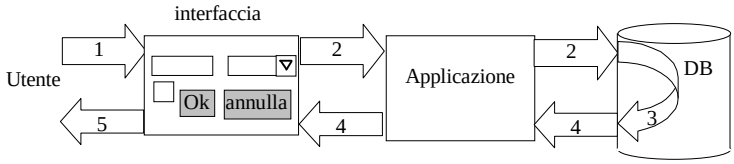
Esercizio 1.

Pubblicazione di Data Base in rete.

Architettura di un sistema di consultazione di un data base



Un sistema di accesso a un data base è organizzato in tre componenti: 1) l'interfaccia, 2) l'applicazione, 3) un *oggetto* data base. Il termine 'oggetto' in questo contesto allude a una struttura contenente proprietà e metodi.

Il processo di accesso al data base consiste delle seguenti fasi:
1) l'interfaccia propone all'utente un 'dialogo' basato su caselle di testo, pulsanti di opzione, bottoni ecc. contenuti in un form (o maschera);
2) Le informazioni fornite dall'utente vengono consegnate alla componente 'applicazione' che costruisce il comando (di interrogazione, di cancellazione, di inserimento o di modifica);
3) Il comando viene presentato al data base. I dati vengono estratti, eliminati, aggiunti o modificati, a seconda dell'operazione avviata dall'utente;
4) Il risultato dell'operazione viene acquisito dalla componente applicazione che si occupa di interpretarlo e di rappresentalro alla componente interfaccia;
5) L'interfaccia presenta il risultato all'utente.

Come ogni architettura modulare, i vari componenti
1) possono essere sostituiti con moduli più aggiornati;
2) svolgono ruoli specifici permettendo una più facile individuazione dei guasti;
3) possono essere eseguiti da processori residenti anche su macchine diverse.

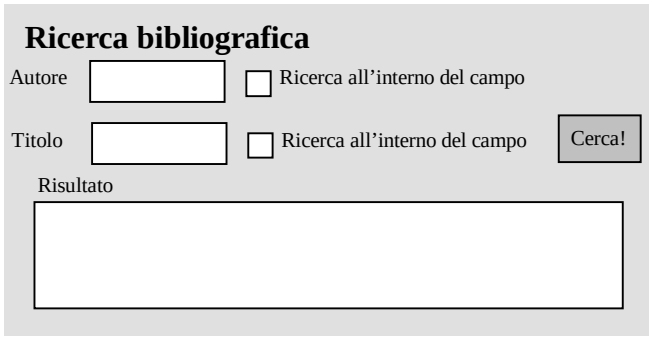
Preparazione del data base.
In MS-Access aprire un data base nuovo dal nome *LibriWeb.mdb* (l'estensione ovviamente l'aggiunge Access) e salvarlo in una sottocartella di Inetpub\wwwroot. Nel data base creare una nuova tabella con la seguente struttura:

Nome Campo	Tipo dati	Descrizione
Autore	Testo	
Titolo	Testo	
Commenti	Memo	

Salvare la tabella con il nome *TabBiblio*.

Chiudere la struttura della tabella. Aprire la tabella e inserirvi una serie libri.

Nella sezione Maschere, preparare l'interfaccia utente contenente un titolo (Ricerca bibliografica), due caselle di testo, una con didascalia 'Autore', l'altra con didascalia 'Titolo', due checkbox, entrambe con didascalia 'Ricerca all'interno del campo', una casella Memo con intestazione Risultato e un bottone con didascalia 'Cerca!'.



Assegnare i seguenti nomi (tramite la finestra Proprietà)
- alle caselle di testo. sAutore e sTitolo;
- ai checkbox: bAutore e bTitolo;
- alla casella Memo: mRisultato.

Salvare e chiudere la maschera.

Al pulsante *Cerca* bisogna associare un gestore associato all'evento `onClick` che svolge il ruolo della componente *Applicazione*: deve acquisire i dati dall'interfaccia utente e costruire un comando per svolgere un'operazione di ricerca nel data base.

Per determinare la sintassi corretta del comando ci si può servire del generatore di Query.

Selezionare la scheda Query. Premere il pulsante Nuovo e preparare una query per estrarre dalla tabella tutti i record il cui autore abbia l'iniziale = "P" e il cui titolo contenga la parola "CPU". Se dal menu Visualizza si sceglie il comando *Visualizza SQL* si dovrebbe leggere l'istruzione SQL corrispondente alla query, ad esempio:

```
SELECT TabBiblio.Autore, TabBiblio.Titolo, TabBiblio.Commenti
FROM TabBiblio
WHERE (([TabBiblio]![Autore]="P*") AND ([TabBiblio]![Titolo]="*CPU*"));
```

Non è necessario salvare questa query: di essa si è interessati solo alla sintassi corretta, perché la costruirà la componente Applicazione, dopo aver acquisito dall'interfaccia utente i parametri di ricerca. In risposta a una query Access restituisce una tabella temporanea (un RecordSet) contenente i dati rispondenti al criterio di ricerca.

Esercizio 2

Il gestore di evento onClick.

```
Private Sub bCerca_Click()
Dim lAutore As String, lTitolo As String, lOut As String
Dim lAutore2 As Boolean, lTitolo2 As Boolean
Dim wildChar As String, SQLstr As String, lineSep As String
```

Parte 1. Acquisizione dei parametri di ricerca dall'interfaccia utente (in VBA l'oggetto `Me` identifica la maschera). Convalida dei dati:

```
  If IsNull(Me.sAutore) And IsNull(Me.sTitolo) Then ' se i campi sono vuoti …
  MsgBox ("Attenzione: indicare una condizione di ricerca")
  End ‘ … si arresta l'elaborazione.
  End If

  lAutore = Trim(Me.sAutore.Value) ‘ TRIM elimina gli eventuali spazi intorno alla stringa
  lAutore2 = Me.bAutore.Value      ‘ acquisisce il valore del checkbox.
  lTitolo = Trim(Me.sTitolo.Value)
  lTitolo2 = Me.bTitolo.Value
```

Parte 2. Composizione del comando di ricerca.

```
  wildChar = "*" ‘ carattere jolly
  SQLstr = "" ‘ inizialmente la stringa di comando SQL è vuota.

If Not IsNull(lAutore) Then         ‘ L'utente vuole ricercare un autore:
    If lAutore2 Then
      SQLstr = "Autore like '" & wildChar & lAutore & wildChar & "'"
    Else
      SQLstr = "Autore like '" & lAutore & wildChar & "'"
    End If
  End If

  If Not IsNull(lTitolo) Then       ‘ L'utente vuole ricercare un titolo:
    If SQLstr <> "" Then SQLstr = SQLstr & " AND "
      If lTitolo2 Then
        SQLstr = SQLstr & "Titolo LIKE '" & wildChar & lTitolo & wildChar & "'"
      Else
        SQLstr = SQLstr & "Titolo LIKE '" & lTitolo & wildChar & "'"
    End If
  End If
  SQLstr = "SELECT * From TabBiblio WHERE (" & SQLstr & ");"
```

Parte 3. Connessione al DB per passare il comando SQL e acquisire il risultato in un RecordSet

```
  Dim oReco As Recordset
  Set oReco = CurrentDb.OpenRecordset(SQLstr)
```

Il tipo `Recordset` e il metodo `OpenRecordset` dell'oggetto `CurrentDb` non appartengono al linguaggio VBA, essi sono definiti nella libreria di oggetti DAO (Direct Access Object) a cui l'interprete Visual Basic di Access attinge per tradurre le istruzioni di accesso a data base. Pertanto quando si dichiara un'istanza di un oggetto di classe **Recordset** bisogna includere, tramite il menu *Strumenti*, il *Riferimento* alla libreria 'Microsoft DAO 3.6 Object Library'.

La prima delle due istruzioni precedenti richiama il costruttore di un oggetto di classe `Recordset`, la seconda inizializza i campi dell'oggetto `oReco` con il risultato della query `SQLstr`.

Parte 4. Preparazione del risultato da consegnare all'interfaccia utente.

```
  If oReco.EOF Then
      MsgBox "Nessun Record trovato"
      End
  End If
  lOut = ""
  lineSep = Chr(13) & Chr(10)
  While Not oReco.EOF
      lOut = lOut & oReco("Autore") & ", " & oReco("Titolo") & " - " &
      oReco("Commenti") & lineSep
      oReco.MoveNext
  Wend
```

Parte 5. Presentazione del risultato

```
  Me.mRisultato = lOut
End Sub
```

Esercizio 3.

Configurazione File Server.

Se il data base deve essere utilizzato in un ambiente multi utente allora la componente Data base deve essere disponibile su un calcolatore Server, mentre le componenti Interfaccia utente e Applicazione risiedono sui singoli calcolatori Utente. La condivisione del data base fornisce i seguenti vantaggi:

- Le prestazioni non vengono degradate: ogni utente usa la parte di data base di suo interesse, anziché procurarsi una copia locale dell'intero data base;
- L'accesso ai dati è controllato tramite opportuni *diritti di accesso* assegnati agli utenti;
- Il sistema offre un accesso in mutua esclusione ai record in fase di modifica e di conseguenza gli utenti sono sicuri di usare sempre informazioni aggiornate.

Per simulare una tale applicazione si prepari, in Word, una maschera (Menu *Visualizza – Barre degli strumenti - casella degli strumenti*) con le caselle di testo di nome `sAutore` e `sTitolo`, i checkbox di nome `bAutore` e `bTitolo`, la casella di testo `mRisultato` e il pulsante `Cerca`. A questo pulsante associare il gestore di evento onClick.

```
Private Sub bCerca_Click()
```

Parte 1. Acquisizione dei parametri di ricerca dall'interfaccia utente (in VBA l'oggetto `Me` identifica la maschera).

```
  lAutore = Trim(Me.sAutore.Value) ‘ TRIM elimina gli eventuali spazi intorno alla stringa
  lAutore2 = Me.bAutore.Value       ‘ acquisisce il valore del checkbox.
  lTitolo = Trim(Me.sTitolo.Value)
  lTitolo2 = Me.bTitolo.Value
```

Convalida dei dati. Il confronto viene fatto con una stringa vuota anziché con il carattere Null:

```
  If lAutore=”” And lTitolo=”” Then ' se i campi sono vuoti …
  MsgBox ("Attenzione: indicare una condizione di ricerca")
  End ‘ … si arresta l’elaborazione.
  End If
```

Parte 2. Composizione del comando di ricerca.

```
  wildChar = "%" ‘ il carattere jolly non è * (Word non usa la libreria ADODB anziché DAO)
  SQLstr = "" ‘ inizialmente la stringa di comando SQL è vuota.

If lAutore <> ”” Then ‘L’utente vuole ricercare un autore:
    If lAutore2 Then
      SQLstr = "Autore like '" & wildChar & lAutore & wildChar & "'"
    Else
      SQLstr = "Autore like '" & lAutore & wildChar & "'"
    End If
  End If

If lTitolo <> ”” Then ‘L’utente vuole ricercare un titolo:
    If SQLstr <> "" Then SQLstr = SQLstr & " AND "
      If lTitolo2 Then
        SQLstr = SQLstr & "Titolo LIKE '" & wildChar & lTitolo & wildChar & "'"
      Else
        SQLstr = SQLstr & "Titolo LIKE '" & lTitolo & wildChar & "'"
    End If
  End If
  SQLstr = "SELECT * From TabBiblio WHERE (" & SQLstr & ");"
```

Parte 3. Connessione al DB per passare il comando SQL e acquisire il risultato in un RecordSet

```
  Dim oConn As ADODB.Connection, oReco As ADODB.Recordset
  DBtoOpen = "libriWeb.mdb"
  Set oConn = CreateObject("adodb.connection")
  oConn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & DBtoOpen
  Set oReco = CreateObject("ADODB.Recordset")
  oReco.Open SQLstring, oConn
```

La libreria di oggetti usata da Word è la libreria ADO (ActiveX Data Object). L'interprete VBA deve essere informato di includere questa libreria nella traduzione spuntando la casella ADO, nel riquadro richiamabile tramite il menu Strumenti (dell'ambiente VBA Editor) e il comando Riferimenti.

Parte 4. Preparazione del risultato da consegnare all'interfaccia utente.

```
  If oReco.EOF Then
      MsgBox "Nessun Record trovato"
      End
  End If
  lOut = ""
```

```
   lineSep = Chr(13) & Chr(10)
   While Not oReco.EOF
       lOut = lOut & oReco("Autore") & ", " & oReco("Titolo") & " - " &
       oReco("Commenti") & lineSep
       oReco.MoveNext
   Wend
```

Parte 5. Presentazione del risultato

```
   Me.mRisultato = lOut
End Sub
```

Il supporto OLE (Object Linking and Embedding). Con il supporto OLE le applicazioni lavorano in collaborazione per produrre un unico documento contenente disegni, tabelle, ecc.. L'intero documento può essere modificato come un'unica entità ed evita quindi la necessità di apportare modifiche separatamente ai singoli componenti e poi ricombinarli nuovamente.

La tecnologia OLE consente di chiamare, quando occorre, l'applicazione appropriata per lavorare su un componente del documento. Se un editor di testo non supportasse la tecnologia OLE allora l'operazione di copiare un intervallo di celle da un foglio Excel e incollarle in un documento di testo comporterebbe che nel documento verrebbe inserita solo la rappresentazione in formato testo dei valori contenuti nelle celle. Se invece l'editor di testo gestisce l'OLE allora possono essere incollati i veri valori numerici e le corrispondenti formule contenute nelle celle. La rappresentazione in forma di testo verrebbe ancora usata, ma solo a scopo di visualizzazione. Una modifica dei valori nelle celle è possibile perché l'applicazione contenitrice cede il controllo a Excel.

Applicazioni Client e applicazioni Server. Un'applicazione che può incorporare un oggetto o un riferimento a un oggetto nei suoi documenti è detta applicazione Client. Un'applicazione che può produrre elementi destinati a essere inseriti in documenti di un'applicazione Client è detta applicazione Server. Nell'esempio precedente Word è un'applicazione Client e Excel un'applicazione Server.

Il documento di un'applicazione Client deve essere in grado di memorizzare e mostrare gli elementi OLE come se fossero dati propri. Il Client deve consentire all'utente di inserire nuovi elementi o di modificare quelli già inseriti. Client e Server non comunicano direttamente, ma impiegano funzioni di libreria che fungono da intermediarie. Questo sistema di comunicazione ha lo scopo di rendere le applicazioni indipendenti, cioè un'applicazione Client non ha bisogno di conoscere l'organizzazione dell'applicazione Server. Di conseguenza un documento Client potrà inserire oggetti provenienti da applicazioni Server future.

Il modello COM (Component Object Model). Le tecnologie OLE si basano su uno schema denominato *modello di componenti ad oggetti*. Un componente di un documento possiede un'interfaccia che è una lista di puntatori a funzione. L'oggetto cioè incapsula le proprietà e i metodi, e mostra solo le funzioni per usare l'oggetto.

Per ottenere l'accesso all'interfaccia contenuta in un oggetto si deve ottenere un puntatore all'interfaccia stessa. Questo puntatore è ottenibile tramite il sistema operativo. Infatti il programma che può rendere disponibili le sue funzionalità deve registrarsi nel registro di configurazione del sistema.

OLE DB. La tecnologia OLE DB generalizza il concetto di applicazioni Client e Server introducendo la terminologia di Produttore (Provider) e Consumatore (Consumer). Il Client OLE DB ricorre ai servizi di un produttore: Il Produttore fornisce un oggetto COM che oltre ai dati fornisce anche i metodi per trattarli.

La tecnologia OLE DB sostituisce ODBC (Open Data Base Connectivity) introducendo una maggiore flessibilità nell'accesso a dati, nel senso che mentre ODBC richiede l'esistenza di uno specifico driver per ciascun possibile insieme di dati organizzati in data base, con OLE DB è consentito l'accesso a tipi di dati di natura generale (database, posta elettronica, processi in esecuzione, …)

Connessione a un data base

L'istruzione

```
Set oConn = CreateObject("ADODB.Connection")
```

inizializza la variabile `oConn` con il puntatore a un oggetto di classe `Connection`, definito nella libreria `ADODB` (ActiveX Data Objects Data Base), e che ha lo scopo di offrire proprietà e metodi per utilizzare un data base.

In particolare il metodo `Open`, usato nell'istruzione

```
oConn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & DbtoOpen
```

richiede come parametro una stringa che contiene il nome del provider e il percorso del data base. Il nome del Provider è registrato nel registro di configurazione come `Microsoft.Jet.OLEDB.4.0`. Il nome Jet è quello del motore con cui Access gestisce i data base. Il metodo `Open` dell'oggetto `oConn` di classe `Connection` agisce come costruttore.

Quando si esegue una query su un data base il risultato è un insieme di record, che soddisfano il criterio di ricerca specificato dalla query, disponibili in una tabella temporanea denominata RecordSet. L'istruzione seguente

```
set oReco = CreateObject("ADODB.Recordset")
```

produce un oggetto di classe RecordSet prelevandone la definizione dalla libreria ADODB. Il metodo Open agisce anche in questo caso da costruttore.

```
oReco.Open comando SQL, oConn
```

il Metodo `Open` di un oggetto `RecordSet` richiede due parametri: il comando SQL e il puntatore all'oggetto Connessione.

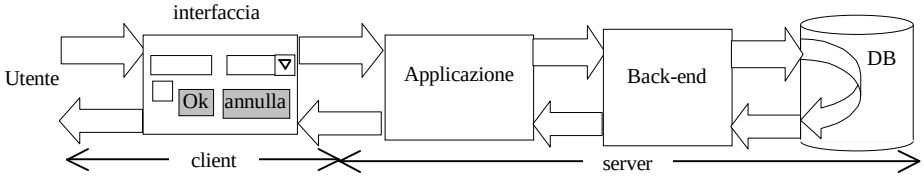
A questo punto l'oggetto `oReco` contiene la tabella con i record estratti dal data base e offre i metodi per accedere ai campi dei singoli record.

Esercizio 4

Elaborazione Server-Side con le Active Server Page (ASP).

In una configurazione distribuita, un data base può essere interrogato simultaneamente da più utenti, con la conseguenza che si deve portare l'interfaccia utente sui calcolatori client. La componente Applicazione risiede sul server, e quando riceve una richiesta da un client prepara il comando di interrogazione e delega il compito di accedere al data base a un programma eseguibile.

Un programma mandato in esecuzione dal server riceve i dati dal server stesso e consegna i risultati della sua elaborazione ancora al server. Il funzionamento di questo programma è nascosto all'interfaccia utente, si dice che viene eseguito *dietro* al server e perciò è detto back-end.



ASP è un ambiente per lo sviluppo di pagine Web che ospita gli interpreti VBScript e JScript. Inoltre mette a disposizione del programmatore alcuni oggetti che verranno esaminati di volta in volta che se ne presenterà la necessità nel corso della costruzione di un'applicazione di accesso a DB tramite pagine Web..

Il metodo Write(Stringa) dell'oggetto Response.

Un documento ASP viene prodotto tramite un comune text editor e salvato con estensione .asp all'interno di una cartella per la quale siano abilitati i diritti di esecuzione.

L'esecuzione di un documento asp viene avviata in vari modi:

- assegnando il nome