ad un oggetto COM che risiede sul server e risponde alle richieste HTTP provenienti dai browser. Una WebClass è un contenitore di metodi ed eventi di un'applicazione che comunica via HTTP e che si avvale delle stesse funzionalità disponibili in ASP. Un'applicazione WebClass, o IIS, è, in definitiva, un'applicazione che utilizza una combinazione di HTML e codice Visual BASIC. Nella sua forma più elementare intercetta le richieste da parte dell'utente ed invia al browser una pagina HTML, senza ricorrere a procedure di script o CGI, ma utilizzando unicamente codice Visual BASIC. Per un utente un'applicazione IIS è composta da una serie di pagine HTML. Per uno sviluppatore, un'applicazione IIS è composta da un oggetto chiamato WebClass, il quale contiene una serie di risorse chiamate WebItem. L'oggetto WebClass agisce come la principale unità funzionale dell'applicazione. I WebItem sono le

pagine HTML e gli altri dati che la WebClass invia al browser. Le applicazioni IIS ricordano quelle ASP, ma le applicazioni ASP sono per gli sviluppatori di script che realizzano pagine Web, mentre le applicazione IIS sono per gli sviluppatori Visual BASIC che vogliono costruire applicazioni web, che includono logiche articolate di elaborazione dei dati. Un'applicazione IIS può anche ricordare un'applicazione DHTML.
Esistono però delle fondamentali differenza tra i due tipi:

- le applicazioni DHTML sono pensate per Intranet e dipendono da Internet Explorer, mentre le applicazioni IIS non richiedono un sistema operativo o un browser particolari;
- le applicazioni DHTML utilizzano un modello ad oggetti diverso rispetto a quello delle applicazioni IIS. In particolare, le applicazioni IIS utilizzano il modello ad oggetti di ASP, le altre quello di DHTML;
- le applicazioni IIS sono progettate per eseguire la maggior parte delle operazioni sul server, mentre quelle DHTML sono eseguite quasi interamente sul client.

L'utilizzo di un'applicazione IIS offre diversi vantaggi:

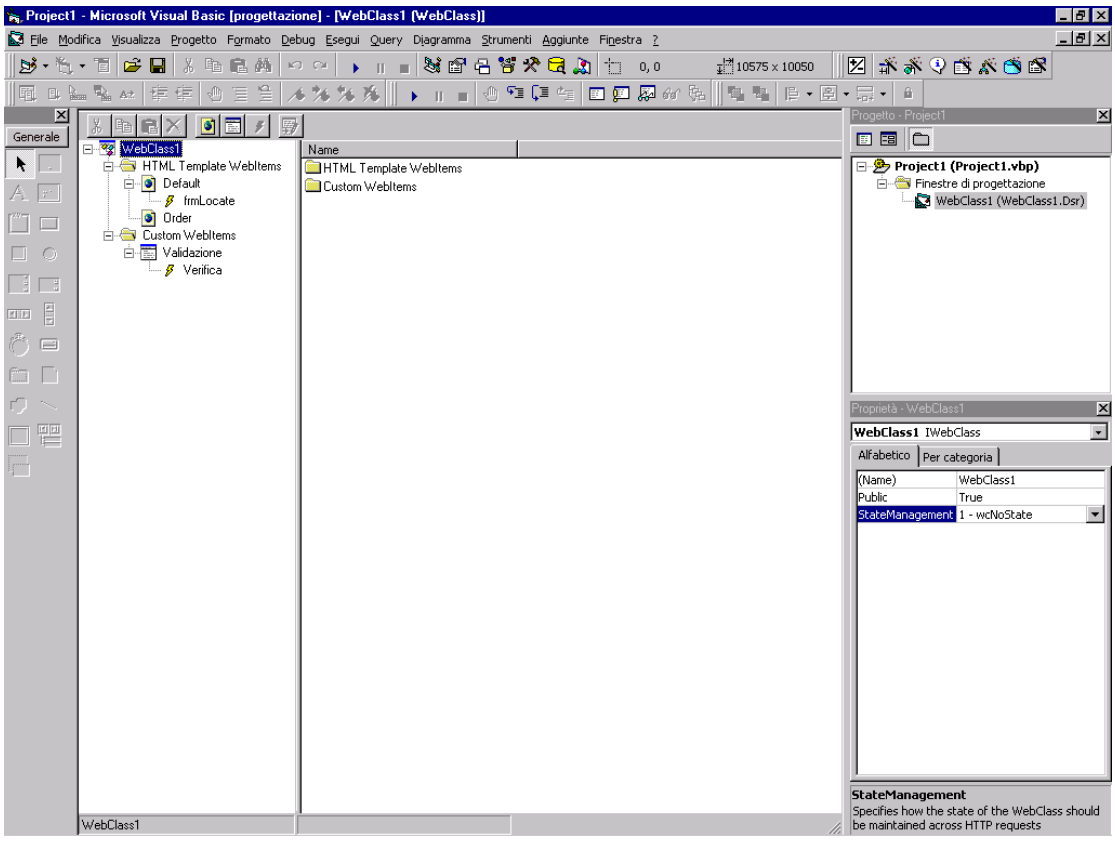
- costi di distribuzione ridotti, perché gli utenti finali necessitano unicamente di un browser;
- ambiente di sviluppo familiare;
- accesso ad una base più ampia di client grazie all'indipendenza dal browser e dalla piattaforma;
- modello ad oggetti che offre accesso diretto alle risorse di IIS;
- separazione del codice e dell'HTML perché il codice non è immerso nell'HTML come accade per le procedure di script;
- gestione automatica della persistenza dello stato.

Ogni WebClass in un'applicazione IIS è associata ad un file .ASP che Visual BASIC genera automaticamente durante la fase di compilazione o di debug. Il file .ASP ospita la WebClass sul server web e genera una componente di run-time quando l'applicazione è lanciata per la prima volta. Ogni WebClass contiene uno o più elementi detti WebItem.
Un WebItem può essere:

- un file modello (template) HTML, ossia una pagina associata alla WebClass che quest'ultima può inviare, su richiesta, ad un browser. I modelli si differenziano dalle normali pagine HTML perché contengono delle aree di testo che è possibile sostituire dinamicamente a runt-time, in modo da personalizzare le risposte;
- un WebItem personalizzato: elementi che non sono associati ad alcuna pagina, ma contengono una o più procedure di gestione dell'evento.

Sia i modelli che gli elementi personalizzati espongono eventi che la WebClass intercetta al verificarsi di determinate azioni nel browser e che è possibile gestire utilizzando codice Visual BASIC. L'interfaccia utente consiste in una serie di pagine HTML anziché di form. Una pagina HTML è simile ad una form perché contiene tutti gli elementi visuali che compongono l'interfaccia, ma è salvata in un file .HTM o .HTML. Visual BASIC crea il file .HTM dai file originali quando si salva o si esegue il debug dell'applicazione. Non è possibile creare le pagine direttamente in Visual BASIC.
Riassumendo, un'applicazione IIS consiste dei seguenti elementi:

- una o più WebClass;
- uno o più modelli HTML;
- uno o più WebItem personalizzati;
- un file .ASP che ospita la WebClass in IIS;
- una componente di run-time (MSWCRUN.DLL) che aiuta a processare le richieste;
- una DLL di progetto generata automaticamente che contiene il codice Visual BASIC ed alla quale fa accesso la componente di run-time.

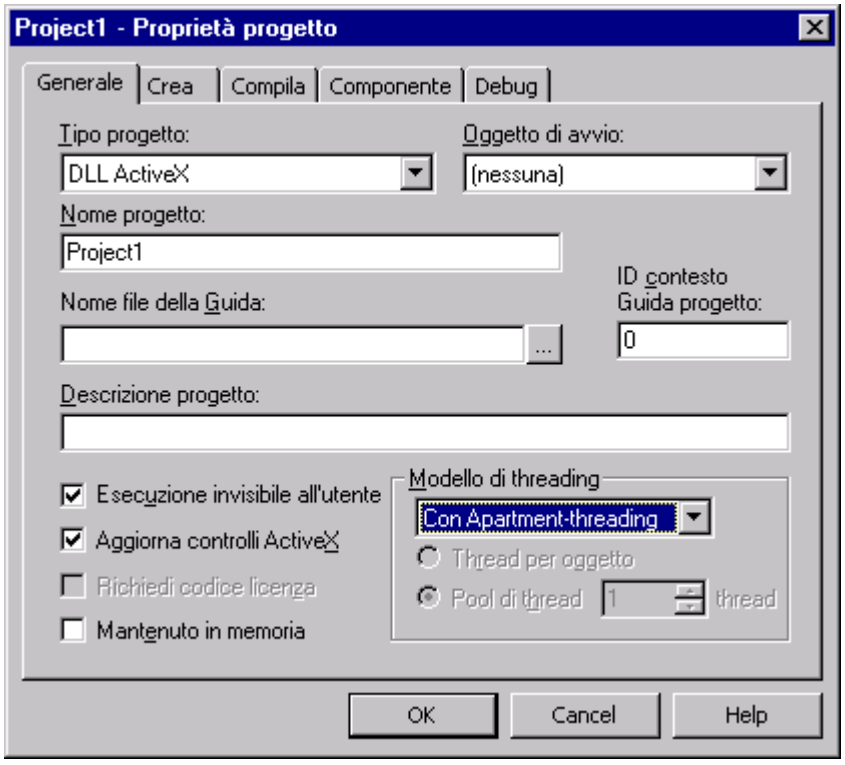


Il processo di creazione di un'applicazione IIS è il seguente:

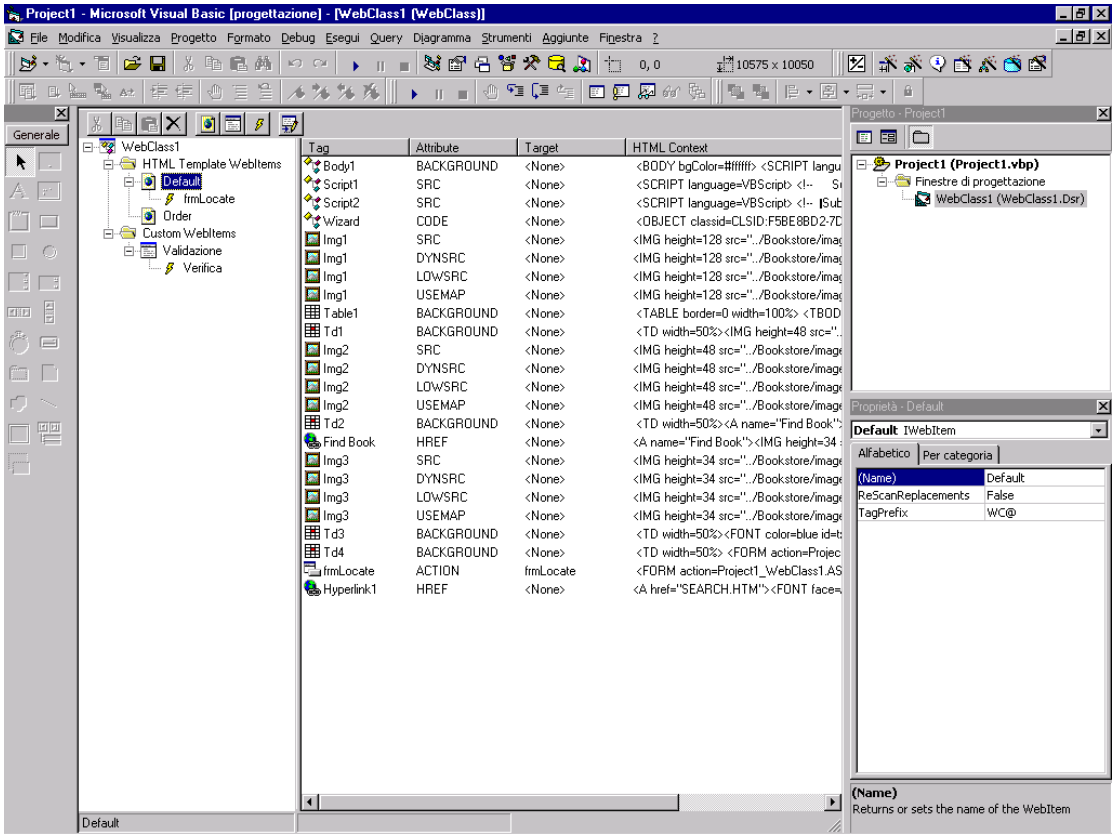
- iniziare un nuovo progetto e selezionare il modello *Applicazione IIS;*
- salvare il progetto, perché in caso contrario non sarà possibile aggiungere i modelli HTML;
- aggiungere i template HTML ed i WebItem personalizzati necessari;
- aggiungere gli eventi personalizzati al progetto;
- scrivere il codice per tutti gli eventi del progetto;
- verificare ed eseguire il debug dell'applicazione lanciando il progetto e visualizzando l'applicazione nel browser;
- compilare il progetto;
- distribuire l'applicazione.

Un'applicazione IIS è un progetto di tipo ActiveX DLL che include automaticamente un'istanza della corretta finestra di progettazione (designer).
Esistono diverse proprietà che occorre impostare per le WebClass.

- **StateManagement**: questa proprietà disponibile nella finestra delle proprietà dell'oggetto WebClass, determina se l'istanza di un oggetto deve essere mantenuta in vita tra due richieste successive o se può essere distrutta al completamento della richiesta.
- **Public**: questa proprietà deve valere True perché l'applicazione possa funzionare.
- **Esecuzione invisibile all'utente**: questa opzione disponibile nella pagina di proprietà del progetto, consente di allocare un'istanza della DLL per ogni thread.
- **Mantenuto in memoria**: consente di tenere il run-time di supporto sempre attivo in memoria, in modo da velocizzare il caricamento dell'applicazione.



Dopo aver impostato le proprietà e salvato il progetto, si aggiungono i template HTML per abilitare la WebClass all'invio di pagine HTML. Quando si aggiunge un modello, Visual BASIC esamina il file e ricerca tutti i tag HTML in grado di effettuare una richiesta al server: elementi form, tag Image e hyperlink.



Questi attributi possono diventare eventi nell'applicazione IIS. Visual BASIC elenca i tag ed i rispettivi attributi nella finestra di destra del designer.
Il nome di ogni tag è determinato sulla base del valore assegnato all'attributo *ID* del tag. Se il modello HTML contiene un tag privo dell'attributo *ID*, il disegnatore assegna un *ID*

univoco, in base alla posizione nello stream HTML.
È possibile aggiungere WebItem personalizzati per creare risorse di programmazione non basate su file. A differenza dei modelli, che sono legati alle pagine HTML, i WebItem personalizzati sono contenitori che raggruppano procedure ed aiutano a creare codice modulare e maggiormente strutturato.
Quando si modifica un modello HTML e si salva il progetto o si esegue il debug dell'applicazione, Visual BASIC salva la WebClass ed i modelli associati nella directory specificata. Quando si esegue il debug, utilizza questa directory come directory virtuale di IIS. Esistono tre tipi di eventi: eventi standard, eventi legati ai modelli ed eventi personalizzati. Ogni modello ed ogni WebItem aggiunto ad una WebClass dispone di un insieme predefinito di eventi, che compaiono automaticamente nella finestra di codice. Esistono tre eventi standard per ogni WebItem.

- **Respond**: imposta l'azione predefinita che il WebItem intraprende quando è attivato da una richiesta utente.
- **ProcessTag**: processa i tag in un modello e sostituisce il loro contenuto con i dati specificati. I tag da analizzare sono identificati dalla proprietà **TagPrefix**.
- **UserEvent**: processa gli eventi della WebClass creati in fase di run-time.

L'evento più utilizzato è sicuramente **Respond**, che è anche l'evento predefinito. Una WebClass innesca questo evento in due situazioni: quando un WebItem è attivato per la prima volta o quando riceve una richiesta da un browser e non esiste un corrispondente evento di template. Il WebItem può essere attivato sia dal client sia dal server. L'attivazione client-side si ha quando una risposta contiene un riferimento ad un altro WebItem da invocare. Un'attivazione server-side si verifica quando è impostata la proprietà **NextItem**. I tag all'interno di un modello HTML possono agire come sorgente di eventi se possiedono un attributo che contiene un URL. Quando si aggiunge un modello, Visual BASIC esamina il file e visualizza i tag in grado di eseguire richieste al server.
Per impostare un evento di template occorre operare come segue:

- selezionare l'attributo del tag che si vuole trattare come evento;
- eseguire il procedimento di connessione attivando il menu contestuale con il tasto destro e selezionando *Connect to Custom Event;*
- scrivere il codice per l'evento.

Solo gli attributi collegati compaiono nella lista Procedure della finestra di codice.
Un evento di template differisce da un evento standard per due motivi:

- gli eventi di template sono generati da un file .HTM;
- gli eventi di template devono essere collegati prima di poter essere programmati.

Si possono anche aggiungere eventi personalizzati ad un WebItem per disporre di una maggiore flessibilità di programmazione. Quando si definisce un evento personalizzato, lo si associa con un modello o con un WebItem e quindi si scrive il codice per indicare le azioni che Visual BASIC deve intraprendere allo scatenarsi dell'evento.
Il metodo **URLFor** innesca gli eventi nel browser, creando un URL che referenzia l'evento personalizzato. Così come accade per le form, anche la WebClass dispongono di un proprio ciclo di vita, diverso da quello delle form e riportato di seguito:

- l'evento **Initialize** si verifica quando un utente accede alla pagina ASP che ospita l'applicazione IIS. Durante questo evento la WebClass è creata come oggetto;
- l'evento **BeginRequest** è lanciato subito dopo e s'innesca tutte le volte che la WebClass riceve una richiesta da un browser. Questo evento segna l'inizio dell'esame della richiesta HTTP;
- l'evento **Start** si verifica la prima volta che l'evento **BeginRequest** è innescato;
- l'evento **EndRequest** si verifica quando la WebClass termina di processare la richiesta HTTP e ritorna la risposta al client;
- l'evento **Terminate** chiede alla DLL di run-time di distruggere l'istanza della WebClass.

Come si vede non esistono gli eventi **Load** ed **Unload**, i cui parenti più prossimi sono gli eventi **BeginRequest** ed **EndRequest**.
Le WebClass utilizzano gli oggetti del modello di ASP.

- **Request**: riceve le richieste da un browser.
- **Response**: invia le informazioni al client.
- **Session**: mantiene le informazioni di stato relative ad una sessione di un client.
- **Application**: gestisce le informazioni di stato da condividere tra più istanze di un oggetto WebClass.
- **Server**: crea altri oggetti.
- **BrowserType**: determina le capacità del browser.

Prima di poter rispondere ad una richiesta occorre scrivere il codice che specifica alla WebClass cosa fare all'avvio dell'applicazione. L'evento Start è il più adatto allo scopo e serve anche per lanciare con successo una WebClass: si utilizza per navigare sul primo WebItem dell'applicazione, come indicato dallo stralcio di codice seguente:

```
Private Sub Webclass_Start()
  Set NextItem=WebItemFirst
End Sub
```

Le risposte inviate da un oggetto WebClass sono costituite da un flusso di dati HTML che il browser mostra all'utente. Questo flusso può essere inviato in due modi:

- utilizzando il metodo **WriteTemplate** che invia il contenuto del file HTML direttamente al browser;
- generando lo stream HTML con codice Visual BASIC.

Il metodo **WriteTemplate** rappresenta il modo più semplice per inviare codice HTML al browser. Quando la WebClass esegue una procedura evento di template che referenzia questo metodo, invia al browser il file HTML specificato. Se si ha familiarità con l'HTML è possibile generare il codice linea per linea ed inviarlo al browser utilizzando l'oggetto **Response**. È un procedimento più complesso ma necessario quando non si dispone di un modello. Si può utilizzare invece l'oggetto **Request** per accedere alle informazioni che un utente immette in una form HTML. Ad esempio, se si connette il tag Action della form, il submit di una form può innescare un evento utilizzabile per manipolare l'informazione. Utilizzando la collezione **Form** dell'oggetto **Request** si accede ai valori dei campi sulla form.

```
Private Sub Booklist_Search()
  Private sTitle As String
  Private sAuthor As String
  Private sPublisher As String
  sTitle = Request.Form("title")
  sAuthor = Request.Form("author")
  sPublisher = Request.Form("publisher")
End Sub
```

La sostituzione di blocchi di testo è utile quando si generano dinamicamente parti di codice HTML, ad esempio se si vuole personalizzare una pagina Web con un nome specificato dall'utente in una pagina precedente.

Si può inserire un indicatore di sostituzione in un file .HTM e rimpiazzarlo con il nome dell'utente in fase di esecuzione. Questo processo implica l'esame di un file modello per individuare i marcatori e la successiva sostituzione con il testo appropriato. Il tutto avviene automaticamente quando si invoca il metodo **WriteTemplate**.

I tag processati sul server dispongono delle seguenti proprietà:

- un prefisso, un insieme di caratteri che indica alla WebClass che è necessaria un'operazione di sostituzione. Il valore predefinito è **WC@** modificabile, per ogni template, tramite la proprietà **TagPrefix**;
- un nome per il tag che identifica l'area di sostituzione. Il nome è solitamente diverso per ogni zona del template soggetto alla sostituzione. Quando si desidera eseguire la stessa sostituzione in più punti, si può utilizzare lo stesso nome;
- **TagContent**, il contenuto del tag prima della sostituzione.

```
<HTML>
<BODY>
```

```
<P>L'utente <WC@UID>UserName<WC@UID> è stato autorizzato alla transazione
<P>
</BODY>
</HTML>
Sub WebClass1_ProcessTag (ByVal TagName as String, _
                          TagContents As String,SendTags As Boolean)
  If TagName = "WC@UID" Then
    TagContents = sUserName
  ElseIf TagName = "WC@PWD" Then
    …
  End If
End Sub
```

WC@ prefisso
UserName contenuto
WC@UID nome

Questi elementi forniscono alla WebClass gli elementi per eseguire le sostituzioni. Il codice necessario è scritto nell'evento **ProcessTag,** invocato dal metodo **WriteTemplate**. Le operazioni eseguite sono.

- Visual BASIC interpreta il codice ed invoca il metodo **WriteTemplate** da una procedura di gestione dell'evento, di solito l'evento **Respond**.
- La WebClass interpreta e sostituisce i tag dotati di prefisso sulla base del codice scritto nell'evento **ProcessTag**.
- La WebClass invia il modello tramite l'oggetto **Response**.
- Il metodo **WriteTemplate** invia il modello al browser.

Nell'esempio precedente, il contenuto del tag WC@UID è sostituito con il nome contenuto nella variabile sUserName prima che il template sia inviato all'utente. La proprietà **NextItem** si utilizza per navigare da un WebItem all'altro. Normalmente la WebClass esegue i passi seguenti quando riceve una richiesta:

- intercetta la richiesta e la fa corrispondere ad un WebItem dell'applicazione;
- identifica l'evento corretto e lo innesca;
- processa il codice per l'evento;
- ritorna la risposta al browser.

NextItem è utile per aggiungere un ulteriore passo a questo processo. Dopo che la WebClass ha eseguito il codice per l'evento, la proprietà **NextItem** passa il controllo ad un altro WebItem che può eseguire ulteriore codice prima di inviare la risposta all'utente. Nel caso in cui l'applicazione IIS contenga più WebClass si può utilizzare il metodo **Redirect** dell'oggetto **Response** per navigare da una WebClass all'altra.

```
Private Sub Order_Search()
  If Request.Form("ACTION") = "Search" Then
    Set NextItem = OrderStatus
  Else
    Response.Redirect "http://www.mioserver.com/miadir/mioasp.asp
  End If
End Sub
```

Oltre a gestire gli eventi che esistono in fase di disegno è possibile creare e gestire eventi in fase di run-time, utilizzando il metodo **URLFor** che riceve due argomenti: il nome del WebItem ed il nome di un evento non definito in fase di run-time.
Il codice per l'evento si scrive nella procedura **UserEvent**.

```
Response.Write "A HREF = """ & URLFor(Response, "MioEvento") & """>"
Private Sub Response_UserEvent (ByVal EventName As string)
  If EventName = "MioEvento" Then
    …
  End If
End Sub
```

Infine ricordiamo che il protocollo HTTP è stateless e quindi occorre utilizzare tecniche particolari quando si vuole che l'applicazione ricordi un'informazione tra due richieste successive. Esistono diversi modi per memorizzare lo stato di un'applicazione Web:

- utilizzare l'oggetto WebClass o altri oggetti;
- utilizzare un database;
- muovere un'informazione avanti e indietro tra il server ed il browser utilizzando cookies o campi nascosti.

Ogni approccio ha vantaggi e svantaggi. Memorizzare lo stato negli oggetti è facile ma diminuisce la scalabilità perché il server deve tenere l'oggetto istanziato tra due richieste e localizzare lo stesso oggetto ad ogni richiesta successiva. L'utilizzo dei database aumenta la scalabilità ma richiede la gestione di connessioni e recordset. Trasferire le informazioni di stato permette di evitare la memorizzazione di queste informazioni, ma aumenta la richiesta di larghezza di banda ed è meno sicuro.

DOCUMENTI ACTIVEX

Se si desidera sviluppare una pura applicazione Internet, per iniziare è possibile usare i documenti ActiveX. Un documento ActiveX ha lo stesso aspetto e comportamento di una normale applicazione Visual BASIC in una finestra del form, a parte il fatto che invia controlli ActiveX al PC dell'utente se la macchina non contiene i controlli ActiveX utilizzati dal documento (il documento appare all'utente come una normale pagina HTML). Il documento ActiveX può contenere collegamenti ipertestuali (controlli ActiveX che sono scaricati o sono già presenti, a seconda del contenuto della macchina dell'utente). I menu del documento ActiveX sostituiscono quelli dell'applicazione in cui esso è ospitato. Il documento ActiveX si collega ad una pagina HTML che si crea o si utilizza. Probabilmente la più importante ragione per creare documenti ActiveX è che Internet Explorer li può eseguire come se fosse un programma di controllo o un launcher di programmi di un sistema operativo. I menu del documento ActiveX si fondono con quelli di Internet Explorer (sovrascrivendo eventualmente le sue funzionalità), e non si deve imparare un nuovo linguaggio, come Java, per attivare le pagine Web. La finestra di dialogo *Nuovo progetto* contiene due icone, *EXE ActiveX* e *DLL ActiveX*, che creano shell di documenti ActiveX. Dopo avere avviato la creazione di un documento ActiveX, è possibile aggiungere qualunque caratteristica desiderata alla finestra del form, come per le normali applicazioni.

OLE

Inserire nel form OLE dalla Casella degli strumenti, è visualizzata la casella di riepilogo Inserisci oggetto selezionare quindi il tipo di oggetto, Grafico di Microsoft Graph. Visual BASIC apre un grafico in cui si possono inserire i dati desiderati, alla fine si esce con Chiudi. Quando si esegue l'applicazione, il grafico è visualizzato nel controllo OLE, inoltre Grafico di Microsoft Graph è aperto automaticamente in modo che sia possibile modificare i dati rappresentati nel grafico.

DDE (DYNAMIC DATA EXCHANGE)

Il DDE funziona tramite gli Appunti; esistono due modi di aprire un collegamento client:

1. automatico, i dati sono aggiornati automaticamente nel client ogni volta che subiscono delle modifiche nel server;
2. manuale, i dati sono aggiornati solo su richiesta del client.

Se si volessero leggere i dati di Excel (server), ci si dovrebbe configurare come client; in Visual BASIC ci sono tre controlli che possono fungere da client, ovvero ricevere dati DDE, le Texbox, le Picturebox e le Label. Invece, solamente le form possono fungere da server ed inviare dati DDE: non possono essere manuali o automatici.
È meglio usare OLE per i seguenti motivi.

- i dati non sono modificabili;
- non si scambia la grafica (solo dati);
- le prestazioni sono inferiori.

Collegamento client in fase di creazione

In fase di disegno si possono stabilire solo collegamenti automatici e non manuali. Per esempio, copiare del testo da Word (server) tramite Modifica/Copia quindi si selezioni Text1 (client) e si scelga Edit/Paste link. Le informazioni relative a questo collegamento sono memorizzate nelle proprietà di Text1: LINKMODE, LINKTOPIC contiene sia il nome dell'applicazione sia il nome del file di dati, LINKITEM contiene il nome dell'elemento particolare, per Excel R1C1, LINKTIMEOUT specifica il periodo di tempo, in decimi di secondo, entro il quale deve essere stabilito il collegamento DDE.

Collegamento server in fase di creazione

Da una Textbox selezionare Edit/Copy, da Word Modifica/Incolla collegamento. Da notare che Text1.LinkMode = 0, ma Form1.LinkMode = 1 (server), Form1.LinkTopic = Form1 (applicazione DDE).

Collegamento automatico client durante l'esecuzione

```
Sub Check1_Click ()
        Text1.LinkMode = NONE
        Text1.LinkTopic = "Excel|Foglio1"
        Text1.LinkItem = "R1C1"
        Text1.LinkMode = LINK_AUTOMATIC
End Sub
```

Si può modificare LinkItem durante l'esecuzione, basta utilizzare un contatore per sapere quante volte si fa clic nella Checkbox; in questo modo si evita l'interruzione del collegamento con il server prima di passare ad un nuovo elemento.

Collegamento manuale client durante l'esecuzione

```
Sub Check1_Click ()
        Text1.LinkMode = NONE
        Text1.LinkTopic = "Excel|Foglio1"
        Text1.LinkItem = "R1C1"
        Text1.LinkMode = LINK_MANUAL
```

```
End Sub
Sub Form_Click ()
    Text1.LinkRequest
End Sub
```

Si prelevano i dati semplicemente facendo clic nella form.

Collegamento server durante l'esecuzione

Dall'applicazione Paintbrush Modifica/Copia, da Visual BASIC si crei una Picturebox con AUTOREDRAW = TRUE e si faccia Edit/Paste. Clic su form1

```
Sub Form_Click ()
    Static first
    first = first + 1
    If first = 1 Then
        LinkTopic = "Figure"
        LinkMode = LINK_SOURCE
    Else
        x1 = Picture1.ScaleWidth * Rnd
        y1 = Picture1.ScaleHeight * Rnd
        x2 = Picture1.ScaleWidth * Rnd
        y2 = Picture1.ScaleHeight * Rnd
        Picture1.Line (x1,y1) - (x2,y2), , B
    End If
End Sub
```

Quando si assegna il nome al file source.exe, diventa anche il nome dell'applicazione DDE. Se le applicazioni la devono usare come server, dovranno rivolgersi ad un'applicazione che ha come LinkTopic = source|form1.

Si può ora sviluppare un'applicazione client che legga l'immagine in source.exe. Lanciare source.exe ed aprire un nuovo progetto con una Picturebox con *autoredraw=true* e *autosize=true* quindi si fa clic nella form, l'immagine deve essere trasferita dal server al client.

```
Sub Form_Click ()
    Picture1.LinkMode = NONE
    Picture1.LinkTopic = "Source|Figura"
    Picture1.LinkItem = "Picture1"
    Picture1.LinkMode = LINK_AUTOMATIC
End Sub
```

Con un collegamento automatico il server invia al client l'intera immagine ogni volta che si è modificato anche un solo pixel. Visual BASIC risolve il problema inviando gli aggiornamenti solo quando il server lo richiede tramite LINKSEND.

Si supponga, per esempio, che al primo clic source.exe s'imposti come server, al successivo clic s'inseriscono dei rettangoli casuali. Si esegua l'applicazione client, si noterà che i rettangoli non appaiono nella Picturebox; si può correggere questo problema inserendo in source frm un Commandbutton con codice:

```
Sub Command1_Click ()
    Picture1.LinkSend
End Sub
```

per inviare al client una versione aggiornata dell'immagine.

Il metodo LinkPoke

Invia i dati dal client al server, solitamente i dati transitano dal server al client in modo automatico o manuale. Tuttavia con questo metodo è possibile invertire la direzione del flusso dei dati.

Modificare l'applicazione client in modo che al primo clic si stabilisca il collegamento, al secondo clic si disegni un cerchio modificando la figura inviata dal server. A questo punto il cerchio è nel client e non nel server; tuttavia, si può, temporaneamente invertire la direzione del collegamento DDE con LinkPoke.

```
Sub Form_Click ()
    Static first
    first = first + 1
    If first = 1 Then
        Picture1.LinkMode = NONE
        Picture1.LinkTopic = "Source|Figura"
        Picture1.LinkItem = "Picture1"
        Picture1.LinkMode = LINK_AUTOMATIC
    Else
        x = Picture1.ScaleWidth / 2
        y = Picture1.ScaleHeight / 2
        r = Picture1.Width / 5
        Picture1.Circle (x, y), r
        Picture1.LinkPoke
    End If
End Sub
```

Il metodo LinkSend

Invia dati dal server al client.

Il metodo LinkExecute

Invia comandi ad un server DDE.

```
Sub Check1_Click ()
    Text1.LinkMode = NONE
    Text1.LinkTopic = "Excel|Foglio1"
    Text1.LinkItem = "R1C1"
    Text1.LinkMode = LINK_AUTOMATIC
    Text1.LinkExecute "[file.apri(""c:\excel\foglio.xls"")][bip][file.chiudi()][file.esci()]"
End Sub
```

Non tutte le applicazioni Windows che supportano il DDE possono accettare comandi in questo modo, ad esempio Word non è in grado di farlo.

Gestione degli errori DDE

Cosa accade se l'altra applicazione non è in esecuzione?

Si otterrebbe un messaggio di errore, che è catturabile per cui è possibile predisporre un sistema di gestione.

```
Sub Check1_Click ()
    On Error GoTo pippo
    Text1.LinkMode = NONE
    Text1.LinkTopic = "Excel|Foglio1"
    Text1.LinkItem = "R1C1"
    Text1.LinkMode = LINK_AUTOMATIC
    Text1.LinkExecute "[file.apri(""c:\excel\foglio.xls"")][bip][file.chiudi()][file.esci()]"
    Exit Sub
Pippo:
    If Err = 28 Then
        temp = Shell ("c:\excel\excel.exe")
        Resume
    Else
        MsgBox "Errore numero" + Str$(Err)
        Exit Sub
    End If
End Sub
```

Quando si verifica un errore DDE è generato un evento LinkError, associato alle form, alle Textbox, alle Picturebox ed alle label. Per esempio, due errori DDE molto frequenti sono:

1. 7, numero eccessivo di collegamenti;
2. 11, memoria esaurita.

```
Sub Text1_LinkError (LinkError As Integer)
    If LinkError = 7 Then
        MsgBox “Troppi collegamenti DDE”
    End If
    If LinkError = 11 Then
        MsgBox “Memoria esaurita per il DDE”
    End If
End Sub
```

Si noti che se l’applicazione fosse stata server gli errori di questo tipo avrebbero dovuto essere intercettati dalla form.

L’evento LinkOpen

È generato quando è stabilito un collegamento, inoltre può rifiutarlo perché il suo codice accetta come argomento Cancel impostato a True.

```
Sub Form_LinkOpen (Cancel As Integer)
```

L’evento LinkClose

È generato quando il collegamento è chiuso. Entrambi si verificano per l’applicazione server o per quella client.

Incolla collegamento nel codice (client) e copia nel codice (server)

Si supponga di avere una form con due Textbox e di voler permettere all’utente di stabilire un collegamento tra la form ed un’applicazione esterna a sua scelta, in pratica non è possibile prevedere con quale applicazione sarà stabilito il collegamento. Quando l’utente seleziona Modifica/Incolla collegamento, vuole incollare nell’applicazione, dati che si trovano negli Appunti: memorizzati in una zona riservata al DDE.

```
Sub PasteItem_Click ()
DDEData$ = Clipboard.GetText (CF_LINK)
Screen.ActiveControl.LinkMode = NONE
Screen.ActiveControl.LinkTopic = Left$ ((DDEData$, InStr (DDEData$,”!”)-1)
Screen.ActiveControl.LinkItem=Right$(DDEData$,Len(DDEData$)-InStr(DDEData$,”!”))
Screen.ActiveControl.LinkMode = LINK_AUTOMATIC
End Sub
```

La costante CF_LINK indica che si vogliono prelevare dati DDE, quindi si deve sapere quale delle due Textbox possiede il focus. Successivamente si devono decifrare i nomi dell’oggetto e dell’elemento DDE: Applicazione|Oggetto!Elemento.

Si dà, ora, la possibilità all’utente d’impostare un collegamento server con una qualunque applicazione che supporti il DDE. Si devono caricare i dati DDE negli Appunti con Modifica/Copia.

```
Sub CopyItem_Click ()
    Clipboard.Clear
    DDEData$ = “Custom|Form1!” + Screen.ActiveControl.Tag
    Clipboard.SetText DDEData$, CF_LINK
    If TypeOf Screen.ActiveControl Is TextBox Then
        Clipboard.SetText Screen.ActiveControl.Text
    Enf If
    If TypeOf Screen.ActiveControl Is Label Then
        Clipboard.SetText Screen.ActiveControl.Caption
    Enf If
    If TypeOf Screen.ActiveControl Is PictureBox Then
        Clipboard.SetText Screen.ActiveControl.Picture
    Enf If
End Sub
```

Per prima cosa si cancellano gli Appunti, quindi si assembla la stringa DDEData$. Il problema è sapere quale Texbox fornisce i dati, Visual BASIC mette a disposizione la proprietà TAG che è una stringa per ciascun oggetto: Text1.Tag = Text1. A questo punto s’inserisce DDEData$ negli Appunti, inoltre è possibile verificare il tipo di controllo attivo

tramite TypeOf.

DLL (DYNAMIC LINK LIBRARY)

Le DLL non sono nient'altro che controlli ActiveX senza elementi grafici, possono essere utilizzate da qualsiasi applicazione e più applicazioni utilizzano una stessa libreria in fase di esecuzione. Per progettare una DLL in Visual BASIC bisogna selezionare.

File/Nuovo progetto/DLL ActiveX

A questo punto nella finestra di progettazione compare *Class1* che indica il nome della classe con le sue proprietà, la più importante è *Instancing* che può essere settata come.

1. Private: se si costruisce un oggetto dalla libreria DLL si possono generare istanze solo dal progetto di cui fanno parte. Quindi almeno una delle classi deve essere non privata altrimenti non è possibile utilizzare il codice.
2. PublicNotCreatable: consente ad altre applicazioni di utilizzare le istanze esistenti della classe, ma non di creane delle nuove. In pratica, le classi generate sono visibili e utilizzabili da tutti, ma sono generate solo da un progetto.
3. Multiuse: consente di realizzare diversi oggetti di quella classe. Chiunque può utilizzare una certa classe e generare delle istanze.
4. Global Multiuse: è possibile generare diverse istanze della classe, ma se si genera un riferimento ad una DLL non serve usare la parola chiave *New* per generare l'istanza, in quanto è come se le funzionalità della classe fossero già integrate nella propria applicazione.

Scrittura del codice della DLL, la classe si chiama Class1 e il progetto MiaDLL.VBP.

```
Option Explicit
'Nome da inserire nell'intestazione della MsgBox
Const scuola = "ITIS Specializzazione Informatica"
Public Sub MiaMsgBox(Frase As String)
    'Metodo pubblico della classe in modo che sia richiamabile dall'esterno
    MsgBox Frase, , scuola
End Sub
```

File/Crea MiaDLL.dll nella cartella Windows

Adesso si crea un nuovo progetto che utilizzerà questa DLL.

File/Nuovo progetto/EXE standard

Progetto/Riferimenti/Sfoglia

Dopo che la DLL è stata selezionata come riferimento esterno, sarà possibile per qualsiasi elemento del progetto richiamare i suoi metodi o le sue proprietà pubbliche come se si trovasse all'interno del progetto stesso.

```
Private Sub Command1_Click()
    MiaMsgBox "Questa è una prova di DLL"
End Sub
```

Per poter utilizzare una DLL è necessario dire prima al progetto che questa DLL esiste tramite il menu **Progetto/Riferimenti**. È fastidioso ripetere questa procedura tutte le volte che si crea un nuovo progetto, allora è possibile registrare la DLL, ovvero fare in modo che sia compresa nell'elenco delle DLL che il sistema è in grado di vedere senza alcuna operazione aggiuntiva e che di conseguenza compaia nella finestra *Riferimenti* senza dover fare nulla. Per fare questo le informazioni sulla DLL devono essere aggiunte al registro di configurazione utilizzando il comando in una sessione MS-DOS.

Regsvr32 percorso\nomedll

Nel caso in cui si voglia fare in modo che non sia più registrata si usa lo stesso comando.

Regsvr32 /U percorso\nomedll

API (APPLICATION PROGRAMMING INTERFACE)

Il sistema operativo Windows consente di accedere ad un insieme elevato di funzionalità grazie alle API, usate per controllare l'aspetto ed il comportamento di ogni elemento del sistema operativo (dalla visualizzazione del desktop, alla allocazione della memoria per un

nuovo processo). Ogni azione dell'utente provoca l'esecuzione di una o più funzioni API che informano Windows su cosa stia accadendo. Quando ad esempio nel codice scriviamo Form1.Print Visual BASIC non fa altro che richiamare la funzione API TextOut scambiandosi i parametri necessari. Le funzioni API risiedono in DLL (User32.dll, GDI32.dll, Shell32.dll) installate nella directory di sistema. Con Visual BASIC è possibile accedere a questa interfaccia dichiarando queste routine come esterne ed utilizzandole come qualsiasi altra funzione. Si tratta di routine scritte in linguaggio C, con dichiarazioni molto complesse, ma Visual BASIC mette a disposizione uno strumento efficace per la realizzazione di chiamate alle API che si chiama **Visualizzatore API**.

C:\Programmi\Microsoft Visual Studio\Common\Tools\Winapi\APILOAD.EXE"

Il Visualizzatore API consente di visualizzare le dichiarazioni, le costanti e i tipi inclusi in un file di testo API o in un database Jet. Gli elementi che si desidera utilizzare possono essere copiati negli Appunti e quindi incollati nel codice Visual BASIC. Il Visualizzatore API contiene il file di testo Win32api.txt per le interfacce di Win32. Per rendere più rapida la visualizzazione del contenuto dei file e quindi la ricerca, è possibile convertire questi file in database Jet. Per convertire i file .TXT in file di database Jet, è sufficiente caricare il file .TXT desiderato e scegliere Converti testo in database dal menu File. Dopo aver convertito il file .TXT in un database Jet, è consigliabile caricare sempre il database per ottenere le prestazioni migliori. Per caricare un file API, scegliere Carica file di testo o Carica file database dal menu File. È inoltre possibile caricare un file di testo o un file database specificando il file desiderato nella riga di comando utilizzando i seguenti parametri:

APILOD32.EXE [/T | /D] nomefile

La casella di riepilogo nella parte superiore della finestra consente di visualizzare le costanti, le dichiarazioni e i tipi. Selezionare un elemento e scegliere il pulsante Aggiungi. L'elemento selezionato sarà aggiunto nel riquadro Elementi selezionati disponibile nella parte inferiore della finestra. Per copiare negli Appunti gli elementi contenuti nel riquadro Elementi selezionati, fare clic su Copia. Tutti gli elementi disponibili nell'elenco saranno copiati negli Appunti. Sarà quindi possibile inserire tali elementi nei moduli scegliendo Incolla dal menu Modifica quando il modulo desiderato ha lo stato attivo. Per rimuovere una voce dall'elenco degli elementi selezionati, fare clic sul pulsante Rimuovi.

Il file Mapi32.txt è invece utilizzato con la Messaging API, che permette di aggiungere alle applicazioni funzionalità correlate alla posta elettronica.

Come esempio di potenzialità delle API si realizza un semplice progetto in grado di fornire tutta una serie d'informazioni sul mouse che è installato sul proprio sistema.

Naturalmente è necessario conoscere il nome della API, *GetSystemMetrics*, che si deve utilizzare altrimenti non sarebbe possibile selezionarla nel Visualizzatore API.

File/Nuovo progetto/EXE standard
Start/Tutti i programmi/Microsoft Visual BASIC 6.0/API Text Viewer
File/Carica file di testo/Win32API.txt
Tipo API: dichiarazioni
Elementi disponibili: GetSystemMetrics/AggiungiCopia
Modifica/Incolla nella sezione dichiarazioni del form de progetto.

Se in compilazione compare il messaggio "… non ammessi come membri Public …" basta dichiarare l'API Private.

```
Option Explicit
Private Declare Function GetSystemMetrics Lib "user32" (ByVal nIndex As Long) As Long
Const sm_mousepresent = 19
Const sm_cmousebuttons = 43
Const sm_swapbutton = 23
Private Sub Command1_Click()
    'verifica se il mouse è presente
   If GetSystemMetrics(sm_mousepresent) Then
      MsgBox "Il mouse è installato correttamente", vbExclamation, "AVVISO"
```

```
    End If
End Sub
Private Sub Command2_Click()
    'verifica il numero di bottoni
    Text1.Text = GetSystemMetrics(sm_cmousebuttons)
End Sub
Private Sub Command3_Click()
    'verifica se i bottoni destro e sinistro siano stati invertiti da un utente mancino
    If GetSystemMetrics(sm_swapbutton) Then
        MsgBox "Il mouse ha i tasti invertiti", vbExclamation, "AVVISO"
    Else
        MsgBox "Il mouse non ha i tasti invertiti", vbExclamation, "AVVISO"
    End If
End Sub
```

Muovere il mouse in una zona particolare dello schermo

```
Private Type rect
        Left As Long
        Top As Long
        Right As Long
        Bottom As Long
End Type
Private Type POINTAPI
        x As Long
        y As Long
End Type
Private Declare Function ClientToScreen Lib "user32" (ByVal hwnd As Long, lpPoint As POINTAPI) As Long
Private Declare Function GetClientRect Lib "user32" (ByVal hwnd As Long, lpRect As rect) As Long
Private Declare Function ClipCursor Lib "user32" (lpRect As Any) As Long
```

Con gli elementi precedenti definiremo un'area rettangolare (intorno alla form) che delimita la zona franca per il mouse. Si effettui quindi la descrizione degli elementi. Rect è una struttura che definisce il punto superiore sinistro e il punto inferiore destro di un rettangolo. Per capirci, un rettangolo ha quattro vertici, però per individuarlo ne bastano due, per esempio il vertice superiore di sinistra e il vertice inferiore di destra. La funzione ClientToScreen converte le coordinate di un punto espresse nel sistema di riferimento di una form in quelle dello schermo. Le coordinate dello schermo sono relative al punto superiore sinistro dello schermo, mentre quelle della form sono espresse rispetto al suo punto superiore sinistro (Top). Il parametro hwnd rappresenta la form che definisce il sistema di coordinate da convertire, mentre lpPoint è un parametro di tipo Pointapi, che contiene le coordinate da convertire. Inoltre, le nuove coordinate sono inserite in questo parametro se la funzione ha successo. La funzione GetClientRect restituisce le coordinate dell'area Client della Form. I parametri della funzione sono.

hWnd che rappresenta la finestra su cui si sta operando,
lpRect che è una struttura Rect che conterrà gli estremi dell'area client.

La funzione ClipCursor confina il mouse in un'area di forma rettangolare specificata in lpRect: in altre parole, dopo l'esecuzione di quest'istruzione, il mouse è costretto a muoversi nel rettangolo definito attraverso il parametro. Per provare l'effetto di queste istruzioni in un bottone (di nome area) inserire le seguenti istruzioni.

```
Private Sub area_Click()
 Dim mypoint As POINTAPI
 Dim myrect As rect
 mypoint.x = 0
```

```
 mypoint.y = 0
 ClientToScreen Me.hwnd, mypoint
 GetClientRect Me.hwnd, myrect
 myrect.Bottom = myrect.Bottom + mypoint.Y
 myrect.Left = myrect.Left + mypoint.X
 myrect.Right = myrect.Right + mypoint.Y
 myrect.Top = myrect.Top + mypoint.Y
 ClipCursor myrect
End Sub
```

In sintesi, con la procedura precedente sono fatte le seguenti azioni.

1. Sono definite le variabili necessarie: una di tipo Punto e l'altra di tipo rettangolo.
2. Sono convertite le coordinate riferite alla form in quelle riferite allo schermo (ClientToScreen).
3. È catturato il sistema di riferimento della form ed inserito in un rettangolo (GetClientRect).
4. È definito il rettangolo in cui sarà costretto a muoversi il mouse. A tal fine utilizziamo le dimensioni del rettangolo creato con l'istruzione GetClientRect e le coordinate dei punti definiti con ClientToScreen.
5. Infine sulla base delle impostazioni precedenti il mouse è relegato in un rettangolo.

Per riportare il mouse nelle condizioni precedenti bisogna passare il valore 0& (corrispondente a Null) alla ClipCursor, come espresso nella seguente funzione.

```
Private Sub Command1_Click()
 ClipCursor ByVal 0&
End Sub
```

Il rivelatore di dischi

Supponiamo di voler stabilire il numero e il tipo di dischi presenti nel sistema. Per rilevare il tipo di disco utilizzeremo GetDriveType, per mostrare i drive rilevati utilizzeremo MessageBox, infine per conservare l'elenco dei drive rilevati dichiareremo un array di un tipo definito dall'utente. Il tipo da definire è Dischi.

```
Private Type Dischi
   Lettera As String
   Nome As String
End Type
```

Il tipo contiene due elementi stringa: Lettera, in cui inseriremo il nome del root del drive e Nome, in cui inseriremo il tipo di drive. La funzione GetDriveType della libreria "kernel32", che restituisce il tipo di Drive la cui root è specificata in nDrive.

```
Private Declare Function GetDriveType Lib "kernel32"
Alias "GetDriveTypeA" (ByVal nDrive As String) As Long
```

I valori, principali, che restituisce, la funzione, sono:

```
Private Const DRIVE_REMOVABLE = 2
'rilevato un Floppy
Private Const DRIVE_FIXED = 3
'Disco rigido
Private Const DRIVE_REMOTE = 4
'Disco di rete
Private Const DRIVE_CDROM = 5
'CDROM
```

REGEDIT

È un registro in cui il sistema operativo memorizza tutti i parametri di configurazione hardware e software del PC. I dati contenuti sono denominati key. Per visualizzare il contenuto del registro digitare **regedit.exe**.

Il registro è suddiviso in sezioni strutturate ad albero distinte, ognuna delle quali contiene i

dati relativi ad un aspetto hardware o software del PC. La gestione deve essere effettuata dall'amministratore del sistema. Il programmatore può utilizzare questo registro per memorizzare e recuperare tutti i dati di configurazione di un'applicazione. Per questo motivo, il Visual BASIC dispone delle seguenti routine.
Per scrivere una key: *SaveSetting "Applicazione", "Sezione", "key", valore*
Per leggere una key: *valore = GetSetting "Applicazione", "Sezione", "key"*
Il percorso riservato alle applicazioni utente sviluppate in Visual BASIC, è:
HKEY_CURRENT_USER\Software\VB and VBA program Setting\Applicazione\Sezione

CONTROLLO WINSOCK (WINDOWS SOCKET)

INTRODUZIONE

Il controllo Winsock, progettato nel 1991, consente di connettersi a un PC remoto e di scambiare dati utilizzando il protocollo **UDP** (*User Datagram Protocol*) o **TCP** (*Transmission Control Protocol*). Entrambi i protocolli consentono di creare applicazioni client e server.
Analogamente al controllo timer, il controllo Winsock non presenta un'interfaccia visibile in fase di esecuzione.
La progettazione del TCP ha risposto a problemi relativi a come i PC devono parlare gli uni con gli altri. Il problema ancora più grande però è come devono parlare le applicazioni tra con le altre applicazioni.
L'operazione che avviene tra client e server è la seguente.

1. Il PC server è in attesa di richieste di comunicazione sulla porta indicata;
2. il PC client richiede al server l'autorizzazione ad iniziare una comunicazione;
3. una volta accettata la richiesta del client, il PC server invia i dati;
4. terminato lo scambio di dati, la comunicazione è chiusa;
5. il PC server torna in uno stato di attesa di eventuali richieste di comunicazione sulla porta indicata.

Un socket è un oggetto attraverso il quale un'applicazione che ne fa uso in modo specifico invia o riceve dati attraverso la rete con altri socket che fanno uso dell'I**PS** (*Internet Protocol Suite*). I socket sono bidirezionali ossia consentono allo stesso momento un flusso di dati sia in entrata che in uscita. Esistono due tipi di socket.

1. **Stream sockets** che consentono un flusso di dati continuo ed illimitato con la garanzia di evitare duplicazione di dati inviati.
2. **Datagram sockets** non garantiscono che l'ordine di ricezione dei dati sia lo stesso di quello di invio col rischio di non poter evitare duplicazione di dati (cioè stessi pacchetti di dati possono arrivare a destinazione più volte).

LOCALHOST

Ogni interfaccia di rete connessa a Internet ha un indirizzo IP, lo standard prevede per ogni PC anche un'interfaccia interna, chiamata di **loopback**, il cui indirizzo è 127.0.0.1, (maschera 255.0.0.0, indirizzo di rete 127.0.0.0) usata dalle applicazioni client e server per parlarsi direttamente, anche in assenza di connessioni esterne.
Il protocollo TCP permette di gestire contemporaneamente più connessioni su una stessa interfaccia, associando ad ognuna di esse una porta, in pratica un indirizzo logico differente.
Quando è avviato, ogni server si pone in ascolto su una porta prestabilita, per esempio la numero 80 per HTTP, la 21 per FTP.
Le porte TCP attive si controllano con il prompt di DOS, con il seguente comando.

C:\>netstat -a

Connessioni attive

Proto	*Indirizzo locale*	*Indirizzo esterno*	*Stato*
TCP	*ubertini:smtp*	*0.0.0.0:0*	*LISTENING*
TCP	*ubertini:http*	*0.0.0.0:0*	*LISTENING*
TCP	*ubertini:epmap*	*0.0.0.0:0*	*LISTENING*
TCP	*ubertini:https*	*0.0.0.0:0*	*LISTENING*
TCP	*ubertini:microsoft-ds*	*0.0.0.0:0*	*LISTENING*
TCP	*ubertini:1025*	*0.0.0.0:0*	*LISTENING*
TCP	*ubertini:1027*	*0.0.0.0:0*	*LISTENING*
TCP	*ubertini:1029*	*0.0.0.0:0*	*LISTENING*

TCP ubertini:5000 0.0.0.0:0 LISTENING
*UDP ubertini:microsoft-ds *:**
*UDP ubertini:1026 *:**
*UDP ubertini:1028 *:**
*UDP ubertini:3456 *:**
*UDP ubertini:ntp *:**
*UDP ubertini:1900 *:**
C:\>

PROGRAMMAZIONE

Nella programmazione client-server con il controllo Winsock è il client che avanza sempre la richiesta per una connessione. Il server è sempre in ascolto per una richiesta del client. Al ricevimento di una richiesta del client, il server risponde prima alla richiesta iniziale. Quando sono connessi esiste una relazione paritetica il cui il client o il server possono richiedere informazioni uno all'altro.
L'applicazione client ha la capacità di connettersi al controllo Winsock e di trasferire dati. L'applicazione server ha la capacità di accettare una o più connessioni e d'indicare lo stato di ogni connessione quando questa è creata.
Un **thick client** è un'applicazione che esegue il lavoro di elaborazione sul singolo PC client, mentre un **thin client** esegue l'elaborazione sul server.
Esempi di utilizzo.

- ✓ Creazione di un'applicazione client che recuperi informazioni sull'utente inviandole quindi a un server centrale.
- ✓ Creazione di un'applicazione server che funga da punto di raccolta centrale per i dati inviati da vari utenti.
- ✓ Creazione di un'applicazione di "conversazione".

PROTOCOLLO

Quando si utilizza il controllo Winsock, è innanzitutto necessario considerare se è preferibile utilizzare il protocollo TCP o il protocollo UDP.
La differenza principale tra i due protocolli riguarda lo stato di connessione.

- ✓ Il protocollo TCP è un protocollo basato sulla connessione ed è analogo a un telefono, in quanto l'utente deve stabilire una connessione prima di poter procedere.
- ✓ Il protocollo UDP non necessita invece di connessione e la transazione tra due PC è analoga al passaggio di un messaggio che è inviato da un PC all'altro, senza che sia stabilita alcuna connessione esplicita tra i due PC. La dimensione massima dei dati che è possibile inviare in ciascuna trasmissione dipende dalla rete.

La scelta del protocollo è in genere determinata dal tipo di applicazione che si desidera creare. Di seguito sono riportate alcune considerazioni che possono risultare utili per la selezione del protocollo appropriato.

1. Stabilire se l'applicazione richiederà un riconoscimento da parte del server o del client all'invio o alla ricezione dei dati. In caso affermativo, il protocollo TCP richiede una connessione esplicita prima di inviare o ricevere dati.
2. Stabilire se i dati saranno di grandi dimensioni, ad esempio nel caso di file immagine o audio. Dopo aver stabilito la connessione, il protocollo TCP la mantiene e garantisce l'integrità dei dati. Questa connessione, tuttavia, utilizza una maggior quantità di risorse.
3. Stabilire se i dati saranno inviati in un'unica sessione o in sessioni diverse. Se ad esempio si desidera creare un'applicazione che notifica a determinati PC il completamento di particolari operazioni, il protocollo UDP può essere la scelta migliore. Il protocollo UDP è inoltre più adatto all'invio di piccole quantità di dati.

Per impostare il protocollo che sarà utilizzato dall'applicazione, è necessario, in fase di progettazione, fare clic su *Protocol* nella finestra *Proprietà*, quindi selezionare

sckTCPProtocol o sckUDPProtocol. È inoltre possibile impostare la proprietà *Protocol* nel codice, come illustrato di seguito:
Winsock1.Protocol = sckTCPProtocol

PROPRIETÀ

Proprietà	Descrizione	Tipo di dato
BytesReceived	Corrisponde al numero di bytes scambiati nell'operazione di comunicazione Client/Server. Rappresenta dunque le dimensioni dei pacchetti di dati scambiati. Tale proprietà è associata al metodo GetData;	Long
LocalHostName	Corrisponde al nome del computer sul quale opera l'applicazione contenente il controllo Winsock;	String
LocalIP	Corrisponde all'indirizzo del computer sul quale opera l'applicazione contenente il controllo Winsock;	String
LocalPort	È il numero che identifica la porta attraverso la quale sarà effettuata la comunicazione Client/Server. E' una proprietà di lettura e scrittura anche in fase di esecuzione. In particolare: Per il client, questa proprietà definisce la porta locale dalla quale sono inviati i dati. Se l'applicazione non richiede una determinata porta, specificare la porta 0. Il controllo effettuerà la selezione casuale di una porta. Dopo aver stabilito la connessione, la porta locale sarà utilizzata per la connessione TCP. Per il server, questa proprietà definisce la porta locale di attesa. Se è specificata la porta 0, sarà utilizzata una porta casuale. Dopo aver richiamato il metodo Listen, la proprietà conterrà la porta selezionata;	Long
Protocol	E' il tipo di protocollo che sarà utilizzato. La scelta è tra TCP ed UDP. Prima di reimpostare questa proprietà è necessario chiudere il controllo utilizzando il metodo Close. Le impostazioni di Protocol possono essere: *sckTCPProtocol* caratterizzato dalla costante 0 che indica l'utilizzo (predefinito) del protocollo TCP; *sckUDPProtocol* caratterizzato dalla costante 1 che indica l'utilizzo del protocollo UDP;	val/cost
RemoteHost	È il nome del computer al quale ci si connette per la richiesta di scambio dati;	String
RemoteHostIP	È l'indirizzo IP del computer al quale ci si connette per la richiesta di scambio dati;	String
RemotePort	Rappresenta la porta alla quale connettersi per richiedere i dati. Quando è impostata la proprietà Protocol, la proprietà RemotePort è automaticamente impostata sulla porta predefinita appropriata di ciascun protocollo, ossia: porta 80 ->HTTP, utilizzata comunemente per le connessioni al World Wide Web; porta 21 ->FTP	Long
SocketHandle	È un valore corrispondente all'handle di socket utilizzato dal controllo per comunicare con il livello Winsock. E' una proprietà di sola lettura e non disponibile in fase di progettazione. Tale proprietà sarà utilizzata per essere passata alle API Winsock.	Long
State	Restituisce lo stato del controllo Winsock nel momento in cui n'è richiesta la verifica. I tipi di stato sono: *sckClosed* corrispondente al valore 0. E' l'impostazione predefinita che corrisponde allo stato 'chiuso' del controllo; *sckOpen* corrispondente al valore 1 ed indica lo stato 'aperto' del controllo; *sckListening* corrispondente al valore 2 ed indica lo stato 'in attesa di comunicazione' del controllo; *sckConnectionPending* corrispondente al valore 3 ed indica lo stato	Long

	'comunicazione in corso' del controllo; *sckResolvingHost* corrispondente al valore 4 ed indica lo stato 'risoluzione dell'host in corso' del controllo; *sckHostResolved* corrispondente al valore 5 ed indica lo stato 'host risolto' del controllo; *sckConnecting* corrispondente al valore 6 ed indica lo stato 'aperto' del controllo; *sckConnected* corrispondente al valore 7 ed indica lo stato 'connesso' del controllo; *sckClosing* corrispondente al valore 8 ed indica che il client sta chiudendo la comunicazione; *sckError* corrispondente al valore 9 ed indica un errore in un'operazione eseguita da Winsock;	

METODI

Metodo	Descrizione
BytesReceived	Consente di accettare una richiesta di comunicazione da parte del client. Generalmente il metodo Accept è utilizzato nell'evento ConnectionRequest ed in una nuova istanza del controllo Winsock anziché nel controllo in attesa;
Bind	Specifica la porta locale e l'IP locale da utilizzare per le connessioni TCP. La sintassi è la seguente: Winsock.Bind PortaLocale, IPLocale dove PortaLocale è la porta utilizzata per effettuare la connessione e IPLocale è l'indirizzo IP locale utilizzato per effettuare la connessione;
Close	Chiude la comunicazione TCP o una socket in attesa per entrambe le applicazioni client e server;
Connect	Invia una richiesta di connessione ad un computer remoto. La sintassi è la seguente: Winsock.Connect HostRemoto, PortaRemota dove HostRemoto è il nome del computer server al quale ci si connette e PortaRemota è la porta utilizzata;
GetData	Recupera il blocco di dati corrente e lo memorizza in una variabile di tipo Variant. La sintassi è la seguente: oggetto.GetData dati, [tipo,] [lunMax] dove dati memorizza i dati recuperati dopo l'esecuzione del metodo. Se i dati non sono sufficienti per il tipo di richiesta, la variabile sarà impostato su Empty. tipo è invece il tipo di dati da recuperare. Il metodo GetData viene in genere utilizzato con l'evento DataArrival contenente l'argomento totalBytes. Se il valore di lunMax è minore del valore dell'argomento totalBytes, sarà visualizzato il messaggio di avviso 10040 in cui è comunicato che i byte rimanenti andranno perduti. Le possibili impostazioni di tipo sono le seguenti: *vbByte* Byte *vbInteger*Integer *vbLong* Long *vbSingle* Single *vbDouble* Double *vbCurrency* Currency *vbDate* Date *vbBoolean* Boolean *vbError* SCODE *vbString* String *vbArray* + *vbByte* Byte Array lunMax specifica la dimensione desiderata della matrice Byte o della stringa da ricevere. Se non è specificata alcuna matrice Byte o stringa in questo argomento, saranno recuperati tutti i dati disponibili. Se è specificato un tipo di dati diverso da una matrice Byte o una stringa, l'argomento sarà ignorato.
Listen	Crea una socket e la imposta in modalità di attesa. Questo metodo può essere

	utilizzato solo per le connessioni TCP;
PeekData	A differenza di GetData, il metodo PeekData non rimuove i dati dalla coda di input. Questo metodo può essere utilizzato solo per le connessioni TCP. Tale metodo ha una sintassi praticamente simile a quella di GetData ossia: Winsock.PeekData dati, [tipo,] [lunMax]
SendData	Invia i dati al computer remoto. La sintassi è del tipo: Winsock.SendData dati dove dati rappresenta il gruppo di dati da inviare. Per i dati binari, è necessario utilizzare una matrice Byte. Quando è passata una stringa UNICODE, la stringa è convertita in stringa ANSI prima di essere inviata dalla rete;

EVENTI

Evento	**Descrizione**
Close	È generato quando il computer remoto interrompe la connessione;
Connect	È generato nel momento in cui la connessione è stata stabilita con successo;
ConnectionRequest	È generato nel momento in cui un computer client invia una richiesta di connessione. Il server può quindi decidere se accettare o no la comunicazione. In caso di rifiuto, il client riceverà l'evento Close ;
DataArrival	È generato nel momento in cui un computer client riceve dei dati. L'evento è generato nel caso in cui tutti i nuovi dati inviati col metodo GetData siano recuperati con successo. Per evitare duplicazione di pacchetti di dati solamente i nuovi dati saranno considerati. La sintassi è la seguente: Winsock.DataArrival (BytesTotali As Long) dove BytesTotali rappresenta il numero di dati che possono essere recuperati;
Error	È generato nel caso in cui si verifichi un errore qualsiasi nel procedimento di connessione, di invio, di richiesta o di chiusura della connessione. La sintassi è la seguente: Winsock.Error(Numero As Integer, Descrizione As String, Scode As Long, Origine As String, HelpFile as String, HelpContext As Long, CancelDisplay As Boolean) dove Numero indica il codice dell'errore, Descrizione la descrizione dell'errore, Scode è un valore di tipo Long SCODE, Origine descrive l'origine dell'errore, HelpFile è una stringa contenente il nome del file della Guida in linea, HelpContext è il contesto della Guida in linea e CancelDisplay indica l'annullamento della visualizzazione. L'impostazione predefinita (CancelDisplay = False) permette la visualizzazione di una finestra di messaggio di errore predefinita. Di seguito sono elencati i codici degli errori: *sckOutOfMemory* corrispondente al valore 7. Descrive un errore causato dall'esaurimento della memoria; *sckInvalidPropertyValue* corrisponde al valore 380 e descrive un valore della proprietà non valido *sckGetNotSupported* corrispondente al valore 394. Descrive un errore causato dall'impossibilità di leggere la proprietà; *sckSetNotSupported* corrispondente al valore 383. Descrive un errore causato dall'impostazione di una proprietà che è di sola lettura; *sckBadState* corrispondente al valore 40006. Protocollo o stato della connessione errato per la transazione o la richiesta; *sckInvalidArg* corrispondente al valore 40014. L'argomento passato a una funzione è in un formato errato o non è compreso nell'intervallo specificato. *sckSuccess* corrispondente al valore 40017. Connessione completata. *sckUnsupported* corrispondente al valore 40018. Indica un valore di tipo Variant non supportato *sckInvalidOp* corrispondente al valore 40020. Indica un'operazione non valida *sckOutOfRange* corrispondente al valore 40021. Indica un argomento non compreso nell'intervallo. *sckWrongProtocol* corrispondente al valore 40026. Indica che è stato usato un

	protocollo per cui non è consentita l'operazione che si è tentato di compiere. *sckOpCanceled* corrispondente al valore 1004. L'operazione è stata annullata. *sckInvalidArgument* corrispondente al valore 10014. Indica che il flag non è stato impostato. *sckWouldBlock* corrispondente al valore 10035. Indica che non è possibile bloccare il socket sebbene ne sia stato specificato il blocco. *sckInProgress* corrispondente al valore 10036. Indica che è in corso un'operazione Winsock di blocco. *sckAlreadyComplete* corrispondente al valore 10037. Indica che l'operazione richiesta è già stata eseguita. *sckNotSocked* corrispondente al valore 10038. Indica che non si fa riferimento ad un socket. *sckMsgTooBig* corrispondente al valore 10040. Indica un datagramma troppo grosso per essere inserito nel buffer. *sckPortNotSupported* corrispondente al valore 10043. Indica che la porta specificata non può essere utilizzata per l'operazione indicata. *sckAddressInUse* corrispondente al valore 10048. Indica un indirizzo già in uso. *sckAddressNotAvailable* corrispondente al valore 10049. Indica che l'indirizzo indicato del computer locale non è valido. *sckNetworkSubsystemFailed* corrispondente al valore 10050. Indica un errore nel sottosistema di rete. *sckNetworkUnreachable* corrispondente al valore 10051. Indica che l'host non è in grado di raggiungere la rete. *sckNetReset* corrispondente al valore 10052. Indica che la connessione è stata interrotta quando è stato impostato SO_KEEPALIVE. *sckConnectAborted* corrispondente al valore 10053. Indica una connessione fallita a causa di un timeout (tempo massimo di attesa superato) o di un altro errore. *sckConnectionReset* corrispondente al valore 10054. Indica che la connessione è stata reinviata dal computer remoto. *sckNoBufferSpace* corrispondente al valore 10055. Indica che lo spazio nel buffer è esaurito. *sckAlreadyConnected* corrispondente al valore 10056. Indica che la connessione è già stata stabilita. *sckSocketShutdown* corrispondente al valore 10058. Indica che la connessione è già stata chiusa. *sckTimedout* corrispondente al valore 10068. Indica che la connessione è stata chiusa. *sckConnectionRefused* corrispondente al valore 10061. Indica che la connessione è stata rifiutata dal computer server. *sckNotInitalized* corrispondente al valore 10093. Indica che è necessario richiamare WinsockInit per primo. *sckHostNotFound* corrispondente al valore 11001. Indica che è impossibile trovare l'host in modo definitivo. *sckInvalidArgument* corrispondente al valore 11002. Indica che è impossibile trovare l'host. *sckNotRecoverableError* corrispondente al valore 11003. Indica un errore non riconoscibile. *sckNoData* corrispondente al valore 11004. Indica che non è possibile ricevere alcun tipo di dati del tipo richiesto (quando si fa uso della specifica Winsock.GetData (tipo).
SendComplete	È generato quando tutti i dati sono stati inviati con successo;
SendProgress	È generato quando il processo di invio dei dati è ancora in corso;

INDIVIDUAZIONE DEL NOME DEL PC

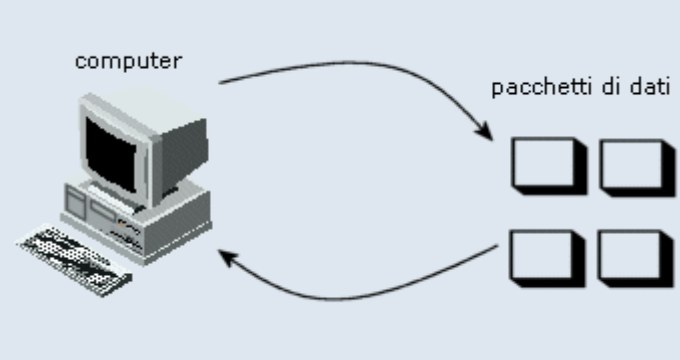
Per connettersi a un PC remoto, è necessario conoscerne l'indirizzo IP o il nome.
Per individuare il nome del PC tramite sistema operativo.

1. Fare clic sul pulsante *Start* e selezionare *Risorse del PC*.
2. Clic pulsante destro, dal menu contestuale scegliere la voce *Proprietà*.
3. Si apre la finestra *Proprietà del sistema*.
4. Selezionare la scheda *Nome PC*.

Il nome del PC potrà quindi essere utilizzato come valore per la proprietà *RemoteHost*.
Per individuare il nome del PC tramite linguaggio di programmazione.
Aprire un nuovo progetto EXE Standard, inserire un controllo Winsock sulla form, *Progetto/Componenti…(CTRL+T)/Microsoft Winsock Control 6.0* ed aggiungere nel modulo di codice di Form1 il seguente segmento di codice.

```
Private Sub Form_Load()
        MsgBox Winsock1.LocalHostName
        MsgBox Winsock1.LocalIP
End Sub
```

Avviando l'applicazione si riceverà una finestra di messaggio recante il nome del PC e l'identificativo IP.
Come si è visto dalla prova fatta, non è assolutamente necessaria la presenza di due PC fisici differenti. Client e server possono, infatti, essere il medesimo PC.



NOZIONI FONDAMENTALI SULLA CONNESSIONE TCP

Quando si crea un'applicazione che utilizza il protocollo TCP, è innanzitutto necessario decidere se tale applicazione sarà server o client. Se l'applicazione è un server rimarrà in "ascolto" su una determinata porta. Quando il client invia una richiesta di connessione, il server potrà accettare tale richiesta e completare la connessione. Client e server potranno quindi comunicare. La seguente procedura consente di creare un semplice server.

Per creare un server TCP

1. Creare un nuovo progetto EXE standard.
2. Assegnare al form predefinito il nome f*rmServer*.
3. Modificare la didascalia del form in "*Server TCP*".
4. Trascinare un controllo *Winsock* nel form e modificarne il nome in *tcpServer*.
5. Aggiungere nel form due controlli casella di testo. Denominare il primo *txtSendData* e il secondo *txtOutput*.
6. Specificare per il form il codice riportato di seguito.

```
Private Sub Form_Load()
        ' Imposta su un intero la proprietà LocalPort.
        ' Richiama quindi il metodo Listen.
        tcpServer.LocalPort = 1001
        tcpServer.Listen
        frmServer.Show
End Sub
Private Sub tcpServer_ConnectionRequest (ByVal requestID As Long)
        ' Verifica se il valore della proprietà State è sckClosed. In caso contrario, chiude la
        ' connessione prima di accettare quella nuova.
        If tcpServer.State <> sckClosed Then tcpServer.Close
        ' Accetta la richiesta con il parametro requestID.
        tcpServer.Accept requestID
End Sub
```

```
Private Sub txtSendData_Change()
        ' Il controllo casella di testo denominato txtSendData contiene i dati da inviare
        ' Quando l'utente digita una stringa nella casella di testo, tale stringa è inviata
        ' con il metodo SendData.
        tcpServer.SendData txtSendData.Text
End Sub
Private Sub tcpServer_DataArrival (ByVal bytesTotal As Long)
        ' Dichiara una variabile per i dati in ingresso.
        ' Richiama il metodo GetData e assegna i dati alla proprietà Text del controllo
        ‘ casella di testo txtOutput.
        Dim strData As String
        tcpServer.GetData strData
        txtOutput.Text = strData
End Sub
```

Il codice sopra riportato consente di creare una semplice applicazione server. Per completare l'esempio è tuttavia necessario creare anche un'applicazione client.

Per creare un client TCP

1. Aggiungere un nuovo form nel progetto e denominarlo *frmClient*.
2. Modificare la didascalia del form in "*Client TCP*".
3. Aggiungere un controllo *Winsock* nel form e denominarlo *tcpClient*.
4. Aggiungere in *frmClient* due controlli casella di testo. Denominare il primo *txtSendData* e il secondo *txtOutput*.
5. Disegnare un controllo pulsante di comando nel form e denominarlo *cmdConnect*.
6. Modificare la didascalia del controllo pulsante di comando in "*Connetti*".
7. Specificare per il form il codice riportato di seguito.

```
Private Sub Form_Load()
        ' Per specificare un host remoto, è possibile utilizzare l'indirizzo IP
        ‘ oppure il nome del PC.
        tcpClient.RemoteHost = "127.0.0.1"
        tcpClient.RemotePort = 1001
End Sub
Private Sub cmdConnect_Click()
        ' Richiama il metodo Connect per stabilire una connessione.
        tcpClient.Connect
End Sub
Private Sub txtSendData_Change()
          tcpClient.SendData txtSendData.Text
End Sub
Private Sub tcpClient_DataArrival (ByVal bytesTotal As Long)
        Dim strData As String
        tcpClient.GetData strData
        txtOutput.Text = strData
End Sub
```

Il metodo *GetData* svuota automaticamente la variabile *strData* con le informazioni dopo averle inviate a *txtOutput*. Per evitare ciò si può far ricorso alla proprietà *PeekData* che non rimuove i dati dalla coda d’input. Questo metodo può essere utilizzato solo per le connessioni TCP. Tale metodo ha una sintassi praticamente simile a quella di *GetData* ossia: *Winsock.PeekData dati, [tipo,] [lunMax].*

Il codice sopra riportato crea due semplici applicazioni: il client e il server.

Compilarli in due eseguibili separati.

Il client e il server possono essere eseguiti su un localhost o su due PC in rete.

Per verificarne il funzionamento, prima doppio clic su server.exe, quindi doppio clic su client.exe e fare clic sul pulsante *Connetti*. Digitare quindi del testo nel controllo casella di

testo *txtSendData* di un form: lo stesso testo sarà visualizzato nel controllo casella di testo *txtOutput* del secondo form.

Accettazione di più richieste di connessione

Il semplice server illustrato in precedenza è in grado di accettare un'unica richiesta di connessione. È tuttavia possibile accettare varie richieste di connessione utilizzando lo stesso controllo, creando una matrice di controlli. In questo caso non sarà necessario chiudere la connessione, ma sarà sufficiente creare una nuova istanza del controllo, impostandone la proprietà *Index*, e richiamare il metodo *Accept* per la nuova istanza. Nel codice riportato di seguito si presuppone che sia stato inserito in un form un controllo *Winsock* denominato *sckServer* e che la proprietà *Index* di tale controllo sia impostata su 0. In questo modo il controllo farà parte di una matrice di controlli. Nella sezione Dichiarazioni è dichiarata una variabile *intMax* a livello di modulo. Nell'evento *Load* del form, *intMax* è impostata su 0 e la proprietà *LocalPort* del primo controllo della matrice è impostata su 1001. È quindi richiamato il metodo *Listen* per il controllo, che è pertanto messo in "ascolto". All'arrivo di ciascuna richiesta di connessione, il codice verifica se la proprietà *Index* è uguale a 0 (il valore del controllo in "ascolto"). In caso affermativo, il controllo in "ascolto" incrementa il valore di *intMax* e utilizza tale numero per creare una nuova istanza del controllo. La nuova istanza sarà quindi utilizzata per accettare la richiesta di connessione.

```
Private intMax As Long
Private Sub Form_Load()
        intMax = 0
        sckServer(0).LocalPort = 1001
        sckServer(0).Listen
End Sub
Private Sub sckServer_ConnectionRequest (Index As Integer, ByVal requestID As Long)
        If Index = 0 Then
            intMax = intMax + 1
            Load sckServer(intMax)
            sckServer(intMax).LocalPort = 0
            sckServer(intMax).Accept requestID
            Load txtData(intMax)
        End If
End Sub
```

Nozioni fondamentali sul protocollo UDP

La creazione di un'applicazione UDP è più semplice della creazione di un'applicazione TCP, perché il protocollo UDP non richiede una connessione esplicita. Nell'applicazione TCP illustrata in precedenza, un controllo *Winsock* deve essere esplicitamente impostato "sull'ascolto", mentre l'altro controllo deve stabilire una connessione utilizzando il metodo Connect. Il protocollo UDP, al contrario, non richiede una connessione esplicita. Per inviare dati tra i due controlli, è necessario eseguire le seguenti operazioni su entrambe le parti interessate alla connessione.

1. Impostare la proprietà *RemoteHost* sul nome dell'altro PC.
2. Impostare la proprietà *RemotePort* sulla proprietà *LocalPort* del secondo controllo.
3. Richiamare il metodo *Bind* specificando la proprietà *LocalPort* da utilizzare.

Dal momento che entrambi i PC possono essere considerati "equivalenti" nella relazione che li lega, l'applicazione potrebbe essere definita di tipo peer-to-peer. Il codice riportato di seguito crea un'applicazione di "conversazione" che consente a due utenti di "conversare" in tempo reale.

Per creare un peer UDP

1. Creare un nuovo progetto EXE standard.

2. Assegnare al form predefinito il nome *frmPeerA*.
3. Modificare la didascalia del form in "Peer A".
4. Disegnare un controllo *Winsock* nel form e denominarlo *udpPeerA*.
5. Nella finestra *Proprietà* fare clic su *Protocol* e selezionare il protocollo UDP.
6. Aggiungere nel form due controlli casella di testo. Denominare il primo *txtSend* e il secondo *txtOutput*.
7. Specificare per il form il codice riportato di seguito.

```
Private Sub Form_Load()
        ' Il nome del controllo è udpPeerA
        With udpPeerA
                ' Assicurarsi di assegnare a RemoteHost il nome del PC
                .RemoteHost= "PeerB"
                .RemotePort = 1001          ' Porta a cui ci si ' connette.
                .Bind 1002                  ' Associazione alla porta locale.
        End With
        frmPeerB.Show                       ' Visualizza il secondo form.
End Sub
Private Sub txtSend_Change()
        ' Invia il testo quando viene digitato.
        udpPeerA.SendData txtSend.Text
End Sub
Private Sub udpPeerA_DataArrival (ByVal bytesTotal As Long)
        Dim strData As String
        udpPeerA.GetData strData
        txtOutput.Text = strData
End Sub
```

Per creare un secondo peer UDP

1. Aggiungere nel progetto un form standard.
2. Assegnare al form il nome *frmPeerB*.
3. Modificare la didascalia del form in "Peer B".
4. Disegnare un controllo *Winsock* nel form e denominarlo *udpPeerB*.
5. Nella pagina **Proprietà** fare clic su **Protocol** e selezionare il protocollo UDP.
6. Aggiungere nel form due controlli casella di testo. Denominare il primo *txtSend* e il secondo *txtOutput*.
7. Specificare per il form il codice riportato di seguito.

```
Private Sub Form_Load()
        ' Il nome del controllo è udpPeerB
        With udpPeerB
                ' Assicurarsi di assegnare a RemoteHost il nome del PC.
                .RemoteHost= "PeerA"
                .RemotePort = 1002          ' Porta a cui ci si connette.
                .Bind 1001                  ' Associazione alla porta
                ' locale.
        End With
End Sub
Private Sub txtSend_Change()
        ' Invia il testo quando viene digitato.
        udpPeerB.SendData txtSend.Text
End Sub
Private Sub udpPeerB_DataArrival (ByVal bytesTotal As Long)
        Dim strData As String
        udpPeerB.GetData strData
        txtOutput.Text = strData
```

End Sub

Per verificare il funzionamento dell'applicazione creata nell'esempio, eseguire il progetto premendo F5 e digitare del testo nel controllo casella di testo *txtSend* di uno dei due form. Il testo digitato sarà visualizzato nel controllo casella di testo *txtOutput* dell'altro form.

IL METODO BIND

Come illustrato nel codice sopra riportato, è necessario richiamare il metodo *Bind* quando si crea un'applicazione UDP. Il metodo Bind "riserva" una porta locale perché sia utilizzata dal controllo. Quando ad esempio si associa il controllo al numero di porta 1001, nessun'altra applicazione potrà utilizzare tale porta per "l'ascolto". Questo può risultare utile se si desidera impedire che una porta sia utilizzata da altre applicazioni. Il metodo *Bind* è inoltre caratterizzato da un secondo argomento facoltativo. Se nel PC sono presenti più schede di rete, l'argomento *IPLocale* consentirà di specificare quale scheda utilizzare. Se si omette questo argomento, il controllo utilizzerà la prima scheda di rete elencata nella finestra di dialogo visualizzata facendo doppio clic sull'icona *Rete* del *Pannello di controllo*. Quando si utilizza il protocollo UDP, è possibile modificare il valore delle proprietà *RemoteHost* e *RemotePort* mantenendo l'associazione alla stessa porta locale. Con il protocollo TCP è tuttavia necessario chiudere la connessione prima di modificare il valore delle proprietà *RemoteHost* e *RemotePort*.

SERVER WEB

INTRODUZIONE

I server web sono chiamati a svolgere una varietà di ruoli che vanno oltre la semplice fornitura ai browser di pagine e immagini. Inoltre, bisogna determinare quanti browser possono avere accesso al sito contemporaneamente e con quale velocità otterranno risposta. Anche se si decide di affidare la gestione dei server e dell'infrastruttura ad una società di hosting i problemi rimangono.
Il lavoro essenziale svolto da un server web è semplice: accetta la richiesta di un agente, il browser, determina se concedere o meno l'accesso all'elemento desiderato e poi risponde, a seconda dei casi, inviando un file che risiede nel file system, passando la richiesta stessa a una routine per la generazione di codice dinamico o restituendo un messaggio di errore. Tutte le applicazioni web hanno un componente software fondamentale: un server applicativo, che è la mente che muove ogni sito web.
Per pubblicare informazioni da distribuire occorrono dei linguaggi.

LINGUAGGI A DESCRIZIONE DI PAGINA

HTML (*Hyper Text Markup language*)
CSS (*Cascade Style Sheets*)
XML (*eXtensible Markup Language*)
È un metalinguaggio che permette di creare linguaggi personalizzati di markup, supera così il grande limite attuale del web che è quello della dipendenza da un tipo di documento HTML singolo e non estensibile.

LINGUAGGI DI PROGRAMMAZIONE

Gli script sono procedure immerse direttamente nel codice HTML che il server o il browser possono interpretare ed eseguire, consentendo la realizzazione di vere e proprie applicazioni. Attualmente, si hanno due tipi di server applicativi.

Primo Tipo
Scripting client-side
Sito web statico: richiede l'uso di linguaggi client side.
Il funzionamento di un sito statico è semplice: il browser richiede una pagina, il server web la legge nel file system e la invia in risposta al browser.
Vantaggio: non si occupano le risorse del server.
Svantaggi: browser sempre aggiornati, problemi di compatibilità.
VBScript non è utilizzato come IDE, in pratica non è possibile creare form con i controlli, al contrario VBScript interviene solo dopo l'inserimento di controlli ActiveX in una pagina web (interfaccia utente) e il modo con cui sono aggiunti non sono funzioni VBScript, che si occupa invece del modo in cui i controlli interagiscono con l'utente e tra loro.
JavaScript, applet Java.

Scripting server-side
Sito web dinamico: richiede l'uso di linguaggi server side.
Vantaggio: non dipendono dal browser (*user-agent*) che li richiama, di conseguenza, il codice server side è del tutto invisibile al client che accede soltanto all'output dello script, cioè la pagina generata dalla sua esecuzione.
Svantaggio: server potente.
CGI (*Common Gateway Interface*) - **ISAPI** (*Internet Server Application Programming Interface*) nel periodo iniziale del web le applicazioni Internet erano implementate con programmi scritti in C o Perl che si appoggiavano ad un'API dei server web, la CGI o

ISAPI che richiedevano l'avvio di una nuova istanza di un programma ogni volta che si doveva elaborare una richiesta: rischi per la sicurezza. Un browser può, così, inviare una richiesta ad un server il quale, invece di rispondere con una pagina statica, può eseguire uno script ed inviare un flusso di dati HTML creato al momento. Nel web server, al di sotto della radice del sito, esiste una directory di nome *cgi-bin*, i file presenti sono di tipo eseguibile, per esempio .exe. Lo svantaggio di queste tecnologie risiede nella difficoltà di realizzazione e di gestione delle applicazioni, perché i programmi CGI non sono integrati nei file HTML e richiedono conoscenze ed un processo di sviluppo completamente diversi da quelli necessari per realizzare pagine HTML. Inoltre, trattandosi di tecnologie che si basano interamente sul server, il carico su quest'ultimo aumenta, portando ad un progressivo degrado delle prestazioni.
PERL (*Practical Extracting and Reporting Language*) **l**inguaggio derivato dal C.
ASP (*Active Server Pages*) scripting server-side in IIS, possibilità di scrivere i programmi usando qualunque linguaggio di script ActiveX, inclusi VBScript, Jscript, PERL (1987, Larry Wall, programmatore della Burroughs, azienda, ora confluita in Unisys, progettò il *Practical Extraction and Reporting Language*), Python, PHP e altri.
ASP.NET
È una versione più avanzata di ASP che supporta i linguaggi C# e VB.net, compila tutte le pagine in **IL** (*Intermediate Language*), poi compila IL in codice x86 con un compilatore **JIT** (*Just In Time*).
Altri scripting server-side sono.
JSP (*Java Server Pages*) di Sun, permette d'includere codice Java direttamente nelle pagine web e supporta componenti chiamati JavaBeans che nascondono funzionalià complesse al di là di un'interfaccia semplice.
Servlet Java applet su server, sono applicazioni Java in esecuzione su una JVM residente su un server.
PHP
1. *Personal Home Page*.
2. *Hypertext PreProcessor*.

ColdFusion
Di Macromedia, prima soluzione commerciale.

<u>Secondo tipo</u>
Sono orientati ai componenti, component-based.
EJB (*Enterprise JavaBeans*).
COM+ (*Component Object Model)*.
CORBA (*Common Object Broker Request Architecture*).
.Net Framework.
CMS (*Content Management System*): applicazioni che offrono tutti gli strumenti necessari per organizzare, gestire e pubblicare contenuti web; per esempio Joomla.
Anche se è più facile sviluppare e installare siti dinamici con i server basati su tecniche di scripting, i sistemi di questo tipo uniscono il codice usato per la visualizzazione della pagina, la cosiddetta presentation logic, con quello che implementa invece la business logic, che normalmente preleva o inserisce in un database le informazioni opportune. Ciò dà origine a codice molto contorto e a volte impossibile da aggiornare e mantenere in efficienza.
I server applicativi di fascia più elevata usano invece un modello di sviluppo orientato ai componenti, che separa la business logic da quella di presentazione. Questa divisione in compartimenti consente ai programmatori di ristrutturare rapidamente anche i siti di più grandi dimensioni: si possono infatti aggiungere nuovi componenti ogni qualvolta se ne presenta la necessità e non è necessario ricostruire l'intero sistema.
I server applicativi sono affidabili (supporto per il clustering, il carico di lavoro è ripartito tra più macchine) e scalabili (gestiscono un aumento del carico).
In molti casi è opportuno che il server s'interfacci con il server di directory aziendale, di

solito basato su **LDAP** (*Lightweight Directory Access Protocol*)o su Active Directory di Microsoft. In futuro SQL potrebbe essere sostituito da un linguaggio basato su **XTML** (*eXtensible HyperText Markup Language*) e chiamato XML Query.

ARCHITETTURA

Modello Forked (a forcella)

Usa un processo dedicato di I/OP per ogni richiesta di connessione, usato da Apache dove ogni istanza del server è capace di supportare 256 connessioni per porta. Pone però un grosso carico sulle risorse del sistema ed ha un limite prefissato sul numero di connessioni per porta.

Modello Select

Noto anche modello event-driven (azionato dagli eventi) usa un sistema di I/O non bloccante nel quale un piccolo numero di processi a thread unico sfruttano sfruttano la chiamata di sistema select() per gestire un gran numero di connessioni.

Modello Multithread

Ogni processo sfrutta un pool di thread di esecuzione. Per conservare le risorse del sistema, un algoritmo specializzato distribuisce il tempo di elaborazione della CPU tra i vari thread. Il modello d'implementazione del multithreading nel sistema operativo usato influisce sulle prestazioni, usato da IIS.

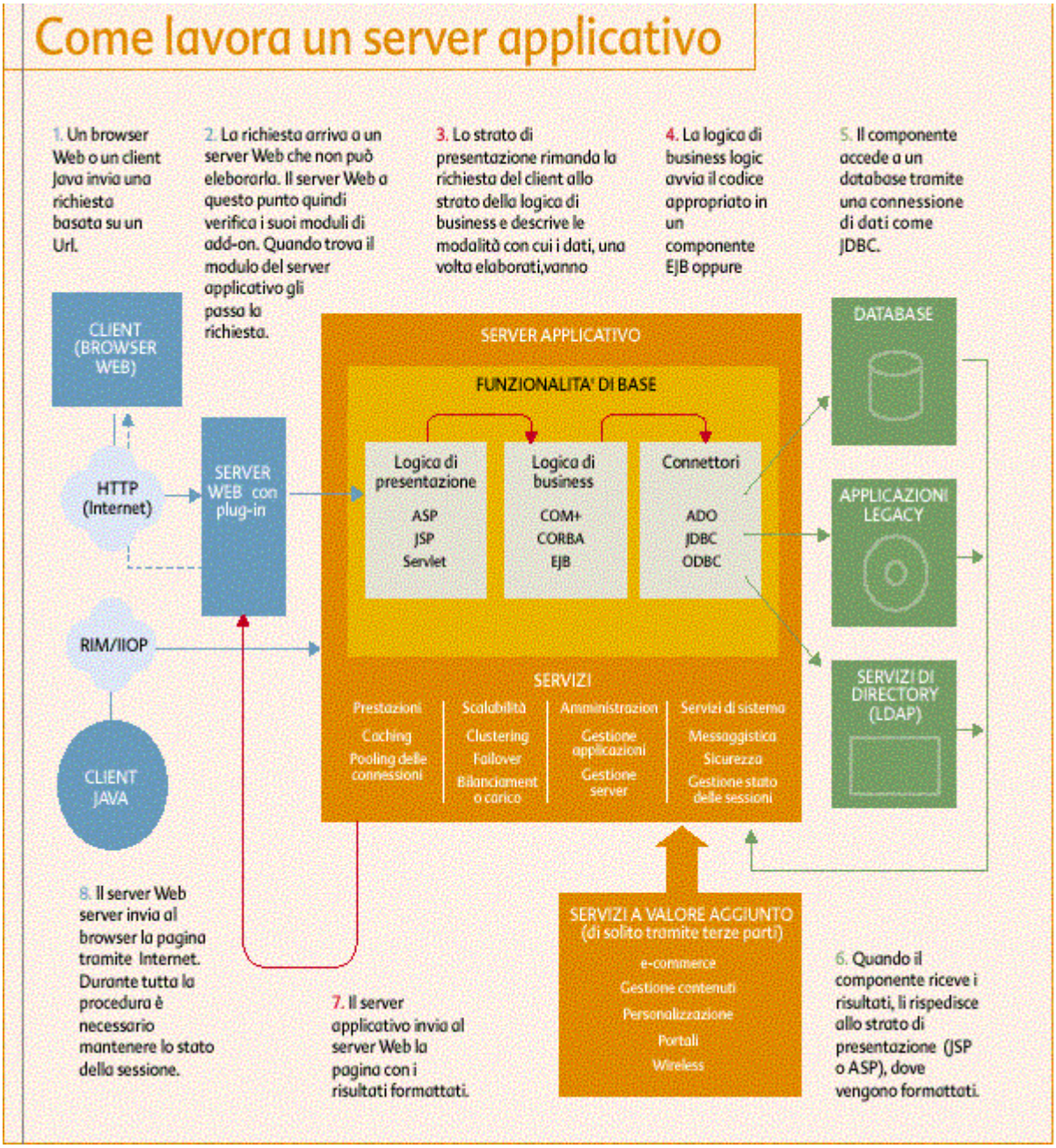
Il Web non conosce orari di chiusura. Come fare,allora, per tenere aperte le porte virtuali della propria azienda ventiquattro ore al giorno per sette giorni la settimana?

La risposta è nei cosiddetti sistemi ad alta disponibilità (high availability),che si avvicinano al 100%di uptime. I principi dell'alta disponibilità definiscono livelli precisi di backup e di recovery. Fino a poco tempo addietro, l'alta disponibilità implicava semplicemente la capacità di recupero hardware o software tramite la tecnologia **RAID** (*Redundant Array of Indipendent Disks*) che permette di ottenere una condizione di fault tolerance per i dati, ma non risolve il problema nel caso di un collasso completo del DBMS. Per raggiungere un uptime superiore, gli amministratori di database si stanno sempre più orientando verso il clustering. I cluster operano associando server che hanno la capacità di condividere un gruppo di unità disco. Ogni nodo ha nel suo cluster un nodo di backup. In caso di guasto del Nodo uno, il Nodo due è in grado di subentrare istantaneamente prendendosi carico delle risorse, della logica e delle funzioni transazionali del DBMS in avaria. Un vantaggio aggiuntivo del clustering è che i nodi non devono necessariamente risiedere nello stesso luogo, ma possono essere distanti anche numerosi chilometri tramite connessioni in fibra ottica.

Le tecnologie di clustering sono costose, una soluzione alternativa e molto efficace dal punto di vista economico è quella del log shipping (spedizione dei log), che prevede la sincronizzazione di database separati tramite l'invio dei log delle transazioni da un server all'altro. In caso di guasto, si possono usare i log per ripristinare la situazione fino al momento dell'avaria.

Altri sistemi prevedono le copie snapshot dei database ("istantanee" effettuate in momenti prestabiliti) o l'uso di tecnologie di replica.

Come lavora un server applicativo



DATABASE PER IL WEB

INTRODUZIONE

Non si può connettere il browser di un utente web remoto direttamente ad un sistema database, anche se ci sono eccezioni a questo (le Java applet sono una cosa di questo tipo), ma nella maggioranza dei casi il browser dialoga con un'applicazione che gira su un server web che ha funzioni d'intermediario rispetto al database. Questo programma può essere uno script CGI, una servlet Java o una parte di codice che sta all'interno di un documento ASP o JSP. L'utente non sa se la pagina è un documento HTML ordinario oppure l'output di un qualche script che genera l'HTML.
L'architettura web/database si sviluppa verso diverse direzioni.

La soluzione Open Source

LAMP (*Linux, Apache, MySQL, P** si può sostituire con PERL, Python o PHP).
JDBC (*Java DataBase Connectivity*) è un set di librerie che connettono i programmi Java, inclusi le servlet e i JSP, ai server SQL.

La soluzione Microsoft

ASP, ADO.

La Piattaforma .NET

Sposta tutta la programmazione verso un modello web centrico dove le applicazioni comunicano l'una con l'altra usando l'HTTP e **XML** (*eXtensible Markup Language*). È un ambiente di esecuzione (framework) che si appoggia su Windows. Le applicazioni .NET usano i servizi del framework e non hanno accesso diretto al sistema operativo. Approccio concettualmente analogo alla **JVM** (*Java Virtual Machine*). L'architetto di .NET è Anders Hejlsberg che ha realizzato in casa Borland Turbo Pascal e Delphi e in casa Microsoft Visual J++.
ADO.NET

DATABASE PER GLI UTENTI MOBILI

Un venditore equipaggiato con un **PDA** (*Personal Digital Assistant*) dotato del software opportuno può collegarsi al database aziendale per verificare all'istante la disponibilità di un prodotto, lo stato di un ordine o gli ordini effettuati in passato da un cliente. I database mobili spesso sono versioni ridotte all'osso delle controparti per server, e prevedono solo le operazioni SQL di base per via delle risorse limitate offerte dal dispositivo portatile. I database mobili replicano i dati tra loro e con un database centrale. L'attività di replica implica l'esame di un database sorgente (source) per rilevare le modifiche dovute a transazioni recenti e la successiva propagazione di tali modifiche ai database di destinazione (targets). La replica deve essere asincrona, dal momento che gli utenti non dispongono di una connessione costante con il database centrale. Per mantenere l'integrità dei vari database è fondamentale che siano replicate solo le transazioni completate: se anche quelle parziali fossero replicate ne seguirebbe rapidamente il caos. È anche importante che la replica sia seriale, per mantenere lo stesso ordine in ciascun database e prevenire le incongruenze. Un 'altra considerazione relativa ai database mobili riguarda il modo in cui sono risolti i conflitti nel caso di aggiornamenti multipli dello stesso record.

UNA QUERY SQL ALL'OPERA

L'esecuzione di una query è una delle attività più importanti per un database. Ogni richiesta da parte di un utente che implichi l'interazione con la base di dati dà il via ad una query.
1° Passo
Un utente compila un form per ricercare in una libreria on-line i libri scritti da un certo

autore.
2° Passo
Ricevuta la richiesta da Internet un'applicazione sul server estrae i campi del modulo per costruire una query del database. La query è inviata al database tramite un driver.
3° Passo
Una volta ricevuta la query dal server il DBMS può applicare svariate tecniche di ottimizzazione delle prestazioni per recuperare rapidamente i dati richiesti.
Un indice organizzato come struttura gerarchica ad albero.
Tecnica di parallelismo intraquery che sfrutta più di una CPU per eseguire una singola query.
Query rewrites, analizzano le query in arrivo e le riscrivono dinamicamente usando tecniche avanzate.
4° Passo
La query è elaborata.
5° Passo
Il DBMS restituisce i risultati della query al server che li riformatta e li inserisce in una pagina HTML che poi invia all'utente.

SICUREZZA

Disabilitare i servizi non necessari
Disabilitare i servizi non usati o meglio rimuoverli, per esempio FTP, disabilitare i linguaggi di scripting e gli script di esempio. Limitare i servizi di file sharing (condivisone di file) a singole directory e non a tutto il drive, quindi creare una lista contenente gli indirizzi IP abilitati ad accedere al file sharing.
Mantenersi aggiornati sui problemi di sicurezza
Usare politiche sensate per le password
Modificare sempre le password di default ed eliminare gli account non necessari, per esempio guest. Disabilitare automaticamente un account dopo un numero predefinito di tentativi di accesso falliti. Usare password lunghe almeno sette caratteri di tipo diverso (lettere maiuscole, minuscole, numeri, segni d'interpunzione), cambiarla ogni sei settimane.
L'autenticazione è quella procedura che permette di stabilire se un utente è effettivamente chi dice di essere. Un sistema può autenticare un utente usando tre fattori.
1. Ciò che si conosce: per esempio, password.
2. Ciò che si possiede: per esempio, smart card.
3. Ciò che si è: per esempio, password biometriche, impronte digitali, iride, retina, voce.

Tenere la rete sotto controllo
Sniffing: lettura dei dati in transito perché le applicazioni basate sul TCP/IP, FTP, HTTP, SMNP, telnet offrono protezione scarsa per le password. Implementare un protocollo secondario dedicato alla sicurezza come SSH, per esempio HTTPS. Quando si amministrano i server e i router da remoto mai usare protocolli telnet, SMNP e FTP se non dopo aver attivato un protocollo di sicurezza.
Sfruttare il sistema di autorizzazioni del sistema operativo
Non far girare il server web come administrator (root) sulla macchina che lo ospita.
Monitorare i log
Il server web tiene traccia di tutte le richieste che gli pervengono.
Separare i dati pubblici da quelli privati
Non memorizzare dati riservati su sistemi usati anche come server web, a meno che non sia assolutamente necessario. Per un'extranet, adottare una configurazione con un "agnello sacrificale": ovvero un server web posto esternamente al firewall in modo da non mettere a rischio i dati aziendali dietro al firewall.
Prestare attenzione alla configurazione del server
Controllare le falle di sicurezza dei programmi
Gli attacchi DoS approfittano del fatto che i router, sprovvisti di adeguate misure di

filtraggio, trasmettono il traffico dati senza controllare adeguatamente l'indirizzo IP di provenienza, quello di destinazioni o la tipologia dei dati trasmessi. I socket sono un'API standard che gli sviluppatori usano per stabilire connessioni Internet. Un socket standard collega un'applicazione allo strato IP del sistema operativo tramite il livello di protocollo TCP o UDP, che predispone i dati in un formato standard e crea gli opportuni pacchetti di controllo. I raw socket a tutti gli effetti mettono a disposizione una backdoor per l'accesso diretto al sistema di rete sottostante, consentendo ai programmatori di creare pacchetti e protocolli a loro piacimento. Un'applicazione dannosa potrebbe usare i raw socket per impersonare l'indirizzo di un'altra macchina, in modo che non sia possibile risalire all'indirizzo effettivo del PC che sta generando i dati. raw socket sono stati implementati in origine nel sistema operativo Unix, dove risultano accessibili solo alle applicazioni che lavorano con i privilegi di root. Per esempio, la maggior parte degli utenti di Windows XP accederanno con i privilegi di administartor.

RENDERE SICURE LE APPLICAZIONI WEB

Quando un malintenzionato prende di mira un 'applicazione web, generalmente si pone quattro domande.

1) Posso vedere ciò che gli utenti stanno guardando?

Il metodo per prevenire che altri possano leggere i dati in transito è la cifratura della connessione fra un sito e i suoi utenti.

2) Posso impersonare un utente?

Come fa un sito web a riconoscere i propri utenti?

La maggior parte dei server web supporta due schemi di autenticazione tramite password: uno più semplice da implementare ma anche meno sicuro e uno, più robusto, basato sul cosiddetto digest. Entrambi gli schemi operano mandando una "sfida" al browser: quando il browser la riceve per la prima volta, fa comparire una finestra di dialogo che richiede il nome utente e la password.

Usando il metodo di autenticazione più semplice il browser trasmette al server il nome utente la password sotto forma di testo semplice e in chiaro (cioè non cifrato).

Con il secondo metodo di autenticazione il browser trasmette invece un particolare testo cifrato (detto appunto digest) ottenuto partire dal nome utente e dalla password.

In entrambi i casi il server verifica dati ricevuti e, se sono corretti, risponde con un messaggio di approvazione; il browser a sua volta memorizza in cache le informazioni di logon per tutta la sessione di lavoro in modo che l'utente non debba reinserirle ad ogni pagina visitata. Il primo problema riguarda ancora lo spionaggio dei dati in transito: se gli utenti mandano il nome utente e password in chiaro, un attaccante può catturarli. Questo problema può essere risolto in modo semplice trasmettendo informazioni dell'utente usando l'SSL. Il codice seguente mostra come farlo.

```
<form action="https://foo.com/login.asp "
method="post ">User ID:<input type="text"
name="user "><br>Password:<input
type="password"name="password "><br>
<input type="submit "value="Login ">
</form>
```

Non potendo spiare le comunicazioni, è possibile impersonare un utente. Il punto debole che rende questo possibile è proprio lo stesso utente. La maggior parte dei navigatori del web scelgono password non sicure e, ancora peggio, tendono a usare la stessa coppia nome utente/password per tutti siti che richiedono una procedura di logon.

3) Posso impersonare il gestore del sito?

Non c'è connessione permanente fra il browser e il server: è stabilita una connessione separata per ogni pagina richiesta. Come fa quindi un server web a verificare le credenziali dell'utente dopo il logon iniziale?

Il browser memorizza il nome utente e la password per tutto il tempo in cui la propria finestra rimane aperta, e li ritrasmette al server ogni volta che si ricollega ad esso. Il server

quindi confronta ogni volta le informazioni ricevute con quelle presenti nel suo database degli utenti e consente o nega l'accesso a seconda che esista o meno una corrispondenza. Il DNS è un anello vulnerabile della catena della sicurezza: se un malintenzionato riuscisse ad accedere al server DNS e ne modificasse i registri facendo in modo da farli puntare a uno dei propri PC invece che a quelli legittimi, tale PC potrebbe poi reindirizzare tutte le richieste HTTPS. Dato che durante il reindirizzamento il browser mostra di default la parte finale dell'indirizzo, se quest'ultimo è abbastanza lungo da mettere la porzione *www.sitofinto.com* fuori dal campo di vista, la maggior parte degli utenti non noteranno di essere collegati a un sito dal nome diverso da quello desiderato. Se il gestore di *www.sitofinto.com* ottenesse poi un certificato digitale, il browser indicherebbe una connessione del tutto lecita e non darebbe alcun messaggio di avviso. Solo facendo clic sull'icona a forma di lucchetto per visionare i dettagli del certificato digitale l'utente potrebbe rendersi conto dell'inganno. Dato che è molto facile costruire una pagina web identica a un'altra, il malintenzionato a questo punto potrebbe realizzare una schermata di accesso simile in tutto per tutto a quella di *www.lamiabanca.com* e usarla per catturare nomi utente e password, inserite dai visitatori ignari di essere collegati a un sito di un pirata informatico.

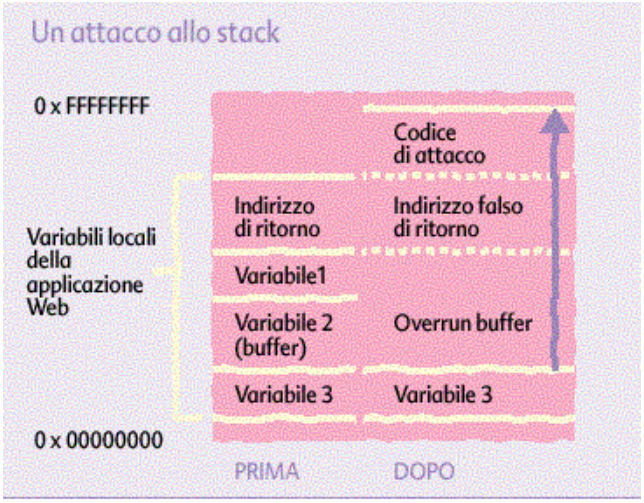
4) Posso usare i PC del sito per eseguire i miei comandi?

Un tipo di attacco molto noto è quello del **buffer overrun** (letteralmente:tracimazione del buffer), che prevede l'invio di una quantità eccessiva di dati al server. Il seguente frammento di codice C++ è vulnerabile a un attacco di buffer overrun perché non controlla se i dati in arrivo superino il limite previsto di 255 caratteri:

```
void ByYourCommand(char*pszData)
{   char szBuffer [255 ];
    strcpy(szBuffer,pszData ); ...}
```

Cosa accade se la funzione *strcpy()* provoca un overflow del buffer basato su stack?

La figura mostra l'aspetto dello stack delle chiamate di sistema dopo che è stata eseguita la funzione *ByYourCommand():* lo stack cresce a partire dalla parte alta della memoria e il buffer si riempie dal fondo della memoria. Scrivendo troppi dati all'interno del buffer, un attaccante può arrivare a sovrascrivere il cosiddetto *function call record*: si tratta di una struttura di dati che contiene i valori del registro salvati dal codice della funzione insieme con l'indirizzo di ritorno della funzione stessa. Sovrascrivendo quest'ultimo con un altro valore diventa possibile mandare in esecuzione sul PC attaccato un'applicazione a piacere.



Un pirata informatico naturalmente vorrà mandare in esecuzione il codice da lui stesso caricato sul sistema, ma come può riuscire ad introdurlo sul PC attaccato?

Scrivendolo nel buffer dei dati! Gli hacker sanno che le funzioni vulnerabili come quella appena mostrata sono di solito richiamate dal codice che elabora l'input dell'utente. Di conseguenza provano a mandare stringhe di input lunghissime al server e stanno a

vedere che cosa succede.
Come si può evitare che un ’applicazione web sia sfruttata illecitamente?
Bisogna cercare tutto il codice esistente che è stato scritto usando linguaggi che permettono un accesso diretto alla memoria, come il C, il C++e Delphi, ed esaminarlo per trovare eventuali vulnerabilità errori di buffer overflow. Si può evitare completamente il problema usando linguaggi di programmazione che non consentono accesso diretto alla memoria, per esempio meccanismi di scripting come JavaScript e PERL oppure un linguaggio interpretato come Java. Sono altrettanto sicuri gli imminenti linguaggi della piattaforma .NET, il C# e il Visual BASIC.NET.

ACCESSO AI DATI

INTRODUZIONE

Uno degli elementi costanti per tutte le applicazioni aziendali è che debbano elaborare informazioni tramite la creazione, il consumo e la gestione d'informazioni in forma di dati memorizzati in un PC. Ne consegue che le tecnologie per l'accesso, la gestione e la manutenzione dei dati e della loro integrità sono un elemento centrale per lo sviluppo di tutte le applicazioni aziendali. Dato che si basano su un modello aziendale, le applicazioni di questo tipo hanno requisiti vari a livello di funzionalità e di prestazioni per l'accesso ai dati.

Un'applicazione database è composta da tre parti.

1. **Applicazione**: è l'interfaccia utente.
2. **Sistema database**: è il modulo di gestione di database ed è indipendente dal database specifico cui si accede.
3. **Archivio dati**: sono i file in cui sono archiviate le tabelle del database.

Le tre parti di un'applicazione database possono essere suddivise in modi diversi.

1. **Database locali**, tutte le tre parti sono presenti in un PC.
2. **Database remoti**, sul server è presente l'archivio dati, sul client si ha l'applicazione e il sistema database.
3. **Database client/server**, sul server è presente il sistema database e l'archivio dati.

REQUISITI PER L'ACCESSO AI DATI

I requisiti di un'applicazione a livello di accesso ai dati sono univoci e dipendono dalla natura e dalla modalità d'implementazione dell'applicazione. Per esempio, le esigenze di accesso ai dati di un'applicazione per il supporto decisionale di sola lettura sono quasi all'opposto di quelle di un'applicazione per l'elaborazione di transazioni basata su operazioni di aggiornamento. In modo analogo, per lo sviluppo di un'applicazione che condivide i dati con PC portatili connessi occasionalmente, uno sviluppatore non può utilizzare la stessa logica di gestione dei dati adottata per un'applicazione implementata su un server centralizzato in larga scala che garantisce la connettività.

Le tecnologie di accesso ai dati sono suddivisi in tre categorie.

1. Complessità funzionale
2. Semplicità d'uso
3. Opzioni d'implementazione

COMPLESSITÀ FUNZIONALE

Velocità o funzionalità

Le prestazioni a livello di accesso a una fonte dati sono sempre un'unità di misura importante e sono uno degli ambiti in cui è necessario raggiungere i classici compromessi. Un'applicazione orientata alle transazioni richiede tempi di risposta veloci e funzioni di throughput di volumi elevati. Gli sviluppatori che si dedicano a queste applicazioni devono considerare la velocità come caratteristica prioritaria. Per le applicazioni per il supporto decisionale, in cui l'elemento centrale è la raccolta e l'analisi dei dati, gli sviluppatori daranno la precedenza alle funzionalità piuttosto che alla velocità. Per esempio, uno sviluppatore può scegliere una tecnologia di accesso ai dati che include funzioni di query e aggiornamento distribuite, un data store locale integrato e il supporto di semplici funzioni di I/O di dati locali per l'importazione e l'esportazione dei dati.

Interfacce proprietarie o non proprietarie

In alcuni casi, gli sviluppatori scelgono l'interfaccia di accesso ai dati che supporta tutte le funzioni a valore aggiunto di ogni singolo data store, scelta che può significare l'utilizzo di un'interfaccia proprietaria con un campo di utilizzo specifico. Non si tratta della scelta

ottimale, perché in questo caso non è possibile trasferire in un'altra applicazione che accede a una fonte dati diversa né il codice sviluppato né le conoscenze acquisite. Bisogna scegliere un'interfaccia aperta che consente di accedere a una vasta gamma di fonti dati e data store e che utilizzi un unico gruppo di interfacce di programmazione.

Supporto delle transazioni - integrità dei dati

Il supporto integrato delle transazioni è un elemento che sta acquisendo sempre più importanza. Tradizionalmente, le applicazioni basate su transazioni includono la logica per conservare l'integrità dei dati correlati. Ciò consente, in caso di problemi durante un aggiornamento, di conservare lo stato originale di tutti i dati correlati (**rollback**) che consente il ripristino dello stato d'integrità precedente all'errore. Inoltre, questo concetto di'integrità dei dati a livello di elaborazione delle transazioni è anche applicato all'integrità delle parti di codice correlate. Durante lo sviluppo di applicazioni basate su componenti, è importante per gli sviluppatori essere sicuri che lo stato dei componenti correlati abbia un livello di qualità dell'integrità pari a quello dei dati.

Componenti di database

Gli sviluppatori di applicazioni utilizzano componenti di database per ottimizzare le funzionalità e i tempi e le modalità di risposta delle applicazioni. Tali componenti possono essere i motori di cursori locali per la gestione a livello di client e senza codice dei gruppi di risultati provenienti dal server, oppure processori di query locali per il recupero e l'aggiornamento di dati da tipi di data store eterogenei.

SEMPLICITÀ D'USO

Uno dei fattori di valutazione principali per la pianificazione di un progetto di un'applicazione è la produttività degli sviluppatori. La scelta ottimale si riferisce a una tecnologia di accesso ai dati che fornisce tutte le funzionalità necessarie al livello più alto, ovvero il più semplice, della programmazione. Non è più necessario programmare a basso livello, ovvero a un livello più completo per creare il codice di un'applicazione altamente funzionale ed efficiente. Queste tecnologie si basano su un modello unico, di alto livello e indipendente dal linguaggio che elimina la necessità di cambiare le interfacce quando si combinano strumenti o linguaggi. La semplicità d'uso è anche influenzata dalla capacità dello sviluppatore d'interagire con tutti i tipi di fonti dati utilizzando un meccanismo unico per l'accesso ai dati. Attualmente, gli sviluppatori possono accedere a fonti dati multiple senza dovere conoscere alla perfezione una vasta gamma di interfacce proprietarie. La possibilità di diventare esperti di un'unica tecnologia per l'accesso ai dati consente di migliorare la produttività e di ridurre le esigenze di formazione.

OPZIONI D'IMPLEMENTAZIONE

La gran varietà di possibili scenari d'implementazione delle applicazioni, tra i quali gli ambienti client/server, multilivello o basati su Internet, rendono sempre più importante la flessibilità della tecnologia di accesso ai dati.

Client/Server

Ogni volta che due PC collaborano per eseguire un'applicazione, ognuno svolgendo compiti differenti, ci si trova sicuramente di fronte ad un'applicazione client/server: c'è un'interfaccia client per accedere ad informazioni remote presenti sul server, il carico di lavoro è distribuito tra il sistema client e il sistema server remoto, in base alle capacità delle applicazioni software del client e del server. Sono sistemi più efficienti perché minimizzano il traffico di rete e ogni porzione dell'applicazioni risulta ottimizzata per svolgere una particolare funzione: lavorano su qualsiasi tipo di rete e a qualsiasi velocità di comunicazione (queste variabili influenzano le prestazioni, non ne determinano le modalità di funzionamento). Il client e il server devono utilizzare lo stesso protocollo di comunicazione e lo stesso formato di dati.

L'essenza di un'applicazione client/server è rappresentata da un'applicazione di accesso a database. Molti **RDBMS** (*Relational DataBase Management System*) mettono a disposizione un motore di database (*database engine*) che gestisce i dati, ma che risulta

privo d'interfaccia utente. L'interfaccia accetta righe di comando, il programmatore deve progettare l'applicazione che gestisce l'interfaccia utente. In questo caso l'applicazione d'interfaccia utente, ovvero il client (le **API**, i driver di database risiedono nel client), formula le richieste dei dati che sono trasmessi al sistema sul quale è gestito il database, ovvero il server. Le richieste di dati sono formulate con comandi SQL, il motore del database eseguirà i comandi, creerà l'insieme dei risultati e lo trasmetterà al client.
Un client che non fornisce alcun supporto elaborativo e si limita a visualizzare i dati che gli sono trasmessi dal server è un **thin client** (magro, sottile).
Un client che prevede una logica elaborata per la manipolazione dei dati che gli sono trasmessi dal server (che è solo un deposito di dati) è un **thick client** (grasso, spesso).

Multilivello

Nelle **architetture a due livelli (two-tier)** il client comunica direttamente con il server, senza intermediari. Scarsa scalabilità, infatti, se il server ricevesse un carico di lavoro troppo elevato da parte dei client, l'unica soluzione consisterebbe nell'aggiornare il server.
Le **architetture a tre o più livelli (multi-tier)** sono adottate in Internet, per esempio l'accesso a un database su un server all'interno di una pagina web.
Questa configurazione supporta l'esecuzione nel livello intermedio della maggior parte dei processi di elaborazione dell'applicazione. Questo tipo di architettura è molto attraente per i programmatori, perché lo spostamento della logica aziendale e di accesso ai dati nel livello intermedio significa una quantità minore di codice per i client. Semplifica, inoltre, la gestione dei componenti di processo aziendali e della logica di accesso ai dati. Ne consegue che l'architettura multilivello rappresenta un metodo più scalabile e flessibile per la creazione di applicazioni: si basano su un'architettura a componenti come quella del modello DCOM. La flessibilità e la natura distribuita delle applicazioni multilivello rendono questa architettura ideale sia per progetti Internet che per progetti di reti aziendali.

Internet

Il browser web è un thick client perché elabora codice, JavaScipt, VBScript, ActiveX, che gli è trasmesso ogni volta che accede all'applicazione. Il web server è il server.
La differenza sostanziale è che, a differenza di una connessione su una rete aziendale, non si hanno garanzie che la connessione a un server di dati su Internet rimanga attiva. Le tecnologie di accesso ai dati utilizzate in Internet spesso partono dal presupposto che una connessione sia continua fino al completamento di una transazione. Le implicazioni sono minime per applicazioni che eseguono solo richieste, ma possono diventare rilevanti per applicazioni che richiedono funzionalità robuste per l'aggiunta, la modifica e l'eliminazione dei dati. Il risultato di una query deve essere conservato anche se la connessione è interrotta. In fase di aggiornamento, è necessario che sia eseguita un'operazione automatica di riconnessione e sincronizzazione del database. Questa situazione è a volte definita accesso ai dati senza connessione (modalità di lavoro offline). Le nuove tecnologie di accesso ai dati includono sia funzionalità con connessione sia senza connessione. La creazione di script server con tecnologie di accesso ai dati incorporate consente di accedere in modalità browser a dati aziendali su Internet e di fornire le informazioni ai client come dati in formato HTML. In questo modo un utente può visualizzare i dati aziendali nello stesso modo in cui visualizza altre informazioni su Internet.

Tecnologie Microsoft di accesso ai dati

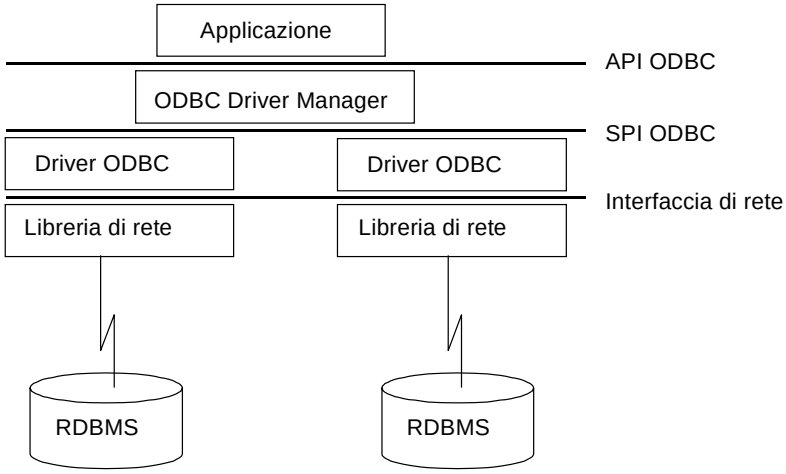
Come tutte le tecnologie di programmazione, anche le tecnologie di accesso ai dati si sono evolute. Le iniziali soluzioni point-to-point prevedono l'inclusione, nell'applicazione che è eseguita sul client, delle librerie proprietarie necessarie per comunicare con il server ove sono contenuti i dati. Sono esempi di queste soluzioni DB Library nel caso di Microsoft SQL Server, Open Client nel caso di Sybase e OCI nel caso di Oracle. Tutte queste soluzioni richiedono interventi di programmazione specifici per il server e in genere di basso livello. Uno dei punti di forza che ha caratterizzato Visual BASIC nel corso di questi ultimi anni, riguarda le grandi potenzialità che esso offre relativamente all'accesso ai dati.

Prima con l'avvento del Data Control e con l'integrazione del JetEngine, poi con ODBCDirect ed infine con i Designer. Infine ha ulteriormente sviluppato il linguaggio accoppiando l'accesso ai dati con le tecniche Object Oriented. Visual BASIC permette.

1. Un accesso ai dati semplificato, potente e veloce.
2. La possibilità di utilizzare contemporaneamente fonti dati eterogenee.
3. Di semplificare la creazione e la manutenzione di database e fonti dati.
4. Di utilizzare le nuove tecnologie di gestione dei dati tramite ADO e OLE DB.
5. Il riutilizzo di connessioni tra applicazioni diverse.

In passato, per fornire accesso ai dati era necessario progettare applicazioni apposite, i dati erano contenuti in file ASCII.
In seguito, l'accesso a DB Library, un'interfaccia basata sul linguaggio C, era realizzato tramite un prodotto denominato **VBSQL** (*Visual BASIC Structured Query Language*), questa interfaccia consentiva ai programmatori di accedere in modo efficiente, in modalità point-to-point, a SQL Server.
Si è poi passati verso database **ISAM** (*Indexed Sequential Access Method*), come Access, sono stati creati il motore di database Jet.
Questo tipo di approccio è stato esteso dall'introduzione di Visual Studio includendo un set di strumenti per l'accesso a database (qualsiasi database via ODBC oppure OLE DB con caratteristiche avanzate per database SQL Server e Oracle) chiamati Visual Database Tools. Consentono la creazione e manutenzione di database in modo rapido ed efficace. Possibilità di creare, manutenere e visualizzare dati e struttura di database SQLServer, Oracle e AS/400.

ODBC



America National Standard Institute Structured Query Language (ANSI92 SQL) è il linguaggio standard per l'accesso ai database relazionali (RDBMS) quali: SQL Server, Access, Sybase, Oracle e Informix. L'utilizzo di SQL consentiva agli sviluppatori d'incorporare istruzioni SQL nel codice delle applicazioni utilizzando metodi tradizionali per l'accesso ai dati SQL. Gli sviluppatori dovevano quindi eseguire il codice tramite un precompilatore specifico RDBMS per inserire nell'applicazione il codice necessario specifico del database. Sebbene la sintassi utilizzabile da un'applicazione fosse comune, indipendentemente dal database cui era necessario accedere, in fase di compilazione ogni applicazione era strettamente legata a un database specifico. L'utilizzo della stessa applicazione con un database diverso richiedeva la ricompilazione dell'applicazione con il precompilatore specifico del nuovo database. Per rispondere all'esigenza di un'interfaccia di programmazione standard per i dati relazionali Microsoft ha creato ODBC: un'API standard aperta, del tutto allineata agli standard ISO e XOpen, che consente alle applicazioni di accedere a diverse fonti dati SQL in fase di run-time senza richiedere la

ricompilazione delle applicazioni per ogni database di destinazione. In questo modo le applicazioni risultano isolate dalle implementazioni di database e di rete sottostanti. ODBC si basa sul concetto dei driver di database che eseguono la conversione tra l'API ODBC e la versione di SQL utilizzata da un database relazionale. In fase di run-time, ODBC Driver manager carica e gestisce le comunicazioni con i driver. I driver comunicano tramite un'interfaccia standard definita **SPI** (*Service Provider Interface*). ODBC è inoltre una tecnologia indipendente dalla rete, perché utilizza librerie di rete sostituibili per le comunicazioni. L'utilizzo dei metodi ODBC consente agli sviluppatori d'includere la logica per determinare, in fase di run-time, quali funzionalità specifiche siano supportate dal sistema RDBMS. In questo modo, gli sviluppatori hanno la possibilità di scrivere codice con logica condizionale basata sulle funzionalità e non su ogni sistema specifico di gestione di database. È così possibile scrivere applicazioni flessibili che possono sfruttare funzioni server diverse in modo coerente.

Ovviamente le prestazioni non sono granchè dato che è necessario attraversare un livello in più prima di arrivare al motore di database.

Il modello ADO 2.0 è semplice, gli oggetti sono soltanto sei (più tre Collection).

- In cima a tutto abbiamo l'oggetto **Connection** che rappresenta la connessione tra il client che utilizza gli ADO ed il server di database. Per quanto sopra l'oggetto Connection gestisce anche le transazioni; nel senso che Begintrans, Committrans e Rollbacktrans avvengono all'interno di un oggetto Connection.
- Logicamente dipendente dall'oggetto Connection abbiamo l'oggetto **Recordset**, ovvero l'insieme dei Record estratti da una tabella o per mezzo di una Query, un comando SQL. dal database.
- L'oggetto Recordset contiene una Collection **Fields** che è l'insieme degli oggetti **Field**, ovvero i campi che compongono ciascun Record del Recordset.
- Sempre dipendente diretto della Connection troviamo l'oggetto **Command** che è, in pratica, ogni comando specifico che vogliamo eseguire nei riguardi del nostro DataBase; una Query memorizzata, magari con parametri.
- L'oggetto Command contiene una Collection **Parameters**, composta da oggetti **Parameter**, ovvero tutti i parametri o argomenti associati al comando che vogliamo eseguire (Query parametrica o, nel caso per esempio di SQL Server od Oracle, anche Stored Procedure).
- Infine la Connection contiene una Collection **Errors** composta da oggetti **Error**, ovvero tutti gli errori generati per effetto di un mancato funzionamento dell'OLEDB Provider (impossibilità a stabilire una connessione, comando SQL sintatticamente errato, parametri mancanti o di tipo diverso da quello previsto dalla query).

È importante capire subito che, dato questo modello ad oggetti fondamentale, non è detto che ciascun oggetto abbia sempre le stesse caratteristiche; questo perché gli ADO possono essere usati in molti modi diversi e per accedere a fonti dati eterogenee. Tanto per fare un esempio l'oggetto Connection potrebbe non essere in grado di gestire le transazioni perché non supportate dall'OLEDB Provider in uso per accedere a quella certa fonte dati, oppure perché si sta usando una Connection lato client ed un Recordset sconnesso.

Interfacce ad oggetti per i dati

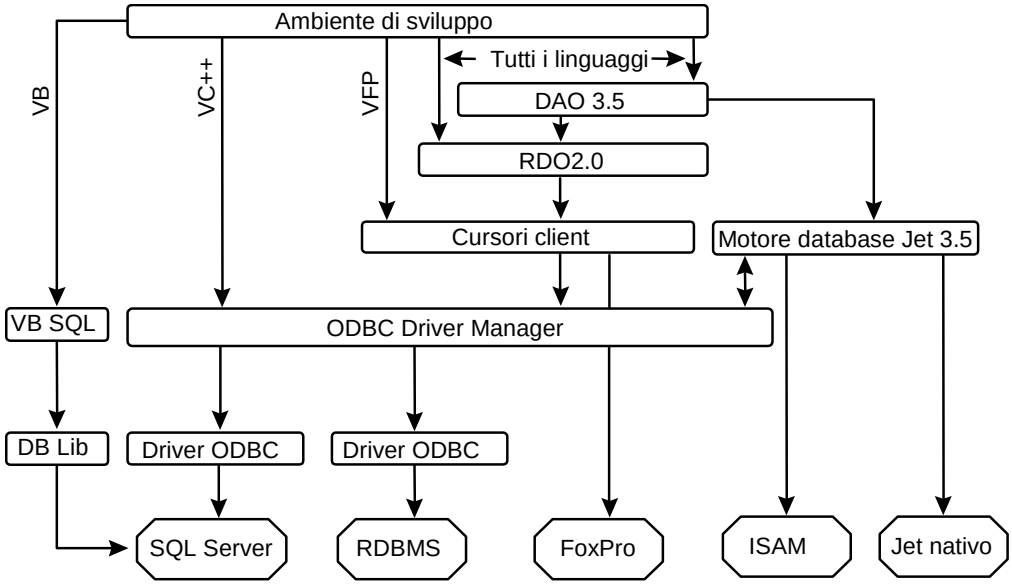
Le interfacce di accesso ai dati basate sulla tecnologia ActiveX rappresentano la base ideale per gli sviluppatori di applicazioni, perché forniscono l'architettura di base per un modello di programmazione di alto livello, indipendente dal linguaggio e basato sugli oggetti. Le singole implementazioni delle interfacce di accesso ai dati basate su ActiveX hanno reso disponibili funzionalità aggiuntive che vanno oltre l'obiettivo raggiunto da ODBC, ovvero l'accesso standardizzato a fonti dati eterogenee.

DAO

È stata la prima interfaccia orientata agli oggetti basata sul motore di database Microsoft Jet. La combinazione di elementi di Jet come il processore di query multifunzionale, il data warehousing e il motore di cursori locale offre la possibilità di utilizzare funzionalità robuste

con le fonti dati. Tra queste funzionalità ricordiamo l'esecuzione di query e aggiornamenti distribuiti su database, la gestione di dati locali e l'accesso a una vasta gamma di dati tra i quali tutti i famosi dati ISAM e tutti i dati basati su ODBC, per esempio Informix, SQL Server e Oracle. Nelle versioni precedenti, l'accesso DAO a dati remoti coinvolgeva tutte le funzionalità estese del motore Jet. Per ottimizzare il controllo e la velocità, gli sviluppatori volevano avere la possibilità di escludere Jet nella creazione di applicazioni basate sulle transazioni e collegate a un sistema RDBMS. La relativa semplicità d'uso e la flessibilità hanno però un prezzo nell'efficienza che si rivela mediocre, si comporta bene per l'accesso in locale a database di piccole dimensioni.

RDO *(Remote Data Objects)*



Interfaccia orientata agli oggetti che richiama direttamente ODBC per ottimizzare la velocità, il controllo e la semplicità di programmazione. DAO e RDO sono simili a livello di progettazione. I metodi RDO sono stati integrati in DAO e denominati ODBC Direct. Gli sviluppatori possono ora decidere se utilizzare il motore Jet per le operazioni locali di data warehousing, di replica, di query e aggiornamento oppure se utilizzare ODBC Direct per avvalersi di funzioni ottimizzate per la gestione dei dati remoti. Anche se DAO e RDO soddisfano molti dei requisiti attuali, tali interfacce sono collegate a tecnologie specifiche come Jet, Client Batch Cursors e ODBC. Le più nuove esigenze come applicazioni Internet robuste e accesso a più tipi di dati (oltre alle capacità basate su SQL di ODBC) richiedono un'architettura ben più flessibile. Più veloce di DAO, utilizza una quantità inferiore di risorse e fornisce maggiore controllo sui database client/server, per contro è più difficile da utilizzare.

ADO

È la nuova tecnologia di accesso ai dati: approccio universale dato che può connettersi a qualsiasi tipo di database relazionale e non relazionale, locale e remoto, alla posta elettronica e al file system. Il concetto alla base di ADO è quello di adattarsi a un ambiente in cui il gruppo di base delle interfacce ad oggetti sia standardizzato, ma facilmente estensibile in presenza di nuovi requisiti delle applicazioni. In questo modo è possibile avere implementazioni multiple di ADO, ognuna delle quali specializzata per un utilizzo specifico, come server e client Internet, client/server, desktop o transazioni distribuite. ADO è destinata alle applicazioni di reti aziendali e si concentra sulle implementazioni Internet, grazie alla possibilità di mantenere informazioni sullo stato in un ambiente senza connessioni. Questa caratteristica include un'implementazione con funzionalità complete di manipolazione dei dati e un'implementazione leggera e scaricabile, disponibile per i client Internet in fase di run-time. Possibilità di gestire eventi prima e dopo l'esecuzione di un comando o di un metodo. Possibilità di gestire, richiedere e fornire **recordset** (insieme dei risultati che sono restituiti quando è interrogato il database) annidati e strutturati

gerarchicamente secondo una relazione parent/child. Maggiore flessibilità nella creazione e modifica dei recordset. Possibilità di rendere persistente un recordset attraverso appositi metodi e funzionalità predefinite.
Possibilità d'indicizzare le righe di un recordset. Possibilità di effettuare ricerche veloci e d'impostare filtri su campi indicizzati. Possibilità d'intervenire all'interno del processo di connessione tra RDS e il database impostando e verificando diritti d'accesso, eseguendo codice custom.
Miglior supporto per la programmazione in linguaggi come il C++ e Java. ADO è anche un insieme di oggetti di accesso ai dati che costituiscono un modello di programmazione. Tali oggetti sono basati su OLE DB, che opera a livello basso attraverso le API: è un'architettura di componenti in linguaggio C++ destinata in particolare all'utilizzo da parte di sviluppatori software di terze parti. Lo scopo di OLE DB è quello di estendere le potenzialità delle applicazioni oltre i confini SQL di ODBC, lavora con diverse fonti dati come MS SQL Server , MS Access, MS FoxPro, Oracle e, tramite SNA Server 4.0, AS/400 e database VSAM. Sono già a disposizione diversi provider OLE DB forniti da terze parti per accedere, ad esempio a DB Informix, Sybase, IBM DB/2. Tutta la tecnologia è infatti costruita ad oggetti; ognuno dei quali preposto ad un compito specifico. In questo modo è possibile costruire e lavorare con tre componenti.

Data Providers (Fornitori di dati)

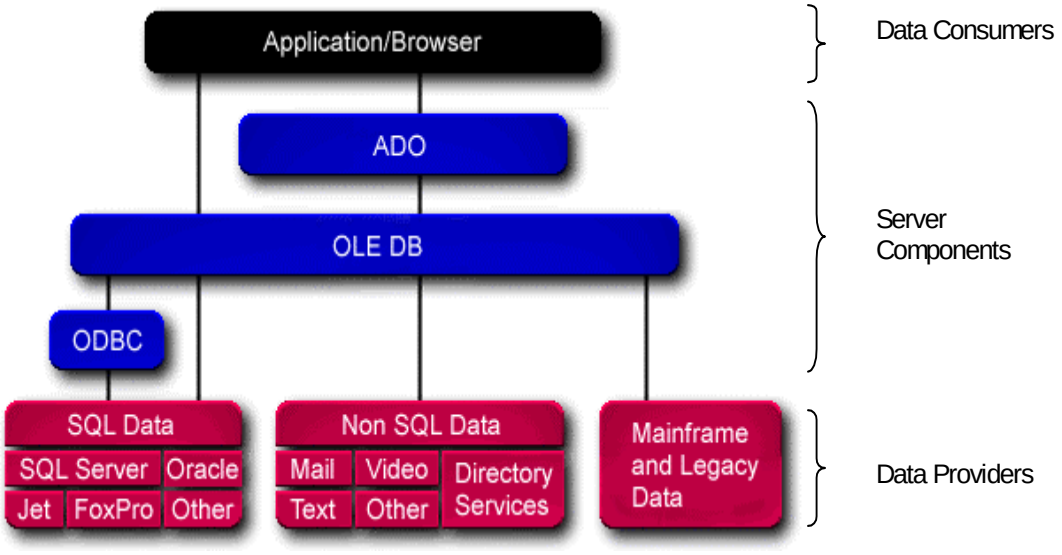
Sono oggetti che contengono ed espongono i dati al mondo esterno. Essi implementano logiche indipendenti di accesso ai dati, ma espongono quanto elaborato attraverso tabelle virtuali standard. Sono questi i componenti che si preoccupano del reperimento dei dati dalla loro locazione originale e della loro pubblicazione.

Server Component (Componenti di servizi)

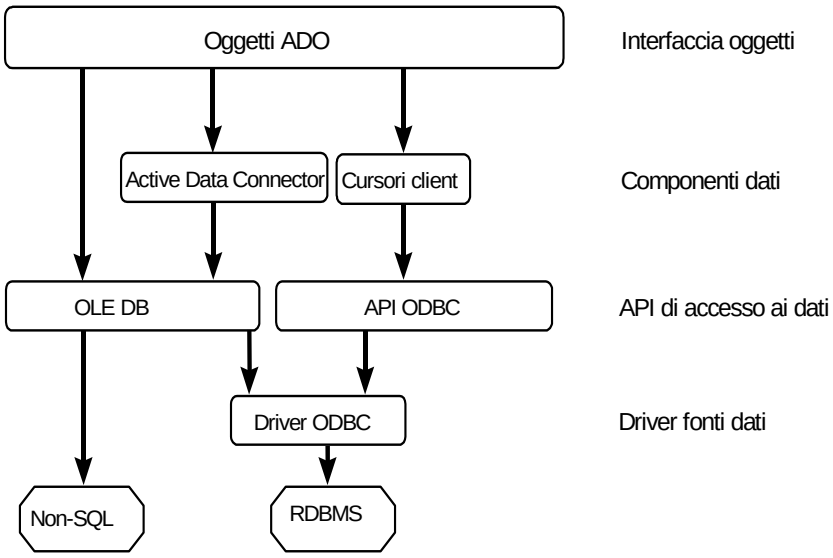
Questi componenti elaborano e trasportano i dati, sostituendo od integrando le logiche DBMS delle fonti dati con cui lavorano.

Data Consumer (Utilizzatori dei dati)

Rappresentano le applicazioni, (strumenti di sviluppo, linguaggi, browser) che accedono alle informazioni contenute nei Data Providers. Ogni soluzione che richiede accesso a dati può essere definita Data Consumer.



Un elemento interessante di ADO è un componente di database basato su OLE DB denominato Advanced Data Connector, che semplifica la realizzazione di comunicazioni efficienti su Internet e fornisce un meccanismo di associazione dei dati che consente di aggregare oggetti di sviluppo database direttamente a componenti visuali del browser.



Per le applicazioni destinate a Internet, ADO è decisamente la scelta migliore perché è un elemento integrante di IIS e di Microsoft Visual Studio, la famiglia di prodotti per lo sviluppo di applicazioni client/server e web. ADO include.

1. Implementazioni leggere sul lato client.
2. Funzionalità complete per il lato server.
3. Funzionalità di accesso ai dati per componenti ActiveX basati su browser.

CONNESSIONE AD UN DB ACCESS CON VISUAL BASIC

INTRODUZIONE

Visual BASIC permette di realizzare applicazioni professionali in grado di accedere a DB. Per esempio, la comodità per tutti i PC di un ufficio di poter accedere ad uno stesso archivio e su di esso effettuare delle modifiche che saranno visibili tutti.

CONTROLLO ADO DATA

1) Selezionare la libreria di oggetti ADO

È distribuito sotto forma di due differenti librerie.

1. Microsoft ActiveX Data Objects 2.7 Library, contiene tutti gli oggetti ADO.
2. Microsoft ActiveX Data Objects Recordset 2.7 Library, contiene solamente il supporto per i recordset, è meno costosa in termini di risorse di sistema.

Una volta selezionata la libreria, tutti gli oggetti ADO, i metodi e le proprietà relativi saranno accessibili attraverso la finestra *Visualizzatore oggetti*.

Creare un nuovo progetto contenente un form vuoto.

Progetto/Riferimenti/ Microsoft ActiveX Data Objects 2.7 Library

2) Creazione di una fonte di dati OLE DB

Questa operazione deve essere effettuata una sola volta.

Visualizza/Finestra Visualizzazione dati

La finestra *Visualizzazione dati* mette a disposizione un metodo per gestire tutte le connessioni a database (aprire, creare, modificare gli oggetti di accesso) necessarie per l'applicazione, senza uscire dall'ambiente di sviluppo; è uno strumento centralizzato organizzata sotto forma di cartelle.

Diagrammi di database.

Tabelle.

Viste.

Stored procedure: **procedure memorizzate sul server di database**, permettono di gestire, accedere o visualizzare informazioni riguardanti il database stesso e gli utenti che lo utilizzano. Possono contenere codice del linguaggio di programmazione del database, logica e query SQL che possono essere eseguite sui dati del database; possono accettare valori d'ingresso, generare parametri, restituire risultati e valori di ritorno. Il debugger **Transact-SQL (T-SQL)** è uno strumento di verifica delle stored procedure accessibile direttamente da Visual BASIC, deve essere installato sia sul lato client sia sul lato server.

Triggers: è un tipo particolare di stored procedure che è attivato quando si modificano i dati di una tabella specificata, eseguendo una o più operazioni di aggiornamento, inserimento o eliminazione. Utili a preservare l'integrità referenziale e la coerenza dei dati.

Nella finestra, fare clic con il pulsante destro del mouse, selezionare *Aggiungi Data Link…*

Nella scheda *Provider*, selezionare *Microsoft Jet 3.51 OLE DB Provider,* clic su Avanti

Nella scheda *Connessione*, selezionare il database *db11997.mdb* sarà chiesto a questo punto di specificare il nome utente e la password; i valori proposti possono rimanere inalterati se, come in questo caso, per il database non è impostato alcun criterio di protezione.

Fare clic sul pulsante *Verifica connessione*, affinchè sia corretta.

Fare clic su *OK* per chiudere la finestra.

Nel caso della connessione effettuata, è possibile solamente visualizzare le impostazioni, questo perché *Visualizzazione dati* dialoga con il database tramite linguaggio **DDL** (*Data Definition Language*) **ANSI**.

3) Creazione di un'interfaccia utente (front end) di database

Aggiungere il controllo nella casella degli strumenti

Progetto/Componenti…/Microsoft ADO Data Control 6.0 (OLEDB)

Posizionare nel form un'istanza del controllo ADO, *Adodc1*
Nella finestra *Proprietà Adodc1* inserire *Name = datPrimaryRS, Caption = " ", ConnectionString*, fare clic sui tre puntini, nella scheda *Generale*, selezionare *Usa stringa di connessione, Genera…,* selezionare *Microsoft Jet 3.51 OLE DB Provider,* clic su Avanti, selezionare il database *db11997.mdb*, OK, OK
Microsoft Jet 3.51 OLE DB Provider ci si connette a database Access 97
Microsoft Jet 4.0 OLE DB Provider ci si connette a database Access 2000
Nella finestra *Proprietà datPrimaryRS*, selezionare *Recordsource*, fare clic sui tre puntini, nella scheda *Recordsource*, *Tipo Command = 2 – adCmdTable, Nome tabella = exallievi*
Oppure
Nella finestra *Proprietà datPrimaryRS*, selezionare *Recordsource*, fare clic sui tre puntini, nella scheda *Recordsource*, *Tipo Command = 8 – adCmdUnknown, Testo Command (SQL) = SELECT * FROM exallievi WHERE CAP = '28010'*
Il controllo DATA non visualizza direttamente i dati, è soltanto un intermediario tra il database e i dati veri e propri rappresentati con gli oggetti soliti. Per questo motivo, posizionare nel form due controlli *Label*, con *Caption* Cognome e Nome
Posizionare nel form due controlli *TextBox*, con *DataSource datPrimaryRS, DataField Cognome e Nome*
Il controllo DATA consente di effettuare altre operazioni oltre la semplice lettura del database: aggiungere, eliminare, aggiornare e rivisualizzare il database. Posizionare nel form cinque CommandButton.

```
Option Explicit
Private Sub Command1_Click()
   On Error GoTo aggiungi
   datPrimaryRS.Recordset.AddNew
   Exit Sub
aggiungi:
   MsgBox Err.Description
End Sub
Private Sub Command2_Click()
   On Error GoTo elimina
   With datPrimaryRS.Recordset
      .Delete                          'cancella la riga della tabella
      .MoveNext
   If .EOF Then .MoveLast
   End With
   Exit Sub
elimina:
   MsgBox Err.Description
End Sub
Private Sub Command3_Click()
   On Error GoTo aggiorna
   datPrimaryRS.Recordset.UpdateBatch adAffectAll
   Exit Sub
aggiorna:
   MsgBox Err.Description
End Sub
Private Sub Command4_Click()
   'Necessario solo per applicazioni multiutente
   On Error GoTo rivisualizza
   datPrimaryRS.Refresh
   Exit Sub
rivisualizza:
   MsgBox Err.Description
```

```
End Sub
Private Sub Command5_Click()
   Unload Me
End Sub
```

4) Codice ADO (alternativa, impostare le proprietà dal codice)

Oltre ad utilizzare unicamente l'interfaccia del controllo DATA per la gestione di un database è possibile effettuare tutte le operazioni del controllo DATA mediante un insieme d'istruzioni dedicate all'utilizzo dei database.

```
Option Explicit
Dim WithEvents adoPrimaryRS As Recordset
'WithEvents specifica che è una variabile oggetto utilizzata per rispondere agli eventi
'generati da un oggetto ActiveX.
Dim mbEditFlag As Boolean
Dim mbAddNewFlag As Boolean
Dim mbDataChanged As Boolean
Private Sub Form_Load()
  Dim oText As TextBox
  Dim db As Connection
  Set db = New Connection
  db.CursorLocation = adUseClient
  db.Open "PROVIDER=Microsoft.Jet.OLEDB.3.51;
        Data Source= " & App.Path & "\db11997.mdb;"
  Set adoPrimaryRS = New Recordset
  adoPrimaryRS.Open "select Cognome,Nome from exallievi", db,
  adOpenStatic, adLockOptimistic
  'Associa le caselle di testo al fornitore di dati
  For Each oText In Me.txtFields
    Set oText.DataSource = adoPrimaryRS
  Next
  mbDataChanged = False
End Sub
Private Sub adoPrimaryRS_MoveComplete(ByVal adReason As
ADODB.EventReasonEnum, ByVal pError As ADODB.Error, adStatus As
ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)
  'Visualizza la posizione del record corrente per questo gruppo di record
  lblStatus.Caption = "Record: " & CStr(adoPrimaryRS.AbsolutePosition)
End Sub
Private Sub cmdAdd_Click()
  'Aggiorna
  On Error GoTo AddErr
  With adoPrimaryRS
    If Not (.BOF And .EOF) Then
      mvBookMark = .Bookmark
    End If
    .AddNew
    lblStatus.Caption = "Aggiunge il record"
    mbAddNewFlag = True
  End With
  Exit Sub
AddErr:
  MsgBox Err.Description
End Sub
Private Sub cmdDelete_Click()
  'Elimina
```

```
  On Error GoTo DeleteErr
  With adoPrimaryRS
    .Delete
    .MoveNext                                   'record successivo
    If .EOF Then .MoveLast
  End With
  Exit Sub
DeleteErr:
  MsgBox Err.Description
End Sub
Private Sub cmdRefresh_Click()
  'Necessario solo per applicazioni multiutente: Rivisualizza
  On Error GoTo RefreshErr
  adoPrimaryRS.Requery
  Exit Sub
RefreshErr:
  MsgBox Err.Description
End Sub
Private Sub cmdEdit_Click()
  'Modifica
  On Error GoTo EditErr
  lblStatus.Caption = "Modifica il record"
  mbEditFlag = True
  Exit Sub
EditErr:
  MsgBox Err.Description
End Sub
Private Sub cmdClose_Click()
  Unload Me
End Sub
Private Sub cmdFirst_Click()
  'Primo record
  On Error GoTo GoFirstError
  adoPrimaryRS.MoveFirst
  mbDataChanged = False
  Exit Sub
GoFirstError:
  MsgBox Err.Description
End Sub
Private Sub cmdLast_Click()
  'Ultimo record
  On Error GoTo GoLastError
  adoPrimaryRS.MoveLast
  mbDataChanged = False
  Exit Sub
GoLastError:
  MsgBox Err.Description
End Sub
Private Sub cmdNext_Click()
  'Record successivo
  On Error GoTo GoNextError
  If (Not adoPrimaryRS.EOF) Then adoPrimaryRS.MoveNext
'RecordCount restituisce il numero di righe della tabella selezionata
  If (adoPrimaryRS.EOF And adoPrimaryRS.RecordCount > 0) Then
```

```
    Beep
     'è stata superata la fine e quindi torna indietro
    adoPrimaryRS.MoveLast
  End If
  'visualizza il record corrente
  mbDataChanged = False
  Exit Sub
GoNextError:
  MsgBox Err.Description
End Sub
Private Sub cmdPrevious_Click()
  'Record precedente
  On Error GoTo GoPrevError
  If (Not adoPrimaryRS.BOF) Then adoPrimaryRS.MovePrevious
  If (adoPrimaryRS.BOF And adoPrimaryRS.RecordCount > 0) Then
    Beep
    'è stata superata la fine e quindi torna indietro
    adoPrimaryRS.MoveFirst
  End If
  'visualizza il record corrente
  mbDataChanged = False
  Exit Sub
GoPrevError:
  MsgBox Err.Description
End Sub
```

RDS (*Remote Data Service)* è una tecnologia basata su controlli ActiveX che utilizza ADO per trasferire dati dal provider al client. Il client effettua il caching dei dati e li manipola ordinandoli o filtrandoli. Il processo di *remotizzazione (remoting)* prevede il passaggio di parametri tra due differenti processi, che è effettuato attraverso una rete tradizionale o via Internet. Per esempio, il client effettua una richiesta di dati passando una serie di parametri allo strato intermedio il quale accetta la richiesta e la inoltra allo strato di gestione dei dati; quest'ultimo, a sua volta, recupera i dati richiesti e li passa allo strato intermedio.

Codice dell'applicazione del lato server.

```
Option Explicit
' Questa porzione di codice deve essere inserita in un file .bas
' Definizione di un tipo definito dall'utente (UDT)
Public Type udtName
   ' Definizione della tabella utilizzata
   SSN As String
   Last Name As String
   First Name As String
   Sex As String
End Type
Public Function PassUDT(myrec As udtName) As udtName
   ' Il codice per modificare i dati deve essere posizionato qui
   ' Restituisce il tipo definito dall'utente
   PassUDT = myrec
End Function
```

Codice dell'applicazione del lato client.

```
Option Explicit
Private myrec As udtName
Public Sub Command1_Click()
   Dim x As udtName
```

x = PassUDT (myrec)
' Codice per qualsiasi elaborazione
End Sub
Un recordset ADO può essere utilizzato anche da HTML o DHTML per accedere ai dati di un'applicazione web server attraverso Internet o intranet. Si usa la libreria Microsoft ActiveX Data Access Recordset 2.0 Library, che include solamente l'oggetto Recordset, e non gestisce l'oggetto Connection, quindi crea applicazioni più leggere in termini di dimensioni.

CONTROLLO DATALIST

Consiste in una lista che può essere associata ad un campo di un recordset.
Creare un nuovo progetto contenente un form vuoto.
Visualizza/Finestra Visualizzazione dati
Nella finestra, fare clic con il pulsante destro del mouse, selezionare *Aggiungi Data Link…*
Nella scheda *Provider*, selezionare *Microsoft Jet 3.51 OLE DB Provider,* clic su Avanti
Nella scheda *Connessione*, selezionare il database *db11997.mdb* sarà chiesto a questo punto di specificare il nome utente e la password; i valori proposti possono rimanere inalterati se, come in questo caso, per il database non è impostato alcun criterio di protezione.
Fare clic sul pulsante *Verifica connessione*, affinchè sia corretta.
Fare clic su *OK* per chiudere la finestra.
Progetto/Componenti…/Microsoft ADO Data Control 6.0 (OLEDB)
Posizionare nel form un'istanza del controllo ADO, *Adodc1*
Nella finestra *Proprietà Adodc1* selezionare *ConnectionString*, fare clic sui tre puntini, nella scheda *Generale*, selezionare *Usa stringa di connessione, Genera…,* selezionare *Microsoft Jet 3.51 OLE DB Provider,* clic su Avanti, selezionare il database *db11997.mdb*, OK, OK
Nella finestra *Proprietà Adodc1*, selezionare *Recordsource*, fare clic sui tre puntini, nella scheda *Recordsource*, *Tipo Command = 2 – adCmdTable, Nome tabella = exallievi*
Progetto/Componenti…/Microsoft DataList Controls 6.0 (OLEDB)
Posizionare nel form il controllo *DataList1*
Nella finestra *Proprietà DataList1* selezionare *DataSource* e *RowSource* con *Adodc1*, selezionare *ListFied* su Cognome.

CONTROLLO DATACOMBO

Consiste in una ComboBox che può essere associata ad un campo di un recordset.
Posizionare nel form il controllo *DataCombo1*
Nella finestra *Proprietà DataCombo1* selezionare *DataSource* e *RowSource* con *Adodc1*, selezionare *ListFied* su Cognome.

CONTROLLO DATAGRID

Visualizza informazioni del database in formato tabellare, all'interno di una griglia. Ogni riga è un record, ogni colonna è un campo del record. Il controllo è riempito con dati provenienti da un oggetto recordset e anche le intestazioni delle colonne possono essere ottenute nello stesso modo.
Progetto/Componenti…/Microsoft DataGrid Control 6.0 (OLEDB)
Posizionare nel form il controllo *DataGridt1*
Nella finestra *Proprietà Datagrid1* selezionare *DataSource* con *Adodc1*.
Eseguire l'applicazione, il controllo ha automaticamente impostato le colonne in base ai campi della tabella.
AllowDelete = False l'utente non può cancellare (inserire) record (righe) della griglia.
AllowUpdate = False l'utente non può modificare record (righe) della griglia.
AllowRowSizing = False l'utente non può ridimensionare le righe della griglia.

AllowArrows = True l'utente può utilizzare i tasti cursore per navigare nella griglia.
TabAction = 1 o 2 l'utente può utilizzare il tasto tabulazione per navigare nella griglia.
ColumnHeaders, HeadFont, HeadLines specificano le intestazioni delle colonne che rimangono fisse nella parte alta della griglia.
DefColWidth imposta la larghezza predefinita delle colonne.
RowDividerStyle specifica l'aspetto dei bordi che dividono le righe della griglia.
È possibile intervenire sull'aspetto delle singole colonne (sono nuimerate a partire da zero) con l'oggetto *Column*, tramite codice.

```
Private Sub Form_Load()
   'Rimuove le ultime tutte le colonne eccetto le prime due
   While (DataGrid1.Columns.Count > 2)
      DataGrid1.Columns.Remove (DataGrid1.Columns.Count - 1)
   Wend
   'Imposta la proprietà per la prima colonna
   With DataGrid1.Columns(0)
      .Caption = "Cognome"
      .DataField = "Cognome"
      .Width = 3000
   End With
   'Imposta la proprietà per la seconda colonna
   With DataGrid1.Columns(1)
      .Caption = "Nome"
      .DataField = "Nome"
      .Width = 2000
   End With
        'Imposta il layout corrente della griglia come predefinito
   DataGrid1.HoldFields
        'Ricrea il legame delle colonne ai nuovi campi
   DataGrid1.ReBind
End Sub
```

La singola cella selezionata è chiamata la cella corrente, indicata dalle proprietà *Row* e *Col*. Il loro valore indica la posizione della cella corrente relativamente alle righe e colonne visualizzate nella griglia. Per determinare il numero di riga effettivo della cella corrente bisogna sommare i valori di *Row* e *FirstRow*, mentre per determinare il numero di colonna effettivo bisogna sommare i valori di *Col* e *LeftCol*. Per verificare se la cella corrente è visibile *CurrenteCellVisible = True*. Per verificare se il contenuto della cella corrente è stato modificato *EditActive*.

TRANSAZIONE

Rappresenta un insieme di operazioni che devono essere eseguite, tutte o nessuna, perché la loro esecuzione parziale causerebbe gravi incoerenze nel database. Per esempio, il conto corrente di una banca dove un impiegato è in grado di effettuare bonifici (spostamento di denaro da un conto ad un altro) tra i conti della banca stessa. Durante il bonifico l'importo del conto appartenente alla persona che effettua il bonifico deve essere diminuito della quantità di denaro da spostare. Contemporaneamente l'importo del conto della persona che riceve il denaro deve essere incrementato della quantità di denaro del bonifico. Se tra la prima operazione e la seconda qualcosa non funziona correttamente i soldi sono andati nel nulla!
Visual BASIC mette a disposizione due funzioni per garantire la contemporaneità dell'esecuzione di una serie di operazioni su di un database. Per realizzare una sessione, in altre parole una transazione si utilizza il metodo *BeginTrans* dell'oggetto Workspace. Questo metodo comunica al motore del database di non scrivere fisicamente le informazioni sul database fino a quando non è conclusa la transazione con successo. Se una delle transazioni fallisce si esegue il metodo *RollBack* dell'oggetto Workspace che

ripristina il sistema portandolo al medesimo stato in cui si trovava prima della transazione. Se invece le operazioni terminano con successo si esegue un *CommitTrans* che scrive fisicamente sul disco l'avvenuto bonifico.

```
Modulo1.bas
Public conto As Database
Public transa As Recordset
Form1.frm
'Progettare l'applicazione che esegue un bonifico bancario di 200,00 euro
'dal conto numero 1 al conto numero 2
'Inserire Progetto/Componenti Microsoft DAO 3.6 Object Library
Option Explicit
Private Sub Command1_Click()
   Dim creditore As Recordset
   Dim debitore As Recordset
   Set creditore = conto.OpenRecordset("SELECT Transazioni.Somma,
Transazioni.NumeroConto From Transazioni WHERE (((Transazioni.NumeroConto)=2));")
   Set debitore = conto.OpenRecordset("SELECT Transazioni.Somma,
Transazioni.NumeroConto From Transazioni WHERE (((Transazioni.NumeroConto)=1));")
   Text1.Text = debitore![Somma]
   Text2.Text = creditore![Somma]
   Workspaces(0).BeginTrans
   creditore.Edit
   debitore.Edit
   creditore![Somma] = creditore![Somma] + Val(Text3.Text)
   debitore![Somma] = debitore![Somma] - Val(Text3.Text)
   debitore.Update
   creditore.Update
   Workspaces(0).CommitTrans
   Text4.Text = debitore![Somma]
   Text5.Text = creditore![Somma]
   creditore.Close
   debitore.Close
End Sub
Private Sub Form_Load()
   Set conto = OpenDatabase(App.Path & "\db1.mdb")
   Set transa = conto.OpenRecordset("transazioni", dbOpenDynaset)
End Sub
```

L'AMBIENTE DI PROGETTAZIONE DATA ENVIRONMENT

È un'interfaccia che rappresenta uno strumento interattivo per la creazione di strutture per accesso ai dati all'interno delle applicazioni con poche e semplici operazioni visuali a design time (tutte operazioni che normalmente dovevano essere fatte tramite codice). In fase di progettazione è possibile impostare i valori delle proprietà degli oggetti, scrivere codice, eseguire comandi e rielaborare i dati organizzandoli in strutture gerarchiche. È inoltre possibile trascinare (drag and drop) oggetti dal Data Environment a un form per creare controlli data-bound (associati ai dati). Raggruppa numerose funzioni.

Definire la connessione al database.

Creare comandi SQL per accedere ai dati.

Specificare come differenti comandi siano collegati per formare query più complesse.

Definire funzioni di aggregazione per le query.

Specificare l'ordinamento dei dati delle query.

1) Definizione della connessione

Creare un nuovo progetto contenente un form vuoto.

Per accedere ai dati è necessario creare un oggetto *Connection* destinato a rappresentare

la connessione al database che sarà utilizzata come sorgente di dati per i comandi associati. Una nuova connessione chiamata "*Connection1*" è creata automaticamente quando si crea un'istanza di Data Environment nel progetto. In fase di progettazione, il Data Environment aprirà una connessione per ottenere i "metadati", ovvero le informazioni sulla struttura del database, necessari per poter visualizzare la struttura delle tabelle e degli altri oggetti del database.
Fare clic con il pulsante destro del mouse nella finestra *Progetto*, scegliere la voce *Inserisci/ Data Environment*
Oppure
Visualizza/Finestra Visualizzazione dati, pulsante *Aggiungi Data Environment al progetto corrente*
Nella finestra *DataEnvironment1*, fare clic con il pulsante destro del mouse su *Connection1*, selezionare *Proprietà*
Nella scheda *Provider*, selezionare *Microsoft Jet 3.51 OLE DB Provider,* clic su Avanti
Nella scheda *Connessione*, selezionare il database *db11997.mdb* sarà chiesto a questo punto di specificare il nome utente e la password; i valori proposti possono rimanere inalterati se, come in questo caso, per il database non è impostato alcun criterio di protezione.
Fare clic sul pulsante *Verifica connessione*, affinchè sia corretta.
Fare clic su *OK* per chiudere la finestra.
La struttura del Data Environment è identica a quella di Visualizzazione dati. Una fondamentale differenza riguarda il fatto che il Data Environment è in realtà un file DSR (Designer) integrato nel progetto e dunque sarà compilato con esso. Naturalmente è possibile variare in qualsiasi momento le proprietà di ogni elemento del Data Environment. È altrettanto interessante notare che in questo stesso designer è possibile gestire dati provenienti da origini diverse: ad esempio, si potrebbe avere la tabella clienti proveniente da un Database VSAM AS/400 e la tabella ordini proveniente da SQL Server. Il Data Environment esporrebbe due diverse connessioni dando comunque la possibilità di creare relazioni *Data Shape*. Queste ultime sono delle relazioni ADO che assomigliano molto alle istruzioni JOIN utilizzate nel linguaggio SQL. Per permettere di unire dati provenienti da fonti dati e sistemi diversi, Microsoft ha creato questo set di istruzioni proprietario di ADO, che consente appunto di unire tramite relazioni diversi oggetti *Command.*

2) Integrare Data Environment con i form

Dalla finestra *Visualizzazione dati* trascinare la tabella exallievi nella finestra *DataEnvironment1*.
Dalla finestra *DataEnvironment1* trascinare i campi Cognome e Nome nel form.
Con poche righe di codice posso navigare nel database. Posizionare nel form tre CommandButton.

```
Option Explicit
Private Sub Command1_Click()
  'Record successivo
  If (Not DataEnvironment1.rsexallievi.EOF) Then
                DataEnvironment1.rsexallievi.MoveNext
  If (DataEnvironment1.rsexallievi.EOF And
     DataEnvironment1.rsexallievi.RecordCount > 0) Then
         DataEnvironment1.rsexallievi.MoveLast
  End If
End Sub
Private Sub Command2_Click()
  'Record precedente
  If (Not DataEnvironment1.rsexallievi.BOF) Then
                DataEnvironment1.rsexallievi.MovePrevious
  If (DataEnvironment1.rsexallievi.BOF And
     DataEnvironment1.rsexallievi.RecordCount > 0) Then
```

```
            DataEnvironment1.rsexallievi.MoveFirst
  End If
End Sub
Private Sub Command3_Click()
  Unload Me
End Sub
```

3) Impostare le proprietà dal codice

Una volta aggiunto un oggetto Data Environment all'applicazione, è possibile fare riferimento alle connessioni e ai comandi relativi all'interno del codice per accedere ai dati del database. In fase di esecuzione, Data Environment crea gli oggetti *Command* e *Connection* relativi a ogni comando e connessione definiti al suo interno. Per collegare i controlli ad un campo di un oggetto *Command* è necessario agire su tre proprietà.

1. *DataSource*: specifica il nome del Data Environment di origine, *DataEnvironment1*
2. *DataMember*: nome dell'oggetto *Command* da utilizzare, exallievi
3. *DataField*: nome del campo da gestire, Cognome, Nome.

DataSource non deve essere necessariamente un oggetto di design time. È possibile quindi assegnare 'al volo' un oggetto Data Provider ad un oggetto, generalmente un controllo, che dispone della proprietà DataSource. Le tre proprietà sopra elencate sono automaticamente impostate quando si trascina un *Command* o un *Field* su un form.

Accedendo a un recordset ADO dal codice, i nomi degli oggetti *recordset* sono gestiti con il prefisso *rs* per distinguerli dai rispettivi oggetti *Command*. Per esempio, un oggetto *Command* chiamato *Ordine* crea un oggetto *recordset* chiamato *rsOrdine*.

```
Option Explicit
Private Sub Command1_Click()
    'Record successivo
  If (Not DataEnvironment1.rsexallievi.EOF) Then
            DataEnvironment1.rsexallievi.MoveNext
  If (DataEnvironment1.rsexallievi.EOF And
    DataEnvironment1.rsexallievi.RecordCount > 0) Then
            DataEnvironment1.rsexallievi.MoveLast
  End If
  txtCognome.Text = DataEnvironment1.rsexallievi.Fields!Cognome
  txtNome.Text = DataEnvironment1.rsexallievi.Fields!Nome
End Sub
Private Sub Command2_Click()
  'Record precedente
  If (Not DataEnvironment1.rsexallievi.BOF) Then
            DataEnvironment1.rsexallievi.MovePrevious
  If (DataEnvironment1.rsexallievi.BOF And
    DataEnvironment1.rsexallievi.RecordCount > 0) Then
            DataEnvironment1.rsexallievi.MoveFirst
  End If
  txtCognome.Text = DataEnvironment1.rsexallievi.Fields!Cognome
  txtNome.Text = DataEnvironment1.rsexallievi.Fields!Nome
End Sub
Private Sub Command3_Click()
   Unload Me
End Sub
Private Sub Form_Load()
   txtCognome.Text = DataEnvironment1.rsexallievi.Fields!Cognome
   txtNome.Text = DataEnvironment1.rsexallievi.Fields!Nome
End Sub
```

Il codice scritto è semplice e chiaro e consente di operare tramite i metodi già conosciuti con DAO. È comunque altrettanto importante notare che questo codice *non conosce* quale

sia la fonte dati che è invece gestita dal Data Environment. Questo implica che il codice scritto è equivalente ad un oggetto User di un'architettura client-server three-tier.

4) Utilizzo di SQL Query Designer

È un'interfaccia interattiva d'interrogazione che permette di progettare quasi tutti i tipi di comandi SQL necessari per le elaborazioni di database di un'applicazione.

Creare query per leggere dati da qualsiasi database compatibile ODBC.

Specificare gli elementi, record e colonne, da leggere, gli ordinamenti da imporre sui dati e i filtri da applicare.

Consultare un'anteprima dei dati.

Definire join tra le tabelle e creare query su più tabelle.

Modificare i dati del database aggiornando, inserendo o cancellando record.

Creare query attive che modifichino il database.

Creare un nuovo progetto contenente un form vuoto.

Fare clic con il pulsante destro del mouse nella finestra *Progetto*, scegliere la voce *Inserisci/ Data Environment*

Oppure

Visualizza/Finestra Visualizzazione dati, pulsante *Aggiungi Data Environment al progetto corrente*

Nella finestra *DataEnvironment1*, fare clic con il pulsante destro del mouse su *Connection1*, selezionare *Proprietà*

Nella scheda *Provider*, selezionare *Microsoft Jet 3.51 OLE DB Provider,* clic su Avanti

Nella scheda *Connessione*, selezionare il database *db11997.mdb* sarà chiesto a questo punto di specificare il nome utente e la password; i valori proposti possono rimanere inalterati se, come in questo caso, per il database non è impostato alcun criterio di protezione.

Fare clic sul pulsante *Verifica connessione*, affinchè sia corretta.

Fare clic su *OK* per chiudere la finestra.

Dalla finestra *Visualizzazione dati* trascinare la tabella exallievi nella finestra *DataEnvironment1*.

Nella finestra *DataEnvironment1* clic sul pulsante *Aggiungi Command*

Nelle proprietà di *Command1* impostare *ConnectionName = Connection1*

Nella finestra *DataEnvironment1* fare clic con il pulsante destro del mouse, scegliere la voce *Proprietà*, nella finestra *Proprietà…* scheda *Generale*, selezionare *Istruzione SQL*, quindi *Generatore SQL…* appare la finestra di progettazione delle query SQL costituita da quattro riquadri.

Diagram: visualizza le fonti d'ingresso da interrogare.

Grid: contiene una griglia per specificare le opzioni della query.

SQL: visualizza il comando SQL relativo alla query corrente, permettendo anche la modifica. È possibile scrivere direttamente il comando SQL nel caso in cui fosse particolarmente complesso utilizzare i riquadri Diagram e Grid, che sono visualizzati in grigio per indicare che il loro contenuto non riflette la query che si sta creando.

Results: risultati della query, le informazioni del database possono essere modificate direttamente in questo riquadro intervenendo sul contenuto delle celle della griglia.

Tutti i riquadri sono sincronizzati, per cui una modifica all'interno di uno di essi provoca automaticamente l'aggiornamento degli altri, ad eccezione di Results che è aggiornato solo quando la query è eseguita.

Dalla finestra *Visualizzazione dati* trascinare la tabella exallievi in Diagram, le relazioni tra le tabelle sono visualizzate automaticamente se sono state definite nel database. Se nel database non sono state definite relazioni tra le tabelle, è possibile farlo nella query, trascinando la colonna di una delle tabelle verso la colonna di un'altra tabella che deve essere messa in relazione con la prima.

Trascinare Cognome, Nome e CAP nel riquadro Grid, ordinare per Cognome in ordine crescente e visualizzare solo chi ha il CAP = '28010'.

Query/Esegui

Chiudere SQL Query Designer e salvare le modifiche nell'oggetto *Command1*
Dalla finestra *DataEnvironment1* trascinare la query nel form.

Comandi secondari (Child command)

È una query SQL che dipende da un comando precedente (*parent*), formando una query gerarchica che potrà poi essere utilizzata per visualizzare dati.
Fare clic con il pulsante destro del mouse sul primo comando e dal menu contestuale selezionare *Aggiungi Command secondario*.
Nella finestra delle proprietà del *Command* secondario selezionare la scheda *Relazione*, all'interno della quale si specifica la relazione tra i due *Command*.
Un oggetto Command secondario deve contenere necessariamente almeno una colonna che sia in relazione con il Command principale.
Una volta che l'applicazione sarà eseguita, l'ambiente Data Environment creerà il comando *ADO Shape* utilizzato da ADO per creare il recordset gerarchico, per visualizzare entrambi fare clic destro sull'oggetto *Command* e scegliere la voce *Informazioni gerarchia*.
Il comando *Shape* definisce la struttura del recordset gerarchico e i comandi necessari per popolare tale struttura con i dati relativi, la query eseguita è contenuta in questo comando.
La sintassi della query dipende dal tipo di database, solitamente rispetta lo standard SQL, ma ciò può anche non avvenire, perché ADO non impone l'utilizzo di un linguaggio di query particolare. Inoltre, anche se sarebbe possibile utilizzare SQL JOIN per mettere in relazione due tabelle, un recordset gerarchico permette di rappresentare le informazioni in modo più efficiente. Infatti ogni record di un recordset creato con JOIN ripeterebbe in maniera ridondante le informazioni della tabella padre, mentre un recordset gerarchico utilizza un solo recordset padre per tutti gli oggetti recorset figli.

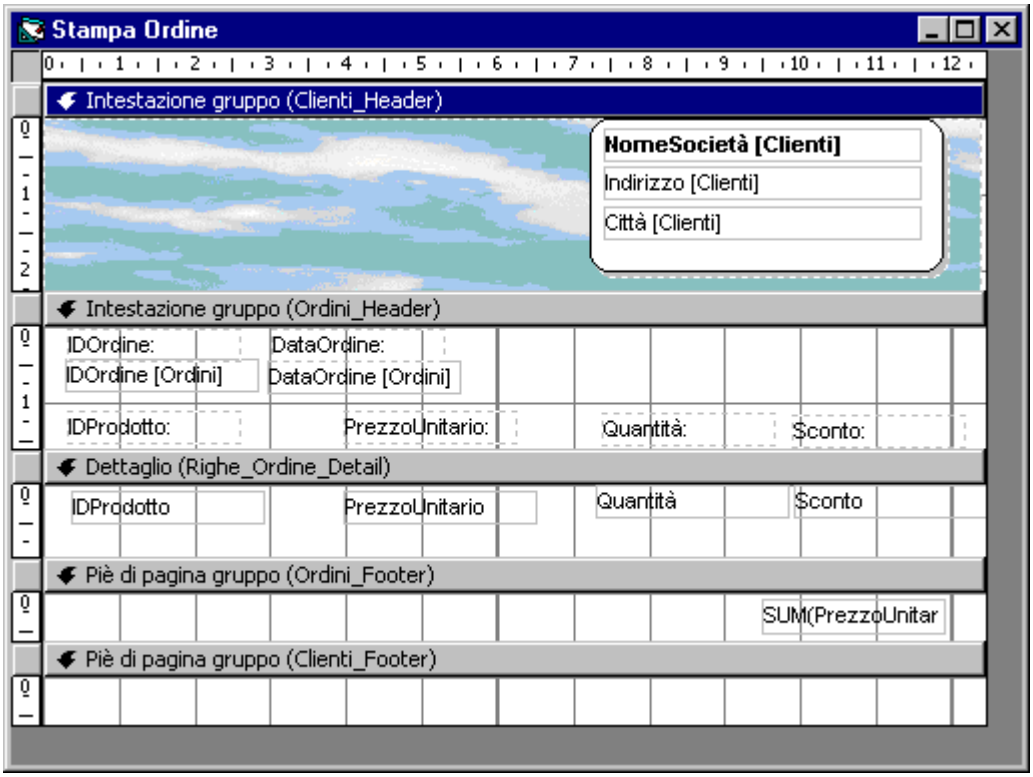
CONTROLLO HIERARCHICAL FLEXGRID

Visualizza i dati all'interno di una griglia formata da righe (record), colonne (campi) e bande, che si occupano di raggruppare campi provenienti da recordset gerarchici. L'utente può selezionare celle singole o gruppi di celle, ma non può modificare i dati.
Creare un nuovo progetto contenente un form vuoto.
Fare clic con il pulsante destro del mouse nella finestra *Progetto*, scegliere la voce *Inserisci/ Data Environment*
Oppure
Visualizza/Finestra Visualizzazione dati, pulsante *Aggiungi Data Environment al progetto corrente*
Nella finestra *DataEnvironment1*, fare clic con il pulsante destro del mouse su *Connection1*, selezionare *Proprietà*
Nella scheda *Provider*, selezionare *Microsoft Jet 3.51 OLE DB Provider,* clic su Avanti
Nella scheda *Connessione*, selezionare il database *db11997.mdb* sarà chiesto a questo punto di specificare il nome utente e la password; i valori proposti possono rimanere inalterati se, come in questo caso, per il database non è impostato alcun criterio di protezione.
Fare clic sul pulsante *Verifica connessione*, affinchè sia corretta.
Fare clic su *OK* per chiudere la finestra.
Dalla finestra *Visualizzazione dati* trascinare la tabella exallievi nella finestra *DataEnvironment1*.
Progetto/Componenti…/Microsoft Hierarchical FlexGrid Controls 6.0 (OLEDB)
Posizionare nel form il controllo *MSHFlexGrid1*
Nella finestra *Proprietà MSHFlexGrid1* selezionare *DataSource* con *DataEnvironment1*

UTILITY DATA REPORT

È un meccanismo integrato con l'ambiente di sviluppo che consente la generazione e la rapida manutenzione di Report, ha anche la possibilità di convertire ogni report in formato HTML. Vi sono diversi modi di costruire un report; il più semplice è quello di affidarsi,

anche in questo caso, ad un designer. Affidiamoci quindi al Data Environment appena creato per creare un report che mostri tutti gli ordini raggruppati per cliente. Le operazioni da fare sono leggermente diverse rispetto a quelle per la rappresentazione dei dati su un form. Finestra di progettazione, DataEnvironment1, clic con il pulsante destro del mouse, Inserisci data Report. A questo punto sarà necessario assegnare alla proprietà DataSource il nome dell'oggetto Data Environment e alla proprietà DataMember il nome dell'oggetto Command *principale*. Fatto questo è sufficiente cliccare con il pulsante destro del mouse sul Data Report Designer e selezionare il comando "Recupera Struttura". Questo crea tante nuove sezioni del report quanti sono i livelli di raggruppamento impostati nel Command. Mentre il DataEnvironment consente la relazione Uno-A-Molti tra padre e figli, nel report, per motivi di struttura fisica, è possibile inserire solamente relazioni Uno-A-Uno.



Nella figura si può notare un semplicissimo report in fase di design. Interessanti sono le possibilità d'inserire elementi grafici simili ai controlli image di Visual BASIC e campi formula come SUM(PrezzoUnitario). Dopo aver eseguito queste operazioni sarà effettuata l'anteprima di stampa del report scrivendo solamente una riga di codice:

derOrdini.Show

Per modificare il report è possibile reperire i campi da inserire dal Data Environment trascinandoli direttamente sul Designer. Visual BASIC controlla che i campi siano inseriti nella giusta sezione. È impossibile trascinare il campo *Ragione Sociale* sul dettaglio se tale informazione non è presente a quel livello di gerarchia. È comunque necessario scrivere codice per formattare, stampare, calcolare i dati del report a design time e, altra interessante novità, per esportare il contenuto del report in formato HTML. Quest'ultima possibilità è accessibile molto facilmente e, se integrata in un'applicazione che utilizza WebClass con Internet Information Server, consente una rapida, elegante e semplice soluzione di presentazione di informazioni in rete.

Owner Data Objects

Visual BASIC permette ora di creare delle normali classi (.CLS) che fungano da Data Provider. Per effettuarlo è sufficiente impostare la proprietà *DataSourceBehavior = 1 – vbDataSource*. In questo modo è creato un nuovo evento per la classe: *GetDataMember* che è attivato ogni volta che un Data Consumer richiede dei dati. I dati devono essere

passati al consumer tramite il parametro Data dell'evento GetDataMember. Sarà passato un oggetto Recordset in modo da avere un oggetto di riferimento contenente i dati richiesti.

```
Private rs As ADODB.Recordset
Private Sub Class_GetDataMember(DataMember As String, Data As Object)
    Set Data = rs
End Sub
```

Per popolare quest'oggetto è necessario scrivere del codice come quello che segue nell'evento Class_Initialize. In questo caso, ad esempio, è letto un file di testo che sarà poi esposto come Recordset. Il codice mostrato nei prossimi paragrafi è contenuto nella classe cProvider.

```
Private Sub Class_Initialize()
Dim nFile As Integer
nFile = FreeFile
Open "Anagrafico.txt" For Input As #nFile
Set rs = New ADODB.Recordset
' Viene creata la struttura del Recordset
With rs
    .Fields.Append "Codice", adInteger
    .Fields.Append "Nome", adBSTR, 50
    .Fields.Append "Cognome", adBSTR, 50
    .CursorType = adOpenStatic
    .LockType = adLockOptimistic
    .Open
End With
' Creo il contenuto del Recordset
Do While Not EOF(nFile)
    Input #nFile, Codice, Nome, Cognome
    With rs
        .AddNew
            .Fields("Codice") = Codice
            .Fields("Nome") = Nome
            .Fields("Cognome") = Cognome
        .Update
    End With
Loop
rs.MoveFirst
Close #nFile
End Sub
```

Con poche e semplici righe di codice è stato creato al volo un recordset contenente i dati di un file di testo, senza dover utilizzare driver ODBC. A questo punto è possibile creare i propri metodi ed eventi di navigazione e di gestione dei dati.

```
Public Event RecordMoved(Codice As String)
Public Sub MoveFirst()
    rs.MoveFirst
    RaiseEvent RecordMoved(rs!Codice)
End Sub
Public Sub MoveLast()
    rs.MoveLast
    RaiseEvent RecordMoved(rs!Codice)
End Sub
Public Sub MoveNext()
    rs.MoveNext
    RaiseEvent RecordMoved(rs!Codice)
```

```
End Sub
Public Sub MovePrevious()
   rs.MovePrevious
   RaiseEvent RecordMoved(rs!Codice)
End Sub
```
Per effettuare il binding, l'operazione cioè di associazione tra un campo ed una casella di testo, è sufficiente scrivere altrettanto poche righe di codice.
```
Private WithEvents Provider As cProvider
Private oBinding As BindingCollection
Private Sub Form_Load()
   ' Creo una nuova istanza della classe cProvider
   Set Provider = New cProvider
   Set oBinding = New BindingCollection
   ' Associo alla collection di Binding
   ' il DataSource Provider
   Set oBinding.DataSource = Provider
   oBinding.Add Text1, "Text", "Nome"
   oBinding.Add Text2, "Text", "Cognome"
End Sub
```
In pratica è necessario creare una variabile, in questo caso *Provider*, che referenzi la classe appena creata ed un oggetto *BindingCollection,* che è disponibile nel progetto tramite la libreria Microsoft Data Binding Collection, che si preoccupa di mantenere l'allineamento tra il recordset contenuto nella classe ed i controlli.

Dato un database con all'interno un report, questo può essere utilizzato in Visual BASIC.
```
Private Sub Form_Load ()
        Dim appl As Access.Application
        Set appl = New Access.Application
        appl.OpenCurrentDatabase ("C:\pippo.mdb")
        appl.DoCmd:OpenReport "_reportpippo"
        appl.CloseCurrentDatabase
End Sub
```

CONNESSIONE AD UN DB MYSQL CON VISUAL BASIC

INTRODUZIONE

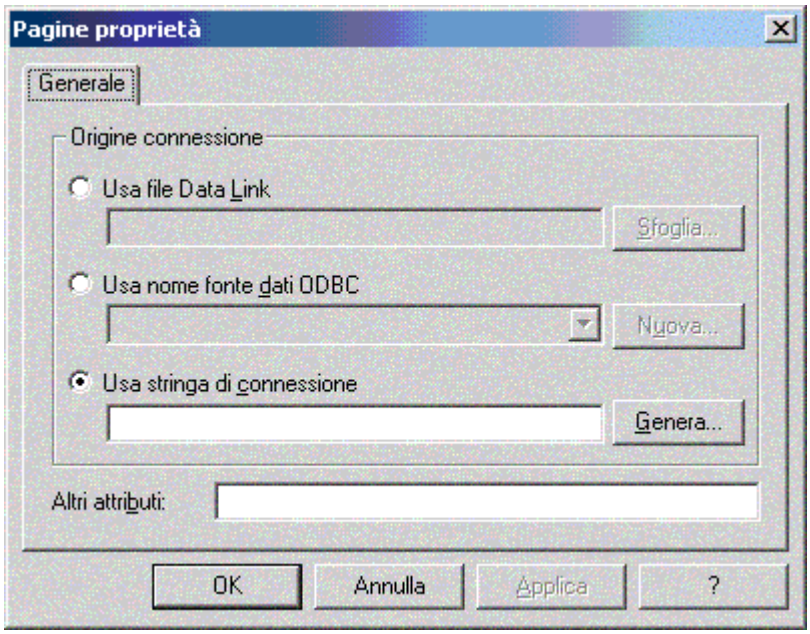
A differenza di Access, che permette di lavorare in locale, in altre parole usando l'applicativo come una normale applicazione per l'ufficio, con il salvataggio dei propri file su disco, MySQL è un database server, offre in pratica una gestione multi-utente e multi-processo dei database.
Ogni utente ha a disposizione nome utente e password per accedere con i propri privilegi (lettura, scrittura) ai database disponibili. È inoltre possibile avere più accessi contemporanei e conseguentemente più processi (thread) in esecuzione allo stesso momento.
Altra differenza di MySQL rispetto ad Access è la necessità di specificare al programma l'host sul quale deve funzionare; essendo, infatti, possibile l'accesso al servizio anche se esso risiede su di un'altra macchina bisogna sempre specificare l'host dove MySQL è installato.
Come host predefinito MySQL si connette all'host "**localhost**", in pratica al PC locale, permettendo di lavorare in locale. MySQL s'installa in Windows come "servizio" ad esecuzione automatica. Se si vuole che il server non parta in automatico all'avvio, basta richiamare il menu **Start/Tutti i programmi/Strumenti di amministrazione/Servizi**. Si avrà una lista dei servizi installati sul PC, quindi selezionare MySQL e fare clic su **Avvia**.
Purtroppo MySQL non è fornito di un'interfaccia grafica e quindi ogni istruzione deve essere inserita manualmente da riga di comando. Fortunatamente esistono software open source, alcuni di terze parti altri sviluppati direttamente da MySQL AB (l'azienda proprietaria di MySQL), che forniscono l'interfaccia necessaria a gestire più comodamente e velocemente i database, per esempio, **MySQL Control Center** permette la creazione e la manutenzione dei database MySQL.
Per connettere un'applicazione Visual BASIC ad un database creato con MySQL bisogna aver installato i driver ODBC appropriati. I driver ODBC necessari alla connessione di un database MySQL sono disponibili gratuitamente sul sito ufficiale MySQL all'indirizzo: http://www.MySQL.com/downloads/api-myodbc-3.51.html, terminata la procedura d'installazione dei driver si può procedere allo sviluppo del progetto.
Quando si parla di database, si può fare confusione. Spesso si scambia il formato, ad esempio *.MDB per Access, con il motore, che in questo caso si chiama JET. Infatti, *.MDB è un formato di dati cui si può accedere tramite il motore di Microsoft JET, anche se Access non è installato sul PC. Allo stesso modo tramite Access possiamo manipolare dati, ad esempio in formato MySQL, senza usare il JET, utilizzando invece il driver MyODBC. Nel caso di Access la confusione sorge dal fatto che l'applicazione usa il formato *.MDB anche per le proprie necessità, maschere, report, oltre che per i dati.
Altra differenza da considerare è quella tra i database server veri e propri ed i motori database generici. Un database server parte come un servizio residente in memoria e rimane in ascolto sulla porta 3306 per MySQL. Un motore database, come JET è un insieme di DLL (librerie) richiamate all'occorrenza dall'applicazione che deve far uso di dati.
Visual BASIC permette la consultazione, la modifica e l'integrazione dei database attraverso interfacce chiamate **sorgenti dati**. In pratica, quando si vuole usare i dati di un database, si deve prima creare una sorgente dati, che indica a Visual BASIC come dialogare con il database stesso. Una volta stabilita questa connessione, l'applicazione integra strumenti che permettono di ottenere risultati anche senza conoscere quasi nulla del motore di database utilizzato.
Si possono creare vari tipi di sorgenti dati, ma l'interesse è per quelle che usano **ODBC** per l'accesso ai database.

Prima però di creare una sorgente dati, è necessario un ulteriore passaggio. Lo standard ODBC prevede, per l'accesso ad uno specifico database, la creazione di un **DSN** (*Data Source Name***)**. Un Data Source è un database server indipendente, i driver sono librerie che trasformano il linguaggio generico ODBC in quello proprio del Data Source corrispondente. In pratica, si tratta di un'interfaccia di configurazione che contiene informazioni sull'archivio cui bisogna collegarsi. Infatti, i server di database possono gestire contemporaneamente molti database diversi, ed è quindi necessario indicare, oltre al tipo di motore che si vuole usare, anche esattamente a cosa si vuole avere accesso.
Il database MySQL da utilizzare deve essere esistente e deve contenere almeno una tabella.

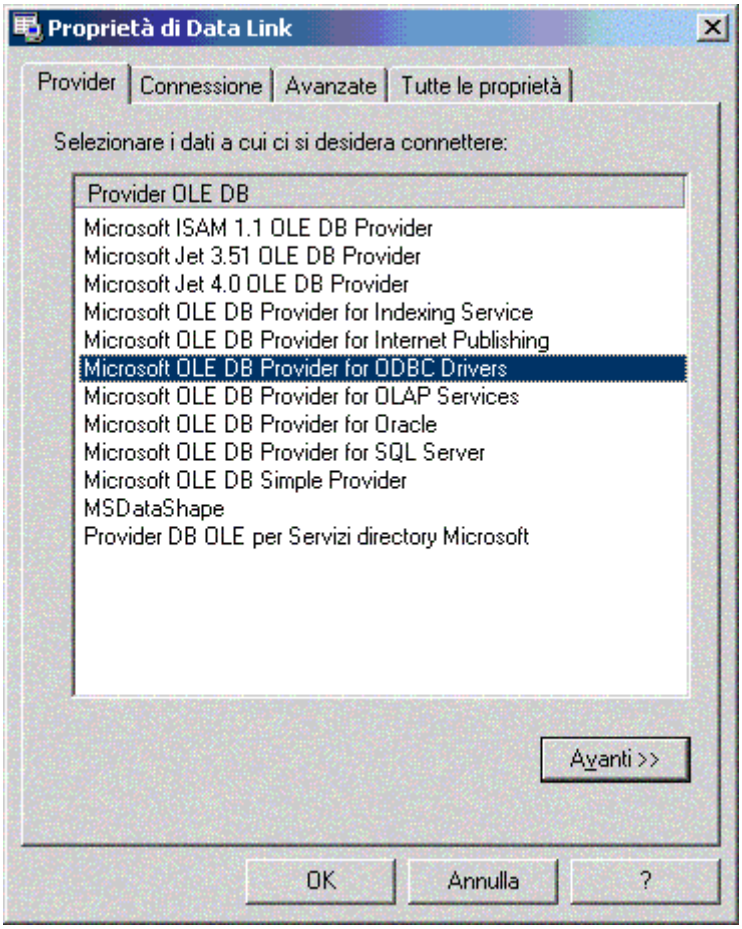
1- Selezionare dal menu **Progetto/Componenti/Microsoft ADO Data Control 6.0 (OLEDB)**.
2- Inserire l'oggetto **ADODC** dalla casella degli strumenti nel form.
3- Nelle proprietà dell'oggetto ADODC selezionare l'opzione **ConnectionString**.

Cliccare sull'icona a destra, quindi:

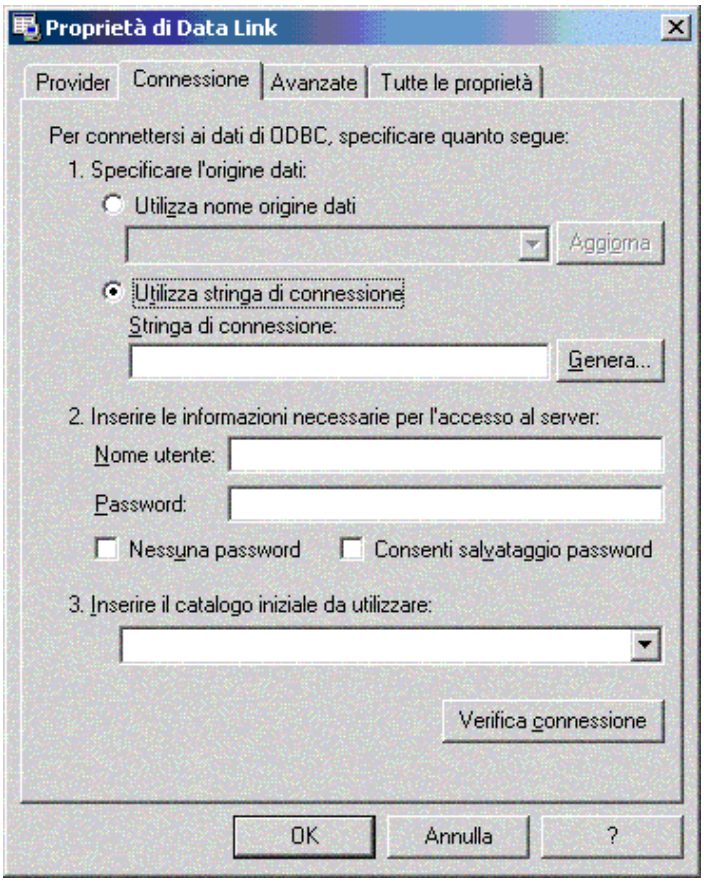
A. Selezionare **Usa stringa di connessione** e cliccare sul pulsante **Genera…**



B. Nella scheda **Provider** selezionare **Microsoft OLE DB Provider for ODBC Drivers**.

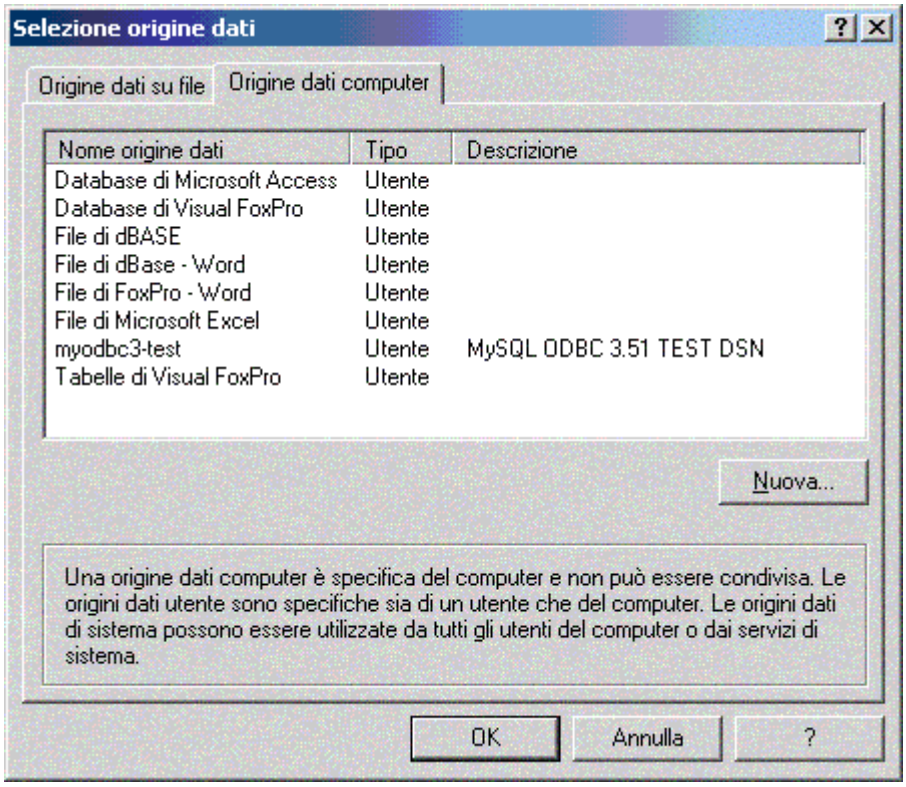


C. Nella scheda **Connessione** selezionare **Utilizza stringa di connessione**, quindi cliccare sul pulsante **Genera…**



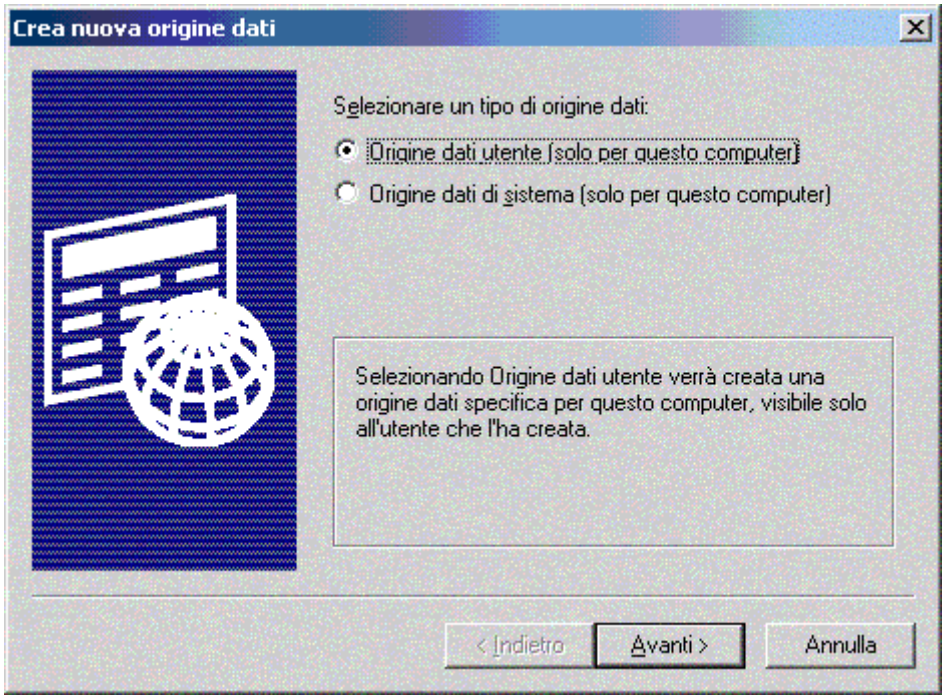
D. Selezionare la scheda **Origine dati computer** se si vuole che la sorgente dati sia

disponibile solo per la macchina corrente, altrimenti selezionare la scheda **Origine dati su file**.

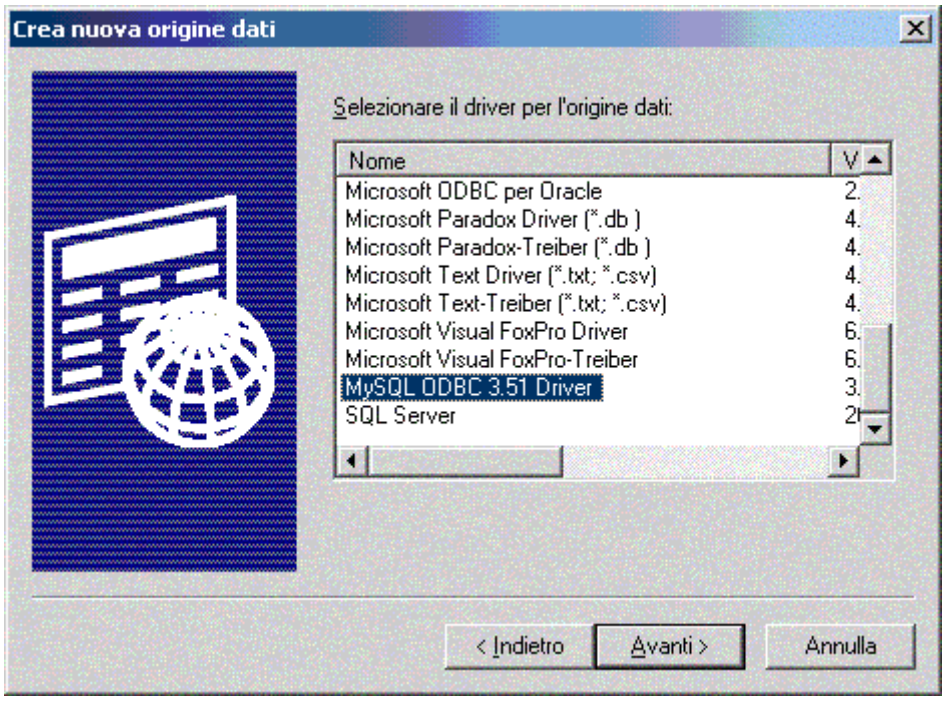


In Windows, i DSN sono gestiti tramite **Start/Tutti i programmi/Strumenti di amministrazione/Amministrazione origine dati ODBC**.
Esistono tre tipi di **DSN** (*Data Source Name*): **utente**, **di sistema** e **su file**.
Il DSN utente ha valore solo per l'utente che lo crea, quello di sistema ha valore per tutti gli utenti del PC. Per il resto la procedura da seguire è identica.
Assicurarsi che il Server MySQL sia attivo, fare clic sul pulsante **Aggiungi...** selezionare un driver tra quelli disponibili sul sistema; scegliere MySql ODBC 3.51 Driver e premere sul pulsante **Fine**.

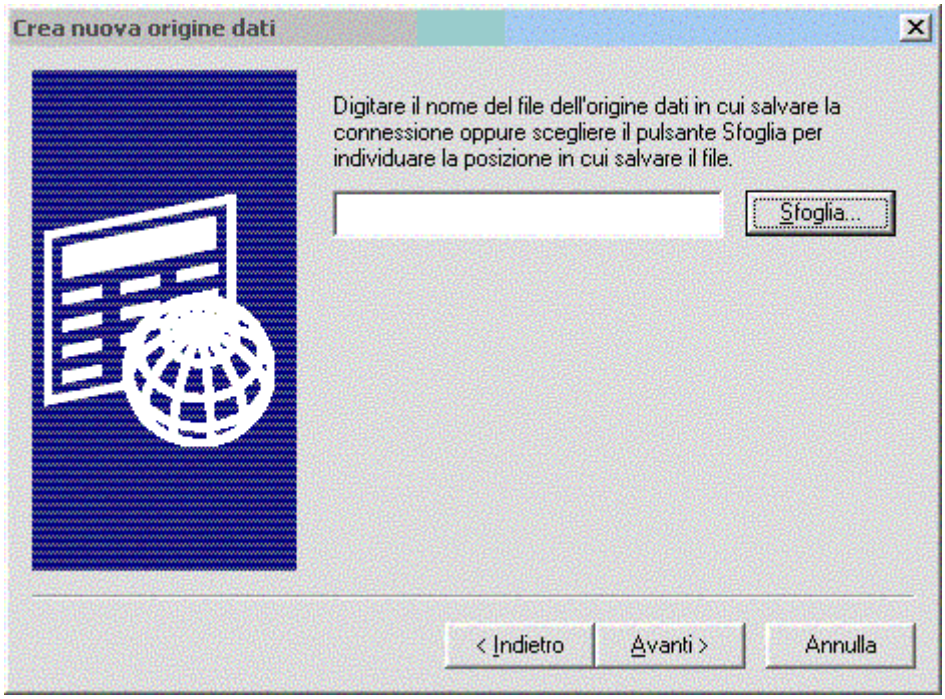
E. In entrambi i casi cliccare sul pulsante **Nuova**
F. Se si è scelto Origine dati computer selezionare **Origine dati utente (solo per questo computer)**, se si vuole che i dati siano disponibili solo all'utente che crea l'origine dati oppure la voce **Origine dati di sistema (solo per questo computer)**, se l'origine dati vuole essere resa disponibile per chiunque abbia accesso al PC, altrimenti passare al punto G.



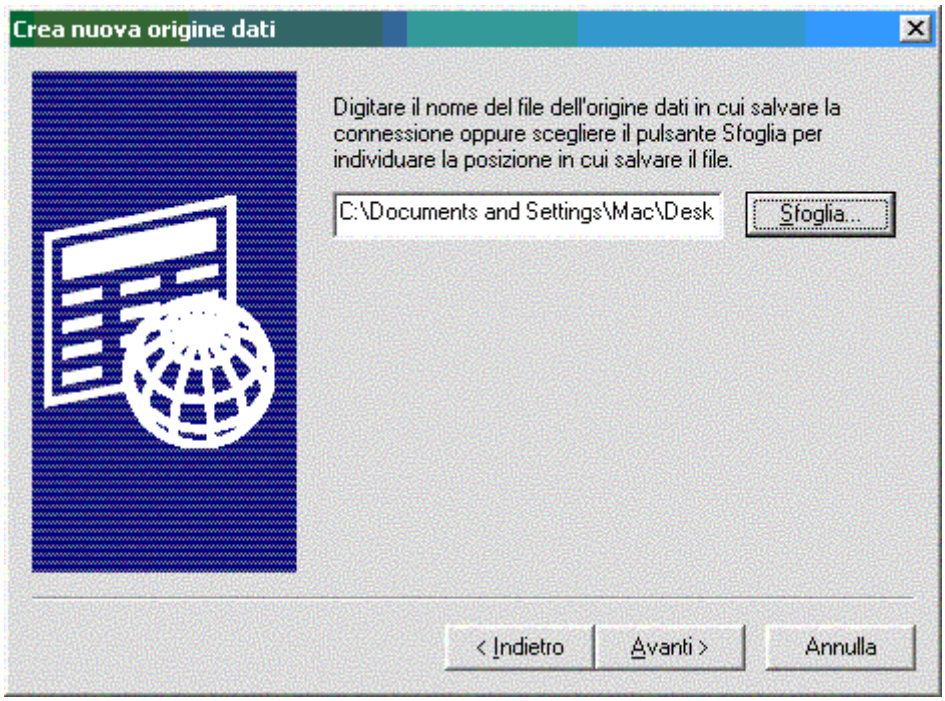
G. Cliccare su avanti e selezionare il driver **MySQL ODBC 3.51 Driver**.



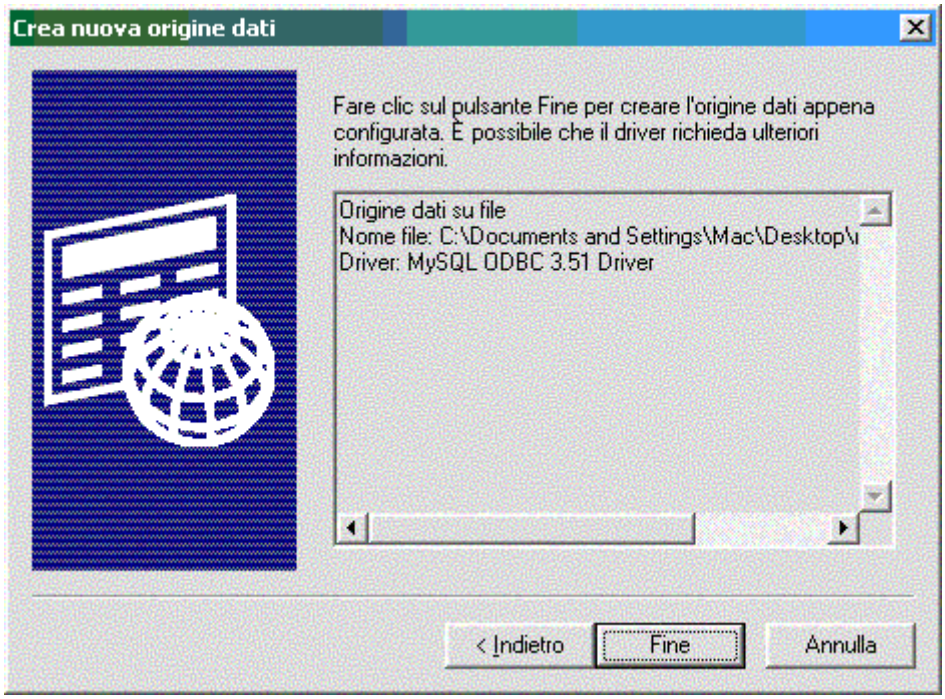
H. Se si è scelto Origine dati su file selezionare il percorso dove l'origine dati deve essere salvata.



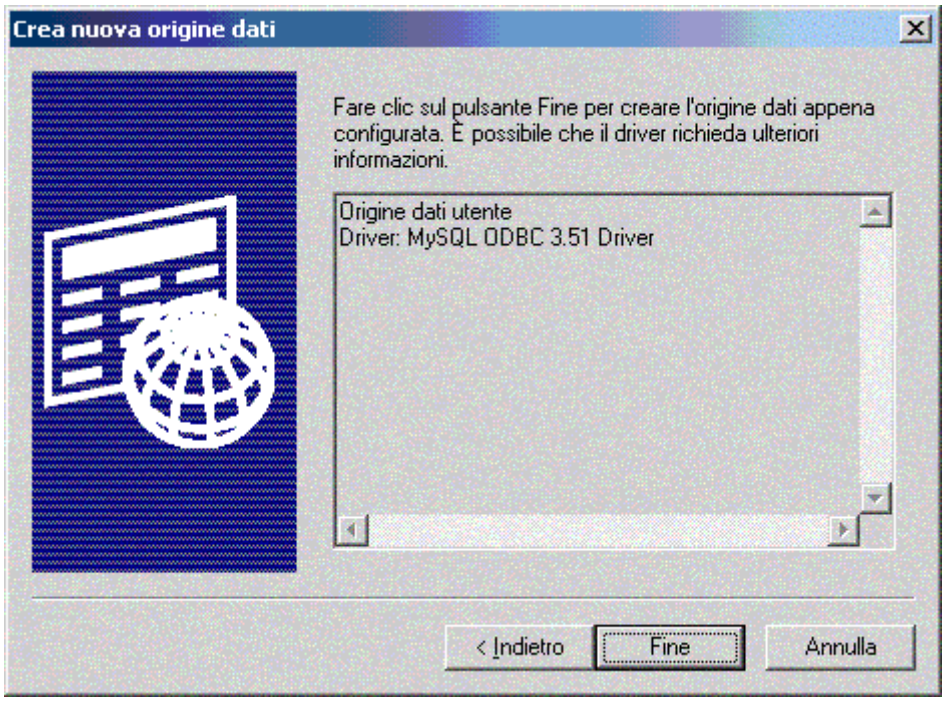
I. Dopo aver dato un nome all'origine dati cliccare su **Avanti >** e salvare.
J. Cliccare su **Avanti >**.



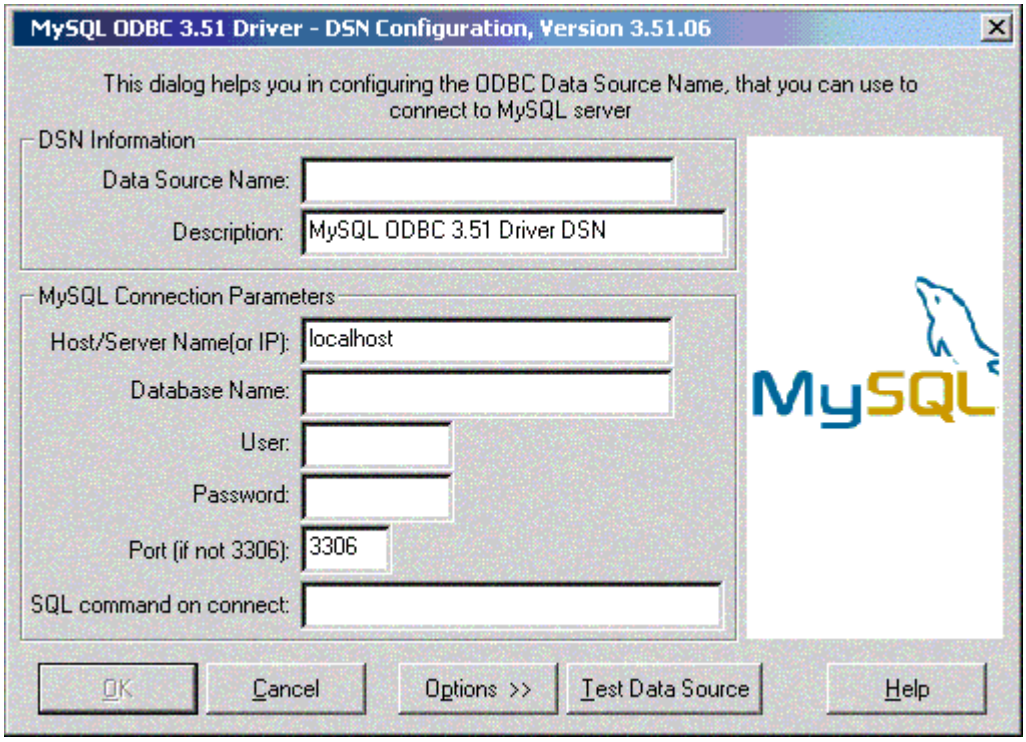
K. Cliccare su **Fine**.



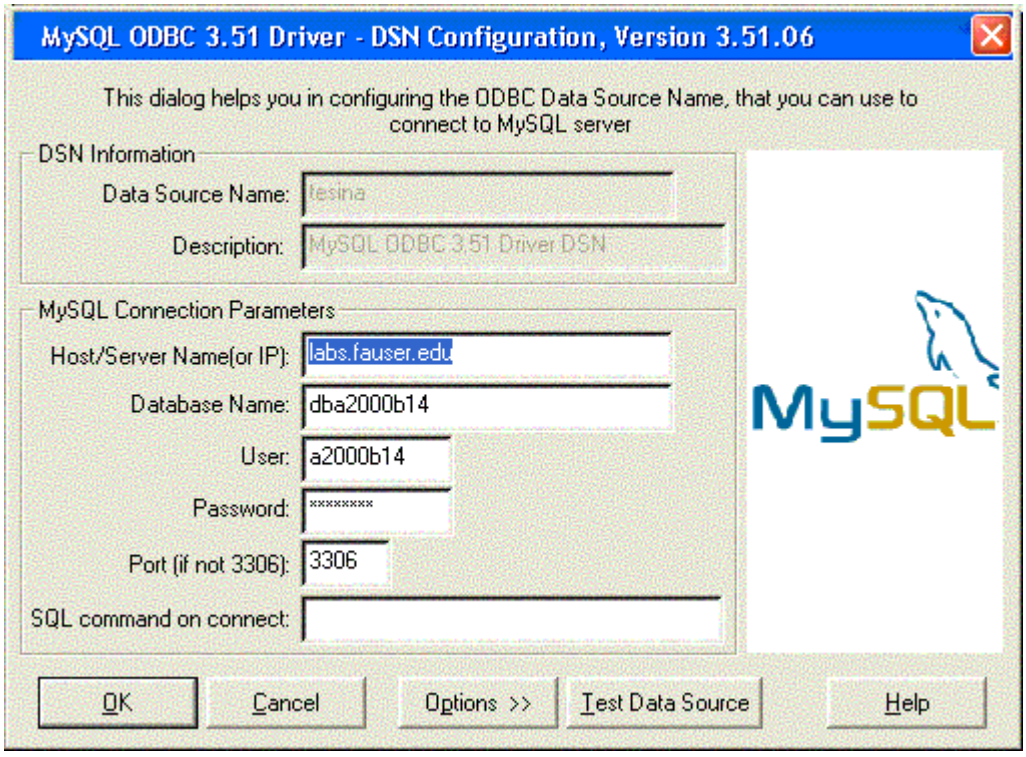
L. Se invece si è scelto Dati origine computer nella finestra che appare cliccare su **Fine**.



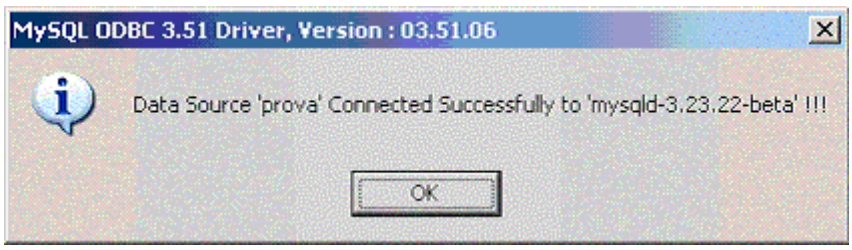
M. Nella finestra che appare specificare il nome che si vuole dare all'origine dati, il DSN, ed una descrizione nel campo **Description** (opzionale). Nei parametri di connessione si deve specificare il **nome o l'indirizzo IP del Server** (localhost se gira sulla PC locale), il **nome del Database** che deve essere lo stesso usato nella creazione, **l'utente** (root) e l'eventuale **password**, la **porta** 3306 è quella di default. Il pulsante **Options** permette di specificare molti parametri per la connessione, ma di solito il DSN funziona egregiamente con i valori di default.



Il server dell'Istituto "Giacomo Fauser" si trova all'indirizzo seguente.



N. Verificare la connessione cliccando sul pulsante **Test Data Source**, quindi cliccare sul pulsante **OK**. È visualizzato il seguente messaggio.



O. cliccare ancora su **OK** poi sul pulsante **Verifica connessione** e di nuovo sul pulsante **OK**.
P. Cliccare ancora sul pulsante **OK**.

La connessione è ora attiva.

Inserire nel progetto **Progetto/Componenti/Microsoft DataGrid Control 6.0 (OLEDB)**.

4- Dopo aver inserito l'oggetto cercare **DataSource** nella finestra proprietà e selezionare il nome dell'oggetto ADODC creato precedentemente.
5- Nella proprietà **RecordSource** dell'oggetto ADODC cliccare sull'icona a destra.
6- Nella finestra proprietà che appare selezionare come tipo command **2 – adCmdTable**.
7- Selezionare la tabella del database che si vuole visualizzare e cliccare su **OK**.

CONNESSIONE AD UN DB CON C++

```
#include <iostream.h>
#include <stdlib.h>
#import "C:\\Programmi\\File comuni\\System\\ado\\msado15.dll"\\no_namespace rename
("EOF","EndOfFile")
//Librerie necessarie per utilizzare gli oggetti ADO
int main(void)
{ ::CoInitialize(NULL);                                    //Inizializza l'ambiente COM
  _ConnectionPtr PuntConn;
  PuntConn.CreateInstance (__uuidof(Connection));//Crea un oggetto di classe Connection
  try {
     PuntConn->Open ("C:\\Borlandc\\Output\\db11997.mdb","","",NULL);
                                                                  //Attiva la connessione
     }
  catch (_com_error &e)
     {
      cout<<"Connessione al database non avvenuta";
      return (-1);
     }
  _CommandPtr PuntCmd;
  PuntCmd.CreateInstance (__uuidof(Command)); //Crea un oggetto di classe Command
  PuntCmd->ActiveConnection=PuntConn;
  PuntCmd->CommandText="SELECT * FROM exallievi";  //Imposta la query
  _RecordsetPtr PuntRs;
  PuntRs.CreateInstance (__uuidof(Recordset));   //Crea un oggetto di classe Recordset
  PuntRs->PutRefSource(PuntCmd);                   //Memorizza il risultato della query
  _variant_t VNull;
  VNull.vt=VT_ERROR;
  VNull.scode=DISP_E_PARAMNOTFOUND;
  PuntRs->Open(VNull,VNull,adOpenDynamic,adLockOptimistic,adCmdUnknown);
  while (!PuntRs->EndOfFile)                //Preleva i dati dall'oggetto RecordSet
       { cout<<(char *)_bstr_t(PuntRs->Fields->GetItem("Cognome")->Value<<",";
         PuntRs->MoveNext();
        }
  _bstr_t StrQuery="INSERT INTO exallievi (Cognome) VALUES ('Rossi')"
 //Inserimento record
  PuntConn->Execute (StrQuery,NULL,adExecuteNoRecords);
  _bstr_t StrQuery="UPDATE exallievi SET Cognome='Rossi' WHERE Nome='Luca'";
//Modifica record
  PuntRs->Close();
  PuntConn->Close();
  CoUninitialize();
  system("PAUSE");
  return (0);
}
```

VBA (VISUAL BASIC FOR APPLICATIONS)

INTRODUZIONE

L'insieme delle applicazioni contenute in Microsoft Office (Visio e AutoDESK) dispone del linguaggio macro di programmazione VBA che permette di:

- automatizzare le procedure e i comandi ripetitivi;
- aggiungere nuove funzionalità;
- riprodurre, con istruzioni, tutte le azioni che altrimenti gli utenti di Office svolgerebbero manualmente;
- è un IDE condiviso tra tutte le applicazioni Office;
- è un sotto insieme di Visual BASIC;
- supporta la tecnologia **COM** ActiveX.

Possiamo classificare le soluzioni VBA per Office nei seguenti tipi principali.

Documento singolo

Inserisce nuove funzionalità e/o automatizza quelle presenti in un documento Office. Il codice sorgente è memorizzato nel file che contiene il documento.

Documento multiplo

Richiama le funzionalità di più di un'applicazione di Office. Il codice sorgente è memorizzato nel file del documento che svolge l'elaborazione principale.

Modello

Gli utenti possono usare le impostazioni, il formato e i nuovi comandi già predisposti. Il codice sorgente è memorizzato nel file modello associato al documento.

In tutti questi casi s'inserisce codice sorgente VBA all'interno dei documenti Office. Il codice sorgente non è compilato, ma interpretato dall'interprete inserito in Office. Per questa caratteristica e per i macro comandi, il VBA è indicato come un **linguaggio di script**.

All'apertura di un documento Office, ci possiamo accorgere che il documento contiene codice VBA, perché l'applicazione visualizza: **Attiva/Disattiva macro**.

Applicazione autonoma

Esterna e indipendente, utilizzano le funzionalità specifiche di Office per creare documenti professionali senza richiedere agli utenti alcuna conoscenza specifica. Il codice sorgente è memorizzato nel file eseguibile o in una DLL.

Un progetto VBA è formato da un insieme di moduli.

Microsoft Word (Excel) Oggetti: ogni applicazione espone una libreria di classi (componenti COM) organizzata in modo gerarchico in un modello a oggetti, di cui il programmatore può creare nuove istanze per richiamare proprietà, metodi ed eventi.

UserForm (Form del Visual BASIC), Moduli, Moduli di classe

Riferimenti: definisce il puntatore ad una libreria di componenti COM di altre applicazioni, oppure ad altri documenti o modelli della stessa applicazione

Le routine di VBA possono essere:

- sub o function;
- metodi ed eventi di oggetti del modello di Office oppure di una classe creata dal programmatore;
- macro.

Una **macro** è una routine (*Public Sub* per default) il cui codice sorgente riproduce una serie di operazioni, svolte da un utente su un documento Office. Terminata la registrazione, il codice sorgente, che riproduce tutte le azioni svolte dall'utente, è memorizzato dall'IDE VBA in una routine pubblica inserita in un modulo standard dell'applicazione associata al documento (è quindi a disposizione del programmatore per eventuali modifiche e/o aggiunte).

Per registrare una semplice macro.

Strumenti/Macro/Registra nuova macro…
Nella casella Nome macro digitare il nome desiderato per la macro. Nella casella Memorizza la macro in fare clic sul modello o documento in cui si desidera memorizzare la macro. Nella casella Descrizione digitare una descrizione della macro.
Avviare, registrare, interrompere la registrazione.
Copiare una macro: **Strumenti/Macro/Macro…(ALT+F8)/Libreria…**
Eseguire una macro: **Strumenti/Macro/Macro…(ALT+F8)/Esegui**
Assegnare una macro ad un pulsante particolare su una barra degli strumenti.
Fare clic su Barre degli strumenti, quindi scegliere la scheda Comandi. Nella casella Comandi fare clic sulla macro in fase di registrazione e trascinarla sulla barra degli strumenti o sul menu a cui si desidera assegnarla. Fare clic su Chiudi per iniziare la registrazione della macro.
Per assegnare la macro a tasti di scelta rapida, fare clic su Tastiera. Nella casella Comandi fare clic sulla macro in fase di registrazione. Nella casella Nuova combinazione digitare la sequenza dei tasti desiderata, quindi fare clic su Assegna. Fare clic su Chiudi per iniziare la registrazione della macro.
Per eliminare una macro dalla barra degli strumenti, fare clic su:
Aggiungi/Rimuovi pulsanti/Formattazione/Reimposta barra degli strumenti
Avviare l'editor VBA: **Strumenti/Macro/Visual Basic Editor (ALT+F11)**
Dispone di tre modalità.
1. Progettazione: per la modifica del codice sorgente.
2. Esecuzione: del codice.
3. Interruzione: per il debugging del codice.

MODELLI A OGGETTI

Per gestire gli oggetti il programmatore deve:
- includere un riferimento alla libreria di classi di Office;
- dichiarare e creare nuovi oggetti tra i componenti della gerarchia;
- sviluppare il codice sorgente dell'applicazione VBA richiamando proprietà , metodi ed eventi degli oggetti istanziati.

Per default, se avviamo un progetto VBA a documento singolo, l'editor inserisce in modo automatico il riferimento al modello a oggetti dell'applicazione Office in esecuzione. Per esempio, in Word nella finestra di dialogo **Strumenti/Riferimenti** compare Microsoft Office 10.0 Object Library. Al contrario, in una soluzione a documento multiplo, è necessario inserire i riferimenti a mano per i modelli di tutte le applicazioni secondarie richiamate.
La dichiarazione e la creazione di nuovi oggetti Office avviene in due fasi.
1. Dichiarazione di una variabile oggetto ovvero un puntatore a una classe, in fase di progettazione: *Dim nomeoggetto As nomeapplicazione_nomeclasse*
2. Creazione di un'istanza del componente
   Set nomeoggetto = New nomeapplicazione_nomeclasse

In alternativa, si possono usare le funzioni *GetObject()* e *CreateObject(),* ma producono un codice macchina meno efficiente.
Excel permette di richiamare le sue funzioni, senza inserirle in un foglio di lavoro; per esempio calcolare la media di un vettore.

```
'Progetto/Riferimenti/Visual Basic for Applications
'Progetto/Riferimenti/Microsoft Excel 10.0 Object Library
Option Explicit
Private Sub Command1_Click()
    Dim i As Integer, V(1 To 5) As Integer, media As Single
    'Per default, ogni oggetto applicazione è invisibile anche se allocato in RAM
    Dim objExcel As Excel.Application
    'Apro il collegamento OLE con l'oggetto excel
    Set objExcel = New Excel.Application
    'Rendo visibile Excel
```

```
    objExcel.Visible = False
    For i = 1 To 5
       V(i) = Int((6 * Rnd) + 1)
    Next i
    media = objExcel.WorksheetFunction.Average(V(1), V(2), V(3), V(4), V(5))
    Print: Print: Print media
    'Chiudo Excel
    objExcel.Application.Quit
    'Dealloca il riferimento a Excel per liberare spazio in RAM
    Set objExcel = Nothing
End Sub
```

Word

```
'Progetto/Riferimenti/Visual Basic for Applications
'Progetto/Riferimenti/Microsoft Word 10.0 Object Library
Option Explicit
Private Sub Command1_Click()
    Dim objWord As Word.Application
    Dim wDocument As Word.Document
    'Apro il collegamento ole con l'oggetto Word
    Set objWord = New Word.Application
    'Rendo visibile Word
    objWord.Visible = True
    'Crea un nuovo documento basato sul modello normal.dot
    Set wDocument = objWord.Documents.Add
    'Rendo il nuovo documento il documento attivo di Word.
    wDocument.Activate
    'Scrivo nel documento attivo il testo del controllo di prova
    objWord.Selection.TypeText Text1.Text
    'Seleziono tutto il testo del documento
    objWord.Selection.WholeStory
    'Chiudo il documento word senza salvare le modifiche
    wDocument.Close SaveChanges:=wdDoNotSaveChanges
    'Elimino il riferimento al documento
    Set wDocument = Nothing
    'Chiudo word
    objWord.Application.Quit
    'Elimino il riferimento all'oggetto Word
    Set objWord = Nothing
End Sub
```

CONTROLLI GRAFICI

La tecnica per fornire ad un documento un'interfaccia grafica è quella d'inserire in un documento dei controlli ActiveX predefiniti o creati dal programmatore.

Visualizza/Barra degli strumenti/Strumenti di controllo

Disegnare i controlli desiderati nel documento

Impostare le proprietà degli oggetti

Nella gerarchia delle classi, un controllo in un documento rappresenta un nuovo oggetto annidato all'interno dell'istanza del documento. I controlli permettono di eseguire in modo interattivo, il codice VBA mentre l'utente sta lavorando e quindi vedendo il documento. I controlli sono inseriti nell'area di stampa attiva, per questo sono visibili nella stampa del documento, devono essere nascosti (al di fuori dell'area di stampa, in un altro documento) per non essere stampati.

Per esempio, calcolare con Excel la funzione seno (OptionButton1) oppure coseno (OptionButton2) dell'angolo in radianti memorizzato nella cella A2, quando l'utente fa clic

su Calcola (CommandButton1).

```
Private Sub CommandButton1_Click()
   Range("A2").Select
   If OptionButton1.Value = True Then
      ActiveCell.FormulaR1C1 = "=Sin(R2C1)"
   End If
   If OptionButton2.Value = True Then
      ActiveCell.FormulaR1C1 = "=Cos(R2C1)"
   End If
End Sub
```

APPLICAZIONI EXCEL

Le attività di un programmatore Excel riguardano.

1. La gestione di cartelle e fogli elettronici.
2. L'elaborazione numerica di dati con l'uso delle funzioni di libreria.
3. La formattazione e la presentazione dei risultati.

Un documento Excel è una cartella che contiene un insieme di fogli elettronici. Nel modello ad oggetti, l'oggetto applicazione (Application) Excel include un insieme di oggetti (Collection) della Worksheet. Se da una cartella Excel, avviamo un progetto VBA associato, possiamo gestire gli oggetti esposti e quindi disponibili nella gerarchia dell'applicazione. Excel crea automaticamente dei moduli di classe relativi agli oggetti Worlbook e Worksheet inserendoli nella finestra Gestione progetti.

Per esempio, per creare una nuova cartella.

```
Public Sub nuovacartella()
   Dim nc As Workbook
   Set nc = Workbooks.Add
   nc.SaveAs "c:\appo"
End Sub
```

Il ciclo di vita di una cartella è regolato da eventi: *Open()* avvio, *Deactivate(), Activate() e BeforeClose()*chiusura del documento.

```
Private Sub Workbook_Open()
        MsgBox "Benvenuti in Excel", vbOKOnly, "Fattura"
End Sub
```

Per esempio, data una cartella contenente tre fogli elettronici, aggiungere un nuovo foglio elettronico, alla cartella corrente e rinominare tutti i fogli in Fattura (con I da 1 a 4).

```
Public Sub inseriscifoglio()
   Dim i As Integer
   Dim nf As Worksheet
   Set nf = Worksheets.Add
   For i = 1 To 4
      Worksheets(i).Name = "Fattura" & Str(i)
   Next i
   Set nf = Nothing
End Sub
```

Per formattare un oggetto Range si usano le proprietà: NumberFormat per il formato dei numeri, Font per i caratteri. Per esempio, calcolare l'equazione di 1° grado.

```
Public Sub calcola()
   Dim foglio As Worksheet
   Set foglio = Application.Worksheets("Foglio1")
   If foglio.Cells(2, 1).Value <> 0 Then
      foglio.Cells(2, 3).Value = -foglio.Cells(2, 2).Value / foglio.Cells(2, 1).Value
      foglio.Range("a2:c2").NumberFormat = "#,##0.00_ ;-#,##0.00 "
   Else
      foglio.Cells(2, 3).Value = "Errore"
```

```
    End If
    Set foglio = Nothing
End Sub
```
Le formule possono essere:
1. inserite in una cella del foglio elettronico;
2. richiamate direttamente come oggetto della classe WorksheetFunction dell'istanza Application.

Per esempio, in C2 calcolare un importo totale scontato.
```
Public Sub calcola()
    Dim foglio As Worksheet
    Set foglio = Application.Worksheets("Foglio1")
    foglio.Range("c2").Formula = "=a2*b2"
   'foglio.Range("c2").FormulaR1C1 = "=R2C1*R2C2" stessa formula
    Set foglio = Nothing
End Sub
```
Il nome di una cella (range) è un oggetto dell'insieme Names di un Worksheet.
```
Public Sub calcola()
    Dim foglio As Worksheet
    Set foglio = Application.Worksheets("Foglio1")
    foglio.Names.Add Name:="Importo", RefersToR1C1:="=Foglio1!R2C1"
    foglio.Names.Add Name:="Percentuale", RefersToR1C1:="=Foglio1!R2C2"
    foglio.Range("c2").FormulaR1C1 = "=Importo*Percentuale"
    Set foglio = Nothing
End Sub
```

APPLICAZIONI WORD

La tecnica per distribuire codice VBA in Word è quella di progettare un modello (.dot DOcument Template) salvato nella cartella di default di Word per i modelli.
C:\Documents and Settings\%UserProfile%\Dati applicazioni\Microsoft\Modelli
Il modello sarà sempre disponibile all'utente all'apertura di un nuovo documento. Quando un utente apre un modello personalizzato, Word duplica il contenuto del modello nel nuovo documento, rendendo disponibile il codice VBA. Tutte le correzioni apportate al codice del modello saranno disponibili in tutti i documenti basati sul modello stesso.
Il testo in un documento è organizzato, nell'ordine, in (non è gerarchica, ma libera):
- sezioni (Sections) è una parte del testo con un formato di pagina differente;
- paragrafi (Paragraphs) è un insieme di frasi terminato da INVIO;
- frasi (Sentences) è un insieme di parole separate tra loro da uno spazio e terminate da un punto;
- parole (Words) è un insieme di caratteri preceduti e seguiti da uno spazio e/o simboli speciali (punteggiatura);
- caratteri (Characters) è la singola digitazione dell'utente sulla tastiera.

Per esempio, l'insieme Sentences di un'istanza Document contiene tutte le frasi dell'intero documento e non le frasi annidate in un determinato paragrafo.
L'editor VBA semplifica la gestione degli oggetti di Word, creando automaticamente nella finestra progetto dei moduli software associati ai seguenti elementi.
1. Normal ThisDocument: il modello Normal.dot.
2. Project ThisDocument: il documento attivo.

Per esempio, contare il numero di paragrafi, frasi, parole e caratteri nel documento attivo.
```
Public Sub calcola()
    Dim np As Integer, nf As Integer, nw As Integer, nc As Integer
    Dim d As Document
    Set d = Application.ActiveDocument
    np = d.Paragraphs.Count
    nf = d.Sentences.Count
```

```
    nw = d.Words.Count
    nc = d.Characters.Count
    MsgBox "Il testo contiene " & Str(np) & " paragrafi; " & Str(nf) & " frasi; " & Str(nw) & " parole; " & Str(nc) & " caratteri. "
    Set d = Nothing
End Sub
```

Per esempio, salvare il documento attivo come pagina web.

```
Public Sub calcola()
    Dim d As Document
    Set d = Application.ActiveDocument
    d.SaveAs FileName:="C:\appo.htm", FileFormat:=wdFormatHTML
    Set d = Nothing
End Sub
```

L'insieme Dialogs contiene tutti i dialoghi predefiniti di un'applicazione Office; per esempio, inserire un nuovo documento e salvarlo con la finestra di dialogo di Windows.

```
Public Sub calcola()
    Dim d As Document
    Set d = Application.Documents.Add
    Application.Dialogs(wdDialogFileSaveAs).Show
    Set d = Nothing
End Sub
```

Una generica parte di un documento restituisce un oggetto di tipo Range, mentre la parte attiva selezionata è un'istanza della classe Selection. Per esempio, dopo aver selezionato come parte attiva l'intero documento, applica il controllo ortografico.

```
Public Sub calcola()
    Dim p As Range
    Set p = Application.ActiveDocument.Content
    p.CheckSpelling
    Set p = Nothing
End Sub
```

Per inserire e formattare il testo in un documento, il programmatore Word può seguire successivamente questi passi.

1. Impostare un Range (sezione, paragrafo, frase) come quello attivo nel documento.
2. Usare i metodi InsertAfter o InsertBefore dell'oggetto Range per inserire rispettivamente, dopo oppure prima del punto attivo del nuovo testo.
3. Formattare il nuovo testo con la proprietà Font e Style.

Per esempio, creare un nuovo documento, inserire due nuovi paragrafi formattandoli con stili predefiniti differenti

```
Public Sub calcola()
    Dim t As Document
    Dim p As Range
    Set t = Application.Documents.Add
    Set p = t.Paragraphs(1).Range                    'Primo paragrafo
    p.InsertAfter "Titolo del documento"
    p.InsertParagraphAfter
    p.Style = t.Styles("Titolo 1")
    Set p = t.Paragraphs(2).Range                    'Secondo paragrafo
    p.InsertAfter "Contenuto del documento"
    p.InsertParagraphAfter
    p.Style = t.Styles("Normale")
    Set t = Nothing
    Set p = Nothing
End Sub
```

SCAMBIO DATI TRA APPLICAZIONI

Si usano gli Appunti di Windows con i metodi Copy (CopyPicture) e Paste dell'oggetto Range.

Per esempio, abbiamo una tabella Excel e la vogliamo copiare in Word.

```
Public Sub copia()
    'Strumenti/Riferimenti/Microsoft Word 10.0 Office Library
    'Aprire Word con un documento
    Dim p As Word.Range
    Dim f As Excel.Worksheet
    Set f = Worksheets("Foglio1")
    f.Range("A1:B5").Copy
    Set p = Word.Application.ActiveDocument.Paragraphs(1).Range
    p.Paste
End Sub
```

VBSCRIPT

INTRODUZIONE

HTML è il linguaggio di formattazione alla base delle pagine web ed è progettato per i seguenti obiettivi.

1. Formattare le pagine web in colonne, con immagini e titoli appropriati.
2. Consentire l'integrazione di ulteriori programmi per servizi Internet, come documenti ActiveX e Java.

HTML è un linguaggio interpretato: il codice HTML non è compilato in un eseguibile, come un'applicazione Visual BASIC.
In Microsoft Visual BASIC Scripting Edition, il linguaggio di programmazione della famiglia Visual BASIC, sono disponibili funzioni per includere script in un'ampia gamma di ambienti diversi, tra cui script client web e script server web.

PROGRAMMAZIONE DI SISTEMA

I compiti di amministrazione che si possono fare con uno script sono: configurare stampanti, criteri di protezione, modificare l'interfaccia di sistema.
Esiste un'area grigia fra i compiti che richiuedono la scrittura di un'applicazione e quelli che si possono fare a mano: è il territorio degli script, applicazioni piccole e veloci che automatizzano un compito ripetitivo.
All'inizio Windows usava i batch file (*.BAT, *.CMD), oggi esiste un motore di scripting di ottime prestazioni e un debugger. La parola script va tradotta con manoscritto o copione. Windows mette a disposizione i servizi di sistema attraverso un insieme di librerie a collegamento dinamico DLL o attraverso oggetti COM.
Uno script è un file di testo semplice e agile progettato dall'amministratore di sistema, questo lo distingue da un'applicazione. Non ha un ambiente di sviluppo integrato.
Il codice VBScript si può usare in VBA, ma non è vero il contrario.

```
' Questo esempio elenca tutte le variabili di sistema del computer.
L_Welcome_MsgBox_Message_Text = "Questo script elencherà tutte le variabili di
sistema del computer."
L_Welcome_MsgBox_Title_Text = "Esempio di Windows Scripting Host"
Call Welcome()
' * Esempio variabili di sistema (Environment)
CRLF = Chr(13) & Chr(10)
Dim WSHShell
Set WSHShell = WScript.CreateObject("WScript.Shell")
Sub show_env(strText)
   MsgBox strText, vbInformation, L_Welcome_MsgBox_Title_Text
End Sub
intIndex = 0
strText = ""
intNumEnv = 0
MAX_ENV = 20
For Each strEnv In WshShell.Environment("PROCESS")
   intIndex = intIndex + 1
   strText = strText & CRLF & Right("    " & intIndex, 4) & " " & strEnv
   intNumEnv = intNumEnv + 1
   If intNumEnv >= MAX_ENV Then
      Call show_env(strText)
      strText = ""
      intNumEnv = 0
```

```
    End If
Next
If intNumEnv >= 1 Then Call show_env(strText)
Sub Welcome()
    Dim intDoIt
    intDoIt =  MsgBox(L_Welcome_MsgBox_Message_Text, vbOKCancel + vbInformation,
L_Welcome_MsgBox_Title_Text )
    If intDoIt = vbCancel Then
        WScript.Quit
    End If
End Sub
```

Aggiunta di codice VBScript a una pagina HTML

VBScript rappresenta un modo facile per rendere interattive le pagine web, che diversamente sarebbero statiche. Con VBScript si scrive del codice e si collauda, ma non esiste il compilatore perché il codice è incorporato in una pagina web ed è tradotto riga per riga (interpretato) quando è visualizzata la pagina stessa, questo può causare un rallentamento, ma la ragione è che può essere utilizzato da qualsiasi piattaforma con la "macchina virtuale" VBScript. La traduzione del codice VBScript incorporato nelle pagine web è una funzione del browser oppure è disponibile come plug-in. VBScript comunica con applicazioni host tramite la tecnologia ActiveX Scripting, grazie alla quale nei browser e in altre applicazioni host non è necessario scrivere codice integrativo speciale per ciascun componente script. Questa tecnologia consente ad un host di compilare script, di ottenere e richiamare punti d'ingresso e di gestire lo spazio dei nomi disponibile per lo sviluppatore.
Una routine per la verifica di una data appare nel modo seguente.

```
<HTML>
<HEAD>
<TITLE>Immissione ordine</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!--
  Function CanDeliver(Dt)
    CanDeliver = (CDate(Dt) - Now()) > 2
  End Function
-->
</SCRIPT>
</HEAD>
<BODY>
```

Il codice VBScript è inserito all'interno dei tag <SCRIPT> iniziali e finali. L'attributo LANGUAGE indica il linguaggio di script utilizzato. Dato che i browser supportano anche altri linguaggi di script, è necessario indicare quello utilizzato. Si noti inoltre che la funzione CanDeliver è racchiusa tra i tag di commento, ovvero <!-- e -->, in modo che il codice non sia visualizzato nei browser che non supportano il tag <SCRIPT>. L'esempio corrisponde ad una funzione generale **non collegata** a nessun controllo di form specifico. È possibile utilizzare i blocchi SCRIPT in qualsiasi punto della pagina HTML, sia nella sezione BODY che nella sezione HEAD. È tuttavia consigliabile inserire tutto il codice script di utilizzo generale nella sezione HEAD, in modo che l'intero codice sia incluso in un'unica posizione. Questa operazione consente di leggere e decifrare tutto il codice prima di essere utilizzato da una chiamata dall'interno della sezione BODY. Nel caso di form, si consiglia tuttavia di fornire codice script su una stessa riga per rispondere agli eventi di oggetti contenuti nel form. È, ad esempio, possibile incorporare codice script per rispondere alla pressione di un pulsante in un form.

```
<HTML>
<HEAD>
```

```
<TITLE>Verifica eventi dei pulsanti</TITLE>
</HEAD>
<BODY>
<FORM NAME="Form1">
  <INPUT TYPE="Button" NAME="Button1" VALUE="Click">
  <SCRIPT FOR="Button1" EVENT="onClick" LANGUAGE="VBScript">
    MsgBox "Pulsante premuto"
  </SCRIPT>
</FORM>
</BODY>
</HTML>
```

Quasi tutto il codice sarà visualizzato nelle routine Sub o Function e sarà richiamato solo quando il codice scritto ne determinerà l'esecuzione. È tuttavia possibile scrivere codice VBScript esternamente alle routine, ma sempre all'interno di un blocco SCRIPT. Questo codice è eseguito una sola volta, quando la pagina HTML è caricata. In questo modo, è possibile inizializzare i dati o modificare dinamicamente l'aspetto della pagina Web quando è caricata.

Nell'esempio, la funzione Celsius consente di eseguire la conversione da Fahrenheit a Celsius. Quando la funzione è richiamata nella routine Button1_OnClick, è passata alla funzione una variabile contenente il valore dell'argomento. Il risultato del calcolo è quindi restituito alla routine che ha eseguito la chiamata e visualizzato in una finestra di messaggio.

```
<HTML>
<HEAD><TITLE>Una semplice pagina di esempio</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub Button1_OnClick
       t = InputBox ("Temperatura in gradi Fahrenheit",1)
    MsgBox ("La temperatura vale a  " & Celsius (t) & "  gradi centigradi")
End Sub
Function Celsius (d)
        Celsius = (d - 32) * 5 / 9
End Function
-->
</SCRIPT>
</HEAD>
<BODY>
<H3>Una semplice pagina di esempio</H3><HR>
<FORM><INPUT NAME="Button1" TYPE="BUTTON" VALUE="Fare clic qui"></FORM>
</BODY>
</HTML>
```

Al momento della lettura della pagina, il browser rileva i tag <SCRIPT>, individua la presenza di codice VBScript e salva il codice. Quando si fa clic sul pulsante, è stabilita la connessione tra il pulsante e il codice e la routine è eseguita. La routine Sub è definita routine di eventi. Il nome della routine è composto da due parti separate da un carattere di sottolineatura, ovvero dal nome del pulsante Button1, che deriva dall'attributo NAME del tag <INPUT>, e dal nome dell'evento OnClick. Ad ogni clic sul pulsante, è cercata ed eseguita la corrispondente routine di eventi Button1_OnClick.

Per associare codice VBScript a eventi, è possibile procedere in un altro modo. Il browser consente di aggiungere brevi sezioni di codice sulla stessa riga nel tag che definisce il controllo. Quando si fa clic sul pulsante, l'azione eseguita dal seguente tag <INPUT>, ad esempio, è equivalente all'azione eseguita dal codice dell'esempio precedente.

```
<INPUT NAME="Button1" TYPE="BUTTON"
     VALUE="Fare clic qui" OnClick='MsgBox "Prova."'>
```

La stessa chiamata alla funzione è racchiusa tra virgolette semplici mentre la stringa della funzione MsgBox è racchiusa tra virgolette doppie. È possibile utilizzare più istruzioni a condizione che siano separate dai due punti (:). È inoltre possibile scrivere un tag <SCRIPT> in modo che sia valido solo per un determinato evento di un controllo specifico.

```
<SCRIPT LANGUAGE="VBScript" EVENT="OnClick" FOR="Button1">
<!--
   MsgBox "Prova."
-->
</SCRIPT>
```

In questo caso, poiché il tag <SCRIPT> specifica l'evento e il controllo, le istruzioni Sub e End Sub non sono utilizzate.

Visual BASIC include 75 istruzioni, VBScript ne supporta solamente 21 (Call, Const, Dim, Do Loop, Erase, Exit, For Next, For Each Next, Function, If Then Else, On Error, Option Explicit, Private, Public, Random, ReDim, Rem, Select Case, Set, Sub, While Wend).
Le routine di VBScript non possono mantenere i valori delle variabili tra le chiamate, pertanto la parola chiave Static non è utilizzabile.
L'istruzione Call non può chiamare routine esterne.
VBScript non supporta i moduli di classe.

TIPI DI DATI

In VBScript è disponibile solo il tipo di dati Variant, un tipo di dati speciale che, a seconda della modalità in cui è utilizzato, può includere vari tipi d'informazioni. Il tipo di dati Variant, essendo l'unico disponibile, è il tipo di dati restituito da tutte le funzioni di VBScript. Nella forma più semplice una variabile Variant può includere informazioni numeriche o stringhe. È equivalente a un numero se utilizzata in un contesto numerico e a una stringa se utilizzata in un contesto stringa. Ciò significa che, se si lavora con dati simili a valori numerici, VBScript interpreterà tali dati come numeri e saranno eseguite le operazioni più appropriate. In modo analogo, se si lavora con dati che possono essere solo stringhe, essi saranno gestiti come dati stringa. È tuttavia possibile fare in modo che i numeri siano gestiti come stringhe racchiudendoli tra virgolette doppie (" ").
Oltre alla semplice classificazione di valore numerico o stringa, con le variabili Variant è possibile definire ulteriori distinzioni in base alla natura specifica delle informazioni numeriche. Se, ad esempio, si utilizzano informazioni numeriche che rappresentano una data o un orario insieme ad altri dati relativi alla data o all'orario, il risultato sarà sempre espresso come data o come orario. È tuttavia possibile utilizzare informazioni numeriche di vario tipo con dimensioni che vanno dai valori booleani ai numeri in virgola mobile di grandi dimensioni. Le varie categorie d'informazioni che possono essere incluse in variabili Variant sono definite sottotipi. Nella maggior parte dei casi, i dati inseriti in variabili Variant sono automaticamente gestiti nel modo più adatto.
Nella tabella seguente sono elencati i possibili sottotipi di variabili Variant.

Sottotipo	Descrizione
Empty	Variabile **Variant** non inizializzata. Il valore è 0 nel caso di variabili numeriche e una stringa di lunghezza zero ("") nel caso di variabili stringa.
Null	Variabile **Variant** che include dati non validi inseriti volutamente.
Boolean	Può contenere **True** (corrisponde a un valore pari a -1) o **False** (corrisponde a un valore pari a 0).
Byte	Contiene un intero compreso tra 0 e 255.
Integer	Contiene un intero compreso tra -32.768 e 32.767.
Currency	Contiene un valore compreso tra -922.337.203.685.477,5808 e

	922.337.203.685.477,5807.
Long	Contiene un intero compreso tra -2.147.483.648 e 2.147.483.647.
Single	Contiene un numero in virgola mobile e precisione singola compreso tra -3,402823E38 e -1,401298E-45 per valori negativi e tra 1,401298E-45 e 3,402823E38 per valori positivi.
Double	Contiene un numero in virgola mobile e precisione doppia compreso tra -1,79769313486232E308 e -4,94065645841247E-324 per valori negativi e tra 4,94065645841247E-324 e 1,79769313486232E308 per valori positivi.
Date (Time)	Contiene un numero che rappresenta una data compresa tra l'1 gennaio dell'anno 100 e il 31 dicembre del 9999.
String	Contiene una stringa di lunghezza variabile composta da un massimo di circa 2 miliardi di caratteri.
Object	Contiene un oggetto.
Error	Contiene un numero di errore.

Per le conversioni tra sottotipi è disponibile un'ampia gamma di funzioni di conversione.

Funzione Asc

Restituisce il codice di carattere ANSI corrispondente alla prima lettera di una stringa.

Asc(stringa)

L'argomento stringa può essere rappresentato da una qualsiasi espressione stringa. Se non include alcun carattere, è generato un errore di run-time. È inoltre disponibile la funzione (AscB) da utilizzare con i dati byte inclusi in una stringa. Anziché restituire il codice della prima lettera, AscB restituisce il primo byte. In piattaforme a 32 bit in cui sono utilizzati caratteri Unicode è disponibile la funzione AscW che restituisce il codice di carattere Unicode ("wide", ovvero ampio). La conversione da Unicode ad ANSI non è pertanto necessaria

Funzione Cbool

Restituisce un'espressione che è stata convertita in un valore Variant del sottotipo Boolean.

CBool(espressione)

L'argomento espressione può essere rappresentato da una qualsiasi espressione valida. Se espressione è uguale a zero, sarà restituito False. In caso contrario, sarà restituito True. Se l'argomento espressione non può essere interpretato come valore numerico, è generato un errore di run-time.

Funzione Cbyte

Restituisce un'espressione che è stata convertita in un valore Variant del sottotipo Byte.

CByte(espressione)

L'argomento espressione può essere rappresentato da qualsiasi espressione valida.

In generale, è possibile rendere più leggibile il codice utilizzando le funzioni di conversione dei sottotipi per indicare che il risultato di una determinata operazione deve essere espresso con un tipo di dati specifico anziché con il tipo di dati predefinito. La funzione CByte, ad esempio, consente di eseguire il calcolo in byte nei casi in cui è in genere eseguito il calcolo per numeri interi, in valuta oppure in precisione singola o doppia. La funzione CByte deve essere utilizzata in sostituzione di Val per eseguire conversioni da un tipo di dati a un sottotipo Byte che tengano in considerazione le diverse impostazioni internazionali. Separatori decimali e delle migliaia diversi, ad esempio, sono riconosciuti correttamente in base alle impostazioni internazionali del sistema. Se l'argomento espressione non è compreso nell'intervallo ammesso per il sottotipo Byte, è generato un errore.

Funzione Cdate

Restituisce un'espressione che è stata convertita in un valore Variant del sottotipo Date.

CData(date)
L'argomento data può essere rappresentato da una qualsiasi espressione di data valida. La funzione IsDate consente di stabilire se l'argomento data può essere convertito in un valore di data o di ora. La funzione CDate riconosce sia valori letterali di data e ora che determinati valori numerici compresi nell'intervallo di date accettabili. Nella conversione di un valore numerico in data, la parte numerica intera è convertita in data, mentre la parte frazionaria è convertita in un'ora del giorno, a partire dalla mezzanotte.
La funzione CDate riconosce i formati di data in base alle impostazioni internazionali del sistema. Se il formato di giorno, mese e anno specificato non corrisponde alle impostazioni della data, l'ordine di giorno, mese e anno potrebbe non essere riconosciuto. I formati di data estesa che includono la stringa del giorno della settimana non sono riconosciuti.

Funzione CDbl

Restituisce un'espressione che è stata convertita in un valore Variant del sottotipo Double.
CDbl(espressione)
L'argomento espressione può essere rappresentato da una qualsiasi espressione valida. In generale, è possibile rendere più leggibile il codice utilizzando le funzioni di conversione dei sottotipi per indicare che il risultato di una determinata operazione deve essere espresso con un tipo di dati specifico anziché con il tipo di dati predefinito. La funzione CDbl o CSng consente, ad esempio, di eseguire il calcolo in precisione singola o doppia nei casi in cui è in genere eseguito il calcolo su numeri interi o in valuta. La funzione CDbl deve essere utilizzata per fornire conversioni da un tipo di dati a un sottotipo Double che tengano in considerazione le diverse impostazioni internazionali. Separatori decimali o delle migliaia diversi, ad esempio, sono riconosciuti correttamente in base alle impostazioni internazionali del sistema.

Funzione Chr

Restituisce il carattere associato al codice di carattere ANSI specificato.
Chr(codicecarattere)
L'argomento codicecarattere è un numero che identifica un carattere.
I numeri da 0 a 31 corrispondono ai codici non stampabili ASCII standard. Chr(10), ad esempio, restituisce un carattere di avanzamento riga. È inoltre disponibile la funzione ChrB da utilizzare con i dati byte inclusi in una stringa. Anziché restituire un carattere che può essere composto da uno o due byte, ChrB restituisce sempre un singolo byte. Nelle piattaforme a 32 bit in cui sono utilizzati caratteri Unicode è disponibile la funzione ChrW . Dato che l'argomento di questa funzione è un codice di carattere Unicode ("wide", ovvero ampio), la conversione da ANSI a Unicode non deve essere eseguita

Funzione Cint

Restituisce un'espressione che è stata convertita in un valore Variant del sottotipo Integer.
CInt(espressione)
L'argomento espressione può essere rappresentato da una qualsiasi espressione valida. In generale, è possibile rendere più leggibile il codice utilizzando le funzioni di conversione dei sottotipi per indicare che il risultato di una determinata operazione deve essere espresso con un tipo di dati specifico anziché con il tipo di dati predefinito. La funzione CInt o CLng consente, ad esempio, di eseguire il calcolo su numeri interi nei casi in cui è in genere eseguito il calcolo in valuta oppure in precisione singola o doppia. La funzione CInt deve essere utilizzata in sostituzione di Val per eseguire conversioni da un tipo di dati a un sottotipo Integer che tengano in considerazione le diverse impostazioni internazionali. Separatori decimali o delle migliaia diversi, ad esempio, sono riconosciuti correttamente in base alle impostazioni internazionali del sistema. Se l'argomento espressione non è compreso nell'intervallo valido per il sottotipo Integer, è generato un errore. CInt si differenzia dalle funzioni Fix e Int in quanto arrotonda la parte frazionaria di un numero anziché troncarla. Quando la parte frazionaria è uguale a 0,5, con la funzione CInt è sempre arrotondata al numero pari più prossimo. 0,5, ad esempio, è arrotondato a 0, mentre 1,5 è arrotondato a 2.

Funzione CLng

Restituisce un'espressione che è stata convertita in un valore Variant del sottotipo Long.

CLng(espressione)

L'argomento expression può essere rappresentato da una qualsiasi espressione valida. In generale, è possibile rendere più leggibile il codice utilizzando le funzioni di conversione dei sottotipi per indicare che il risultato di una determinata operazione deve essere espresso con un tipo di dati specifico anziché con il tipo di dati predefinito. La funzione CInt o CLng consente, ad esempio, di eseguire il calcolo su numeri interi nei casi in cui è in genere eseguito il calcolo in valuta oppure in precisione singola o doppia. La funzione CLng deve essere utilizzata in sostituzione di Val per eseguire conversioni da un tipo di dati a un sottotipo Long che tengano in considerazione le diverse impostazioni internazionali. Separatori decimali o delle migliaia diversi, ad esempio, sono riconosciuti correttamente in base alle impostazioni internazionali del sistema. Se l'argomento espressione non è compreso nell'intervallo di valori validi per il sottotipo Long, è generato un errore. La funzione CLng si differenzia dalle funzioni Fix e Int in quanto arrotonda la parte frazionaria di un numero anziché troncarla. Se la parte frazionaria è pari a 0,5, è arrotondata sempre al numero pari pi&ugrav e; prossimo. 0,5 ad esempio è arrotondato a 0, mentre 1,5 è arrotondato a 2.

Funzione CSng

Restituisce un'espressione che è stata convertita in un valore Variant del sottotipo Single.

CSng(espressione)

L'argomento espressione può essere rappresentato da una qualsiasi espressione valida. In generale, è possibile rendere più leggibile il codice utilizzando le funzioni di conversione dei sottotipi per indicare che il risultato di una determinata operazione deve essere espresso con un tipo di dati specifico anziché con il tipo di dati predefinito. La funzione CDbl o CSng consente, ad esempio, di eseguire il calcolo in precisione singola o doppia nei casi in cui viene in genere eseguito il calcolo su numeri interi o in valuta. La funzione CSng deve essere utilizzata in sostituzione di Val per eseguire conversioni da un tipo di dati a un sottotipo Single che tengano in considerazione le diverse impostazioni internazionali. Separatori decimali o delle migliaia diversi, ad esempio, sono riconosciuti correttamente in base alle impostazioni internazionali del sistema. Se l'argomento espressione non è compresso nell'intervallo di valori validi per il sottotipo Single, è generato un errore.

Funzione CStr

Restituisce un'espressione che è stata convertita in un valore Variant del sottotipo String.

CStr(espressione)

L'argomento espressione può essere rappresentato da una qualsiasi espressione valida. In generale, è possibile rendere più leggibile il codice utilizzando le funzioni di conversione dei sottotipi per indicare che il risultato di una determinata operazione deve essere espresso con un tipo di dati specifico anziché con il tipo di dati predefinito. La funzione CStr consente, ad esempio, di assegnare il tipo di dati String al risultato. La funzione CStr deve essere utilizzata in sostituzione di Str per eseguire conversioni da un tipo di dati a un sottotipo String che tengano in considerazione le diverse impostazioni internazionali. I separatori decimali, ad esempio, sono riconosciuti correttamente in base alle impostazioni internazionali del sistema. I dati dell'argomento espressione determinano il tipo di valore restituito in base a quanto indicato nella tabella seguente.

Se espressione è	CStr restituisce
Boolean	Un valore String contenente True o False.
Date	Un valore String contenente una data nel formato breve del sistema.
Null	Un errore di run-time.
Empty	Un valore String di lunghezza zero ("").
Error	Valore String contenente la parola ErrorE seguita dal numero di errore.
Altri valori numerici	Un valore String contenente il numero.

Funzione Hex

Restituisce una stringa che rappresenta il valore esadecimale di un numero.
Hex(numero)
L'argomento numero può essere rappresentato da una qualsiasi espressione valida. L'argomento numero, se diverso da un numero intero, è arrotondato al valore intero più prossimo prima di essere valutato.

Se numero è	Hex restituisce
Null	Null.
Empty	Zero (0).
Un altro valore numerico	Fino a otto caratteri esadecimali.

È possibile rappresentare i numeri esadecimali in modo diretto facendo precedere i numeri inclusi nell'intervallo corretto da &H. Ad esempio, &H10 rappresenta il numero decimale 16 in notazione esadecimale.

Funzione Oct

Restituisce una stringa che rappresenta il valore ottale di un numero.
Oct(numero)
L'argomento numero può essere rappresentato da una qualsiasi espressione valida.
Se l'argomento numero non è un intero, è arrotondato al numero intero più prossimo prima di essere valutato.

Se numero è	La funzione Oct restituisce
Null	Null.
Empty	Zero (0).
Un altro valore numerico	Fino a 11 caratteri ottali.

È possibile rappresentare i numeri ottali in modo diretto facendo precedere i numeri inclusi nell'intervallo corretto da &O. Ad esempio, &O10 rappresenta il numero decimale 8 in notazione ottale.

Funzione VarType

Restituisce un valore che indica il sottotipo di una variabile.
VarType(nomevar)
L'argomento nomevar può essere rappresentato da una qualsiasi variabile.
Valori restituiti.

Valore	Descrizione del tipo di variabile
0	Empty (non inizializzata).
1	Null (dati non validi).
2	Integer.
3	Long integer.
4	Numero a virgola mobile in precisione singola.
5	Numero a virgola mobile in precisione doppia.
6	Currency.
7	Date.
8	String.
9	Oggetto di automazione.
10	Error.
11	Boolean.
12	Variant (solo con matrici di valori Variant).
13	Oggetto non di automazione.
17	Byte.
8192	Matrice.

La funzione VarType non restituisce mai il valore di una matrice da solo, ma tale valore è sempre aggiunto a un altro valore per indicare una matrice di tipo specifico. Il valore di Variant è restituito solo dopo essere stato aggiunto al valore della matrice per indicare che l'argomento della funzione VarType è una matrice. Il valore restituito per una matrice di interi, ad esempio, è calcolato come 2 + 8192 o 8194. Se per un oggetto è disponibile una proprietà predefinita, la funzione VarType (oggetto) restituisce il tipo della proprietà predefinita dell'oggetto. È possibile utilizzare le costanti stringa elencate di seguito in

qualsiasi punto del codice in sostituzione dei valori effettivi.

Costante	Valore	Descrizione
vbCr	Chr(13)	Ritorno a capo
vbCrLf	Chr(13) & Chr(10)	Combinazione di ritorno a capo e avanz. riga
vbFormFeed	Chr(10)	Foglio finale
vbLf	Chr(10)	Avanzamento riga
vbNewLine	Chr(13) & Chr(10)	Carattere di nuova riga specifico della piattaforma
vbNullChar	Chr(0)	Carattere con valore pari a 0
vbNullString	Stringa con valore pari a 0	

Non equivale a una stringa di lunghezza zero ("") ed è utilizzata per la chiamata di routine esterne

vbTab	Chr(9)	Tabulazione orizzontale
vbVerticalTab	Chr(11)	Tabulazione verticale

Variabili

Le variabili dichiarate con l'istruzione Dim a livello di script sono disponibili per tutte le routine incluse nello script. Se dichiarate a livello di routine, le variabili sono disponibili solo nella routine.
Le variabili dichiarate utilizzando l'istruzione Public sono disponibili nelle routine di tutti gli script in tutti i progetti. Per poter utilizzare le variabili che fanno riferimento a un oggetto, è prima necessario assegnare a tali variabili un oggetto esistente utilizzando l'istruzione Set. Fino al momento dell'assegnazione di un oggetto, il valore della variabile oggetto dichiarata è Nothing.
Le variabili Private sono disponibili soltanto nello script in cui sono dichiarate.
È inoltre possibile dichiarare le variabili in modo implicito specificandone semplicemente il nome in un punto qualsiasi dello script.
È tuttavia consigliabile non adottare questo metodo, in quanto è possibile inserire errori di ortografia nel nome della variabile in una o più posizioni dello script e ottenere di conseguenza risultati imprevisti.
Nell'esempio seguente, è dichiarata una matrice a una sola dimensione contenente 11 elementi*: Dim A(10)*
Anche se tra parentesi è indicato il numero 10, la matrice include in effetti 11 elementi. In VBScript infatti le matrici sono sempre in base zero e di conseguenza il numero degli elementi in esse contenuti corrisponde sempre al numero indicato tra parentesi più uno.

Costanti

Per impostazione predefinita le costanti sono pubbliche. All'interno delle routine tuttavia sono sempre private. Non è possibile modificare la visibilità delle costanti. All'interno di uno script, è possibile modificare la visibilità predefinita di una costante a livello di script specificando la parola chiave Private

Convenzioni di scrittura del codice VBScript

Lo scopo principale delle convenzioni di scrittura del codice è uniformare la struttura e lo stile di uno script o gruppo di script in modo che il codice risulti semplice da leggere e comprensibile a tutti. Grazie alla definizione di buone convenzioni è possibile ottenere codice sorgente preciso, leggibile e chiaro, in conformità con il codice di altri linguaggi e quanto più possibile intuitivo.

Convenzioni di denominazione delle costanti

Questa convenzione consente di utilizzare un formato con caratteri maiuscoli e minuscoli in cui i nomi delle costanti sono preceduti dal prefisso "con", ad esempio:

conCostantePersonale

Convenzioni di denominazione delle variabili

Per motivi di leggibilità e uniformità, in VBScript i nomi delle variabili devono essere

descrittivi e preceduti dai prefissi elencati nella tabella seguente.

Sottotipo	Prefisso	Esempio
Boolean	bln	blnFound
Byte	byt	bytRasterData
Date (Time)	dtm	dtmStart
Double	dbl	dblTolerance
Error	err	errOrderNum
Integer	int	intQuantity
Long	lng	lngDistance
Object	obj	objCurrent
Single	sng	sngAverage
String	str	strFirstName

Area di validità delle variabili

Nella tabella seguente sono elencate le aree di validità che è possibile associare alle variabili VBScript. È consigliabile scegliere sempre l'area di validità minima.

Area di validità	Posizione della dichiarazione della variabile	Visibilità
A livello di routine	Evento oppure routine Function o Sub	Nella routine in cui è dichiarata
A livello di script	Sezione HEAD di una pagina HTML, all'esterno di routine	In tutte le routine dello script

Prefissi per l'area di validità delle variabili

Con script di grandi dimensioni diventa particolarmente importante poter contraddistinguere l'area di validità delle variabili. A tale scopo è sufficiente aggiungere un prefisso di una lettera al prefisso del tipo di variabile in modo da evitare nomi eccessivamente lunghi.

Area di validità	Prefisso	Esempio
A livello di routine	Nessuno	dblVelocity
A livello di script	s	sblnCalcInProgress

Nomi di routine e variabili descrittivi

Il corpo del nome di variabili e routine deve descrivere nel modo più completo possibile la funzione della variabile o routine, nonché includere una combinazione di lettere maiuscole e minuscole. I nomi di routine devono inoltre iniziare con un verbo, ad esempio InitNameArray o CloseDialog. Per termini utilizzati di frequente o particolarmente lunghi, è consigliabile utilizzare abbreviazioni standard, in modo da mantenere ridotta la lunghezza del nome. In generale, i nomi di variabili composti da più di 32 caratteri risultano di difficile lettura. Le abbreviazioni devono essere utilizzate in modo uniforme. Se in uno script o gruppo di script sono, ad esempio, alternate le abbreviazioni Cnt e Count senza alcun criterio, è possibile creare confusione.

Convenzioni di denominazione degli oggetti

Nella tabella seguente sono elencate le convezioni di denominazione di vari oggetti utilizzati nella programmazione in VBScript.

Tipo di oggetto	Prefisso	Esempio
Pannello 3D	pnl	pnlGroup
Pulsante animato	ani	aniMailBox
Casella di controllo	chk	chkReadOnly
Casella combinata o di riepilogo a discesa	cbo	cboEnglish
Pulsante di comando	cmd	cmdExit
Finestra di dialogo comune	dlg	dlgFileOpen
Cornice	fra	fraLanguage
Barra di scorrimento orizzontale	hsb	hsbVolume

Immagine	img	imgIcon
Etichetta	lbl	lblHelpMessage
Linea	lin	linVertical
Casella di riepilogo	lst	lstPolicyCodes
Pulsante di selezione	spn	spnPages
Casella di testo	txt	txtLastName
Barra di scorrimento verticale	vsb	vsbRate
Dispositivo di scorrimento	sld	sldScale

Convenzioni per i commenti del codice

All'inizio delle routine è necessario includere un breve commento per descrivere il funzionamento della routine stessa. Tale commento non deve tuttavia includere informazioni dettagliate sulla modalità d'implementazione. Tali informazioni, essendo soggette a modifiche, comporterebbero infatti un inutile lavoro di manutenzione o addirittura la presenza di commenti non corretti. La modalità d'implementazione è comunque descritta nel codice stesso e nei commenti inseriti sulla stessa riga delle istruzioni. È necessario descrivere gli argomenti passati a routine la cui funzione non risulta evidente o inclusi in un determinato intervallo. All'inizio delle routine è inoltre necessario includere una descrizione dei valori restituiti da funzioni e altre variabili che sono modificate dalla routine, soprattutto se le modifiche sono apportate tramite argomenti di riferimento. Nei commenti d'intestazione delle routine è necessario includere le intestazioni di sezione indicate di seguito.

Intestazione di sezione	**Contenuto del commento**
Scopo	Operazioni eseguite dalla routine, non la modalità di esecuzione.
Presupposti	Elenco di variabili esterne, controlli e altri elementi il cui stato ha effetto sulla routine.
Effetti	Elenco dell'effetto della routine su variabili esterne, controlli e altri elementi.
Input	Descrizione degli argomenti con significato non evidente. Ciascun argomento deve essere incluso in una riga distinta insieme a un commento specifico sulla stessa riga.
Valori restituiti	Descrizione del valore restituito.

È importante tenere presente i seguenti punti.

- Le dichiarazioni di variabili importanti devono includere un commento sulla stessa riga in cui è descritto l'utilizzo della variabile dichiarata.
- I nomi di variabili, controlli e routine devono essere sufficientemente descrittivi in modo che sia necessario aggiungere commenti sulla stessa riga solo per la descrizione di implementazioni complesse.
- All'inizio degli script è consigliabile includere cenni preliminari che descrivano brevemente lo script e in cui siano elencati gli oggetti, le routine, gli algoritmi, le finestre di dialogo e altre dipendenze di sistema. A volte può risultare utile aggiungere pseudocodice che descrivi l'algoritmo.

Formattazione del codice

Lo scopo è quello di risparmiare spazio sullo schermo, ma allo stesso tempo applicare al codice una formattazione che ne rifletta in modo chiaro la struttura logica e i livelli di nidificazione. Di seguito sono indicati alcuni suggerimenti.

- I blocchi nidificati standard devono rientrare di quattro spazi.
- I commenti introduttivi di una procedura devono essere rientrati di uno spazio.
- Le istruzioni di livello principale che seguono i commenti iniziali devono rientrare di quattro spazi, con ciascun blocco nidificato rientrato di altri quattro spazi.

```
'*****************************************************
 ' Scopo: individua la prima occorrenza del nome utente specificato nella matrice
‘ UserList.
 ' Input: strUserList(): elenco dei nomi utente in cui eseguire la ricerca.
 '        strTargetUser: nome utente da cercare.
```

```
 ' Valori restituiti: indice della prima occorrenza di strTargetUser nella matrice strUserList.
 '          Se il nome utente specificato non è individuato, sarà restituito -1.
 '*******************************************************
Function intFindUser (strUserList(), strTargetUser)
    Dim i                                   ' Contatore di ciclo.
    Dim blnFound                            ' Flag elemento trovato
    intFindUser = -1
    i = 0                                   ' Inizializza il contatore di cicli
    Do While i <= Ubound(strUserList) and Not blnFound
       If strUserList(i) = strTargetUser Then
          blnFound = True   ' Imposta il flag su True
          intFindUser = i
' Imposta il valore restituito sul conteggio dei cicli
       End If
       i = i + 1                            ' Incrementa il contatore di cicli
    Loop
End Function
```

VBScript e i form

Convalida semplice

È possibile utilizzare VBScript per le operazioni di elaborazione di form che in genere è necessario eseguire in un server, nonché per le operazioni che non possono essere svolte nel server. Di seguito è riportato un esempio di convalida semplice nel client. Il codice HTML genererà una casella di testo e un pulsante.

```
<HTML>
<HEAD><TITLE>Convalida semplice</TITLE>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub Submit_OnClick
  Dim TheForm
  Set TheForm = Document.ValidForm
  If IsNumeric(TheForm.Text1.Value) Then
    If TheForm.Text1.Value < 1 Or TheForm.Text1.Value > 10 Then
      MsgBox "Immettere un numero compreso tra 1 e 10."
    Else
      MsgBox "Valore corretto."
    End If
  Else
    MsgBox "Immettere un valore numerico."
  End If
End Sub
-->
</SCRIPT>
</HEAD>
<BODY>
<H3>Convalida semplice</H3><HR>
<FORM NAME="ValidForm">
Immettere un valore compreso tra 1 e 10:
<INPUT NAME="Text1" TYPE="TEXT" SIZE="2">
<INPUT NAME="Submit" TYPE="BUTTON" VALUE="Invia">
</FORM>
</BODY>
</HTML>
```

L'unica differenza tra questo esempio e l’esempio precedente “Una semplice pagina” è

che la proprietà Value della casella di testo è utilizzata per verificare il valore immesso. Per leggere la proprietà Value, è tuttavia necessario che nel codice sia specificato il riferimento al nome della casella di testo. È in ogni modo possibile scrivere il riferimento completo Document.ValidForm.Text1. Nel caso di più riferimenti a controlli di form, è tuttavia consigliabile procedere come indicato nell'esempio, ovvero dichiarare innanzitutto una variabile e assegnare quindi il form alla variabile TheForm tramite l'istruzione Set. In questo caso non è possibile utilizzare una normale istruzione di assegnazione quale Dim.

Istruzione Set

Assegna un riferimento di oggetto a una variabile o proprietà.

Set varoggetto = { espressioneoggetto | *Nothing}*

La sintassi dell'istruzione Set è composta dalle seguenti parti.

Parte	Descrizione
varoggetto	Nome della variabile o proprietà espresso in base alle convenzioni di denominazione standard delle variabili.
espressioneoggetto	Espressione costituita dal nome di un oggetto, da un'altra variabile dichiarata dello stesso tipo di oggetto oppure da una funzione o da un metodo che restituisce un oggetto dello stesso tipo.
Nothing	Annulla l'associazione dell'argomento varoggetto a un oggetto specifico. Assegnando varoggetto al valore Nothing, sono liberate la memoria e le risorse di sistema associate all'oggetto cui è stato fatto riferimento precedentemente, a condizione che nessun'altra variabile faccia riferimento a tale oggetto.

Per essere valido, l'argomento varoggetto deve essere un oggetto dello stesso tipo dell'oggetto che vi è assegnato. Le istruzioni Dim, Private, Public e ReDim consentono semplicemente di dichiarare variabili che fanno riferimento a un oggetto. L'assegnazione di un oggetto è eseguita solo quando si utilizza l'istruzione Set indicando un oggetto specifico. Quando si utilizza l'istruzione Set per assegnare un riferimento di oggetto a una variabile, in genere non è creata alcuna copia dell'oggetto per la variabile, ma un riferimento a tale oggetto. Più variabili oggetto possono fare riferimento allo stesso oggetto. Dato che queste variabili sono riferimenti e non copie dell'oggetto, le modifiche apportate all'oggetto si riflettono automaticamente in tutte le variabili che fanno riferimento a tale oggetto.

Utilizzo di valori numerici

Nell'esempio il valore è verificato direttamente in base a un numero, ovvero è verificato che la stringa inclusa nella casella di testo sia un numero tramite la funzione IsNumeric.

Funzione IsNumeric

Restituisce un valore booleano che indica se è possibile valutare un'espressione come valore numerico.

IsNumeric(espressione)

L'argomento espressione può essere rappresentato da una qualsiasi espressione valida.

La funzione IsNumeric restituisce True se l'intero argomento espressione è riconosciuto come numero. Restituisce False in caso contrario. IsNumeric restituisce False se l'argomento espressione è un'espressione di data.

Anche se in VBScript stringhe e numeri sono convertiti automaticamente in modo adeguato, è consigliabile controllare sempre il sottotipo dei valori immessi dall'utente e utilizzare, se necessario, le funzioni di conversione. Quando si eseguono somme con i valori di caselle di testo, è inoltre necessario convertire i valori in numeri in modo esplicito. Il segno più (operatore **+**) rappresenta, infatti, sia l'addizione sia il concatenamento di

stringhe. Se, ad esempio, Text1 include "1" e Text2 include "2", il risultato ottenuto sarà il seguente:

```
A = Text1.Value + Text2.Value              ' A è "12"
A = CDbl(Text1.Value) + Text2.Value        ' A è 3
```

Convalida e restituzione di dati al server

Nell'esempio di convalida semplice è utilizzato un normale controllo pulsante. Se fosse utilizzato un controllo Submit, i dati da controllare non sarebbero mai visualizzati, perché sarebbero inoltrati immediatamente al server. Se non si utilizza il controllo Submit, è possibile verificare i dati, che tuttavia non saranno inoltrati al server. Per poter eseguire questo ulteriore trasferimento, è necessario aggiungere la seguente riga di codice.

```
<SCRIPT LANGUAGE="VBScript">
<!--
Sub Submit_OnClick
  Dim TheForm
  Set TheForm = Document.ValidForm
  If IsNumeric(TheForm.Text1.Value) Then
    If TheForm.Text1.Value < 1 Or TheForm.Text1.Value > 10 Then
      MsgBox "Immettere un numero compreso tra 1 e 10."
    Else
      MsgBox "Valore corretto."
      TheForm.Submit        ' Dati inoltrati al server.
    End If
  Else
    MsgBox "Immettere un valore numerico."
  End If
End Sub
-->
</SCRIPT>
```

Per inoltrare i dati al server, nel codice è richiamato il metodo Submit dell'oggetto form quando i dati sono corretti. Il server gestisce quindi i dati secondo la normale modalità, ma i dati sono corretti prima di essere inoltrati al server.
nella documentazione di Internet Explorer.

UTILIZZO DI VBSCRIPT CON GLI OGGETTI

Per esempio, l'oggetto Window fa riferimento alla finestra del browser nella quale è visualizzata una pagina web, quando è caricata è generato l'evento OnLoad, mentre l'evento OnUnload è generato immediatamente prima che la pagina sia scaricata. Possiede la proprietà Navigate, utilizzata per fare in modo che la finestra del browser visualizzi una pagina web ad un determinato URL. Per includere un oggetto, è necessario utilizzare il tag <OBJECT>, mentre per impostare i valori iniziali delle proprietà dell'oggetto, è necessario utilizzare il tag <PARAM>. L'utilizzo del tag <PARAM> è equivalente all'impostazione dei valori iniziali delle proprietà per un controllo di un form in Visual BASIC. Nell'esempio seguente i tag <OBJECT> e <PARAM> consentono di aggiungere il controllo Label (etichetta) ActiveX a una pagina.

```
<OBJECT
        classid="clsid:99B42120-6EC7-11CF-A6C7-00AA00A47DD2"
        id=lblActiveLbl
        width=250
        height=250
        align=left
        hspace=20
        vspace=0
>
<PARAM NAME="Angle" VALUE="90">
```

```
<PARAM NAME="Alignment" VALUE="4">
<PARAM NAME="BackStyle" VALUE="0">
<PARAM NAME="Caption" VALUE="Etichetta desiderata">
<PARAM NAME="FontName" VALUE="Verdana, Arial, Helvetica">
<PARAM NAME="FontSize" VALUE="20">
<PARAM NAME="FontBold" VALUE="1">
<PARAM NAME="FrColor" VALUE="0">
</OBJECT>
```

È possibile impostare proprietà e richiamare metodi esattamente come per i controlli nei form. Il codice seguente, ad esempio, include i controlli <FORM> che consentono di richiamare due proprietà del controllo Label.

```
<FORM NAME="LabelControls">
<INPUT TYPE="TEXT" NAME="txtNewText" SIZE=25>
<INPUT TYPE="BUTTON" NAME="cmdChangeIt" VALUE="Modifica testo">
<INPUT TYPE="BUTTON" NAME="cmdRotate" VALUE="Ruota etichetta">
</FORM>
```

Quando il form è stato definito, una routine di eventi del pulsante cmdChangeIt consente di modificare il testo dell'etichetta.

```
<SCRIPT LANGUAGE="VBScript">
<!--
Sub cmdChangeIt_onClick
        Dim TheForm
        Set TheForm = Document.LabelControls
        lblActiveLbl.Caption = TheForm.txtNewText.Value
End Sub
-->
</SCRIPT>
```

ESEMPI

Nel seguente codice è spiegato come visualizzare un insieme **File** ed eseguire un ciclo nell'insieme utilizzando l'istruzione For Each...Next:.

```
Sub ShowFolderList(folderspec)
   Dim fs, f, f1, fc, s
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set f = fs.GetFolder(folderspec)
   Set fc = f.Files
   For Each f1 in fc
      s = s & f1.name
      s = s & vbCrLf
   Next
   MsgBox s
End Sub
```

Nel seguente codice è spiegato come visualizzare un insieme **Folders** ed eseguire un ciclo nell'insieme utilizzando l'istruzione For Each...Next.

```
Sub ShowFolderList(folderspec)
   Dim fs, f, f1, fc, s
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set f = fs.GetFolder(folderspec)
   Set fc = f.SubFolders
   For Each f1 in fc
      s = s & f1.name
      s = s &  vbCrLf
   Next
```

```
    MsgBox s
End Sub
```

Nell'esempio seguente è illustrato come utilizzare il metodo **Add** per aggiungere una nuova cartella.

```
Sub AddNewFolder(path, folderName)
  Dim fso, f, fc, nf
  Set fso = CreateObject("Scripting.FileSystemObject")
  Set f = fso.GetFolder(path)
  Set fc = f.SubFolders
  If folderName <> "" Then
    Set nf = fc.Add(folderName)
  Else
    Set nf = fc.Add("Nuova cartella")
  End If
End Sub
```

Se nomecartella esiste già, sarà generato un errore.

Per visualizzare nell'insieme **Drives** le unità con supporto rimovibile non è necessario che il supporto sia inserito. Nel seguente codice è spiegato come visualizzare un insieme Drives ed eseguire un ciclo nell'insieme utilizzando l'istruzione For Each...Next.

```
Sub ShowDriveList
    Dim fs, d, dc, s, n
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set dc = fs.Drives
    For Each d in dc
        s = s & d.DriveLetter & " - "
        If d.DriveType = Remote Then
            n = d.ShareName
        Else
            n = d.VolumeName
        End If
        s = s & n & vbCrLf
    Next
    MsgBox s
End Sub
```

Nel seguente codice è utilizzato l'oggetto **Drive** per accedere alle proprietà dell'unità.

```
Sub ShowFreeSpace(drvPath)
    Dim fs, d, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set d = fs.GetDrive(fs.GetDriveName(drvPath))
    s = "Unità " & UCase(drvPath) & " - "
    s = s & d.VolumeName  & vbCrLf
    s = s & "Spazio disponibile: " & FormatNumber(d.FreeSpace/1024, 0)
    s = s & " KB"
    MsgBox s
End Sub
```

Nel seguente codice è spiegato come visualizzare un oggetto **File** insieme a una delle sue proprietà.

```
Sub ShowFileInfo(specfile)
    Dim fs, f, s
    Set fs = CreateObject("Scripting.FileSystemObject")
```

```
    Set f = fs.GetFile(specfile)
    s = f.DateCreated
    MsgBox s
End Sub
```

Nel seguente codice è spiegato come visualizzare un oggetto **Folder** insieme a una delle sue proprietà.

```
Sub ShowFolderInfo(folderspec)
    Dim fs, f, s,
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFolder(folderspec)
    s = f.DateCreated
    MsgBox s
End Sub
```

Per visualizzare le unità con supporto rimovibile nell'insieme **Drives**, non è necessario che il supporto sia inserito. È possibile eseguire un'iterazione tra gli elementi dell'insieme Drives utilizzando un blocco For Each...Next come descritto nel seguente codice.

```
Sub ShowDriveList
    Dim fs, d, dc, s, n
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set dc = fs.Drives
    For Each d in dc
        s = s & d.DriveLetter & " - "
        If d.DriveType = 3 Then
            n = d.ShareName
        Else
            n = d.VolumeName
        End If
        s = s & n & vbCrLf
    Next
    MsgBox s
End Sub
```

I server di automazione forniscono almeno un tipo di oggetto. Un programma di elaborazione di testi, ad esempio, fornisce un oggetto applicazione, un oggetto documento e un oggetto barra degli strumenti. Per creare un oggetto di automazione, è necessario assegnare ad una variabile oggetto l'oggetto restituito da **CreateObject**.

```
Dim ExcelSheet
Set ExcelSheet = CreateObject("Excel.Sheet")
```

Questo codice avvia l'applicazione creando l'oggetto, in questo caso un foglio di calcolo di Excel. È quindi possibile fare riferimento all'oggetto nel codice utilizzando la variabile oggetto definita. Nell'esempio tramite la variabile oggetto è possibile accedere alle proprietà e ai metodi del nuovo oggetto ExcelSheet e di altri oggetti di Excel, tra cui l'oggetto Application e l'insieme Cells.

```
' Rende Excel visibile tramite l'oggetto Application.
ExcelSheet.Application.Visible - True
' Inserisce del testo nella prima cella del foglio.
ExcelSheet.Cells(1,1).Value = "Questa è la colonna A, riga 1"
' Salva il foglio.
ExcelSheet.SaveAs "C:\DOCS\TEST.DOC"
' Chiude Excel tramite il metodo Quit applicato all'oggetto Application.
ExcelSheet.Application.Quit
' Rilascia la variabile oggetto.
```

```
Set ExcelSheet = Nothing
```
Per motivi di protezione, CreateObject non è disponibile in tutti gli host VBScript, ad esempio in Microsoft Internet Explorer

Il valore restituito dalla proprietà **AvailableSpace** corrisponde al valore restituito dalla proprietà **FreeSpace**. Si possono verificare differenze tra le due proprietà in sistemi che supportano i limiti. Nel seguente codice è spiegato come utilizzare la proprietà *AvailableSpace.*

```
Sub ShowAvailableSpace(drvPath)
   Dim fs, d, s
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set d = fs.GetDrive(fs.GetDriveName(drvPath))
   s = "Unità " & UCase(drvPath) & " - "
   s = s & d.VolumeName  & vbCrLf
   s = s & "Spazio disponibile: " & FormatNumber(d.AvailableSpace/1024, 0)
   s = s & " KB"
   MsgBox s
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **DateCreated** con un file.

```
Sub ShowFileInfo(specfile)
   Dim fs, f, s
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set f = fs.GetFile(specfile)
   s = "Creazione: " & f.DateCreated
   MsgBox s
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **DateLastAccessed** con un file.

```
Sub ShowFileAccessInfo(specfile)
   Dim fs, f, s
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set f = fs.GetFile(specfile)
   s = UCase(specfile) & vbCrLf
   s = s & "Creazione: " & f.DateCreated & vbCrLf
   s = s & "Ultimo accesso: " & f.DateLastAccessed & vbCrLf
   s = s & "Ultima modifica: " & f.DateLastModified
   MsgBox s, 0, "Informazioni sull'accesso al file"
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **DateLastModified** con un file.

```
Sub ShowFileAccessInfo(specfile)
   Dim fs, f, s
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set f = fs.GetFile(specfile)
   s = UCase(specfile) & vbCrLf
   s = s & "Creazione: " & f.DateCreated & vbCrLf
   s = s & "Ultimo accesso: " & f.DateLastAccessed & vbCrLf
   s = s & "Ultima modifica: " & f.DateLastModified
   MsgBox s, 0, "Informazioni sull'accesso al file"
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **Drive**.

```
Sub ShowFileAccessInfo(specfile)
   Dim fs, f, s
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set f = fs.GetFile(specfile)
   s = f.Name & " nell'unità " & UCase(f.Drive) & vbCrLf
   s = s & "Creazione: " & f.DateCreated & vbCrLf
   s = s & "Ultimo accesso: " & f.DateLastAccessed & vbCrLf
   s = s & "Ultima modifica: " & f.DateLastModified
   MsgBox s, 0, "Informazioni sull'accesso al file"
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **DriveLetter**.

```
Sub ShowDriveLetter(drvPath)
   Dim fs, d, s
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set d = fs.GetDrive(fs.GetDriveName(percUnità))
   s = "Unità " & d.DriveLetter & ": - "
   s = s & d.VolumeName  & vbCrLf
   s = s & "Spazio disponibile: " & FormatNumber(d.FreeSpace/1024, 0)
   s = s & " KB"
   MsgBox s
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **DriveType**.

```
Sub ShowDriveType(drvPath)
   Dim fs, d, s, t
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set d = fs.GetDrive(drvPath)
   Select Case d.DriveType
      Case 0: t = "sconosciuta"
      Case 1: t = "rimovibile"
      Case 2: t = "fissa"
      Case 3: t = "rete"
      Case 4: t = "CD-ROM"
      Case 5: t = "disco RAM"
   End Select
   s = "Unità " & d.DriveLetter & ": - " & t
   MsgBox s
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **Files**.

```
Sub ShowFileList(folderspec)
   Dim fs, f, f1, fc, s
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set f = fs.GetFolder(folderspec)
   Set fc = f.Files
   For Each f1 in fc
      s = s & f1.name
      s = s &  vbCrLf
   Next
   MsgBox s
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **FileSystem**.

```
Sub ShowFileSystemType
   Dim fs,d, s
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set d = fs.GetDrive("e:")
   s = d.FileSystem
   MsgBox s
End Sub
```

Nel caso di unità con supporto rimovibile e unità CD-ROM, IsReady restituisce True solo quando il supporto appropriato è inserito e pronto per l'accesso. Nel seguente codice è spiegato come utilizzare la proprietà **IsReady**.

```
Sub ShowDriveInfo(drvpath)
   Dim fs, d, s, t
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set d = fs.GetDrive(drvpath)
   Select Case d.DriveType
      Case 0: t = "sconosciuta"
      Case 1: t = "rimovibile"
      Case 2: t = "fissa"
      Case 3: t = "rete"
      Case 4: t = "CD-ROM"
      Case 5: t = "disco RAM"
   End Select
   s = "Unità " & d.DriveLetter & ": - " & t
   If d.IsReady Then
      s = s & vbCrLf & "Unità pronta".
   Else
      s = s & vbCrLf & "Unità non pronta".
   End If
   MsgBox s
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **Name**.

```
Sub ShowFileAccessInfo(specfile)
   Dim fs, f, s
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set f = fs.GetFile(specfile)
   s = f.Name & " nell'unità " & UCase(f.Drive) & vbCrLf
   s = s & "Creazione: " & f.DateCreated & vbCrLf
   s = s & "Ultimo accesso: " & f.DateLastAccessed & vbCrLf
   s = s & "Ultima modifica: " & f.DateLastModified
   MsgBox s, 0, "Informazioni sull'accesso al file"
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **ParentFolder** con un file.

```
Sub ShowFileAccessInfo(specfile)
   Dim fs, f, s
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set f = fs.GetFile(specfile)
   s = UCase(f.Name) & " in " & UCase(f.ParentFolder) & vbCrLf
   s = s & "Creazione: " & f.DateCreated & vbCrLf
   s = s & "Ultimo accesso: " & f.DateLastAccessed & vbCrLf
   s = s & "Ultima modifica: " & f.DateLastModified
   MsgBox s, 0, "Informazioni sull'accesso al file"
```

```
End Sub
```

Nel caso di lettere di unità, l'unità principale non è inclusa. Il percorso dell'unità C è, ad esempio, C: e non C:\. Nel seguente codice è spiegato come utilizzare la proprietà **Path** con un oggetto File.

```
Sub ShowFileAccessInfo(specfile)
   Dim fs, d, f, s
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set f = fs.GetFile(specfile)
   s = UCase(f.Path) & vbCrLf
   s = s & "Creazione: " & f.DateCreated & vbCrLf
   s = s & "Ultimo accesso: " & f.DateLastAccessed & vbCrLf
   s = s & "Ultima modifica: " & f.DateLastModified
   MsgBox s, 0, "Informazioni sull'accesso al file"
End Sub
```

È possibile utilizzare la proprietà SerialNumber per assicurarsi di avere inserito il disco corretto in un'unità con supporto rimovibile. Nel seguente codice è spiegato come utilizzare la proprietà **SerialNumber**.

```
Sub ShowDriveInfo(drvpath)
   Dim fs, d, s, t
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set d = fs.GetDrive(fs.GetDriveName(fs.GetAbsolutePathName(drvpath)))
   Select Case d.DriveType
      Case 0: t = "sconosciuta"
      Case 1: t = "rimovibile"
      Case 2: t = "fissa"
      Case 3: t = "rete"
      Case 4: t = "CD-ROM"
      Case 5: t = "disco RAM"
   End Select
   s = "Unità " & d.DriveLetter & ": - " & t
   s = s & vbCrLf & "SN: " & d.SerialNumber
   MsgBox s
End Sub
```

Se oggetto non corrisponde a un'unità di rete, la proprietà ShareName restituirà una stringa di lunghezza zero (""). Nel seguente codice è spiegato come utilizzare la proprietà **ShareName**.

```
Sub ShowDriveInfo(drvpath)
   Dim fs, d, s
   Set fs = CreateObject("Scripting.FileSystemObject")
  Set d = fs.GetDrive(fs.GetDriveName(fs.GetAbsolutePathName(drvpath)))
   s = "Unità " & d.DriveLetter & ": - " & d.ShareName
   MsgBox s
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **ShortName** con un oggetto File.

```
Sub ShowShortName(specfile)
   Dim fs, f, s
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set f = fs.GetFile(specfile)
   s = "Il nome abbreviato di " & "" & UCase(f.Name)
```

```
    s = s & "" & vbCrLf
    s = s & "è: " & "" & f.ShortName & ""
    MsgBox s, 0, "Informazioni sul nome abbreviato"
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **ShortName** con un oggetto File.

```
Sub ShowShortPath(specfile)
    Dim fs, f, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFile(specfile)
    s = "Il percorso breve di " & "" & UCase(f.Name)
    s = s & "" & vbCrLf
    s = s & "è: " & "" & f.ShortPath & ""
    MsgBox s, 0, "Informazioni sul percorso breve"
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **Size** con un oggetto Folder.

```
Sub ShowFolderSize(specfile)
    Dim fs, f, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFolder(specfile)
    s = UCase(f.Name) & " utilizza " & f.size & " byte".
    MsgBox s, 0, "Informazioni sulle dimensioni della cartella"
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **SubFolders**.

```
Sub ShowFolderList(folderspec)
    Dim fs, f, f1, s, sf
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.GetFolder(folderspec)
    Set sf = f.SubFolders
    For Each f1 in sf
        s = s & f1.name
        s = s &  vbCrLf
    Next
    MsgBox s
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **TotalSize**.

```
Sub ShowSpaceInfo(drvpath)
    Dim fs, d, s
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set d = fs.GetDrive(fs.GetDriveName(fs.GetAbsolutePathName(drvpath)))
    s = "Unità " & d.DriveLetter & ":"
    s = s & vbCrLf
    s = s & "Dimensione totale: " & FormatNumber(d.TotalSize/1024, 0) & " KB"
    s = s & vbCrLf
    s = s & "Spazio disponibile: " & FormatNumber(d.AvailableSpace/1024, 0) & " KB"
    MsgBox s
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **Type** per restituire il tipo di cartella. In questo esempio, provare a fornire alla routine il percorso del Cestino o di

un'altra cartella univoca.

```
Sub ShowFileSize(specfile)
   Dim fs, f, s
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set f = fs.GetFolder(specfile)
   s = UCase(f.Name) & " è di tipo " & f.Type
   MsgBox s, 0, "Informazioni sulle dimensioni del file"
End Sub
```

Nel seguente codice è spiegato come utilizzare la proprietà **VolumeName**.

```
Sub ShowVolumeInfo(drvpath)
   Dim fs, d, s
   Set fs = CreateObject("Scripting.FileSystemObject")
   Set d = fs.GetDrive(fs.GetDriveName(fs.GetAbsolutePathName(drvpath)))
   s = "Unità " & d.DriveLetter & ": - " & d.VolumeName
   MsgBox s
End Sub
```

Nell'esempio di codice HTML riportato di seguito, il contenuto di un oggetto Dictionary è utilizzato per inserire testo in varie caselle di testo.

```
<HTML>
<HEAD><TITLE>Forms and Elements</TITLE></HEAD>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub cmdChange_OnClick
   Dim d                'Crea una variabile
   Set d = CreateObject("Scripting.Dictionary")
   d.Add "0", "Atene"     'Aggiunge alcune chiavi ed elementi
   d.Add "1", "Belgrado"
   d.Add "2", "Cairo"
   For Each I in d
      Document.frmForm.Elements(I).Value = D.Item(I)
   Next
End Sub
-->
</SCRIPT>
<BODY>
<CENTER>
<FORM NAME="frmForm"
<Input Type = "Text"><p>
<Input Type = "Text"><p>
<Input Type = "Text"><p>
<Input Type = "Text"><p>
<Input Type = "Button" NAME="cmdChange" VALUE="Fare clic qui"><p>
</FORM>
</CENTER>
</BODY>
</HTML>
```

UBERTINI MASSIMO
http://www.ubertini.it
massimo@ubertini.it

Dip. Informatica Industriale
I.T.I.S. "Giacomo Fauser"
Via Ricci, 14
28100 Novara Italy
tel. +39 0321482411
fax +39 0321482444
http://www.fauser.edu
massimo@fauser.edu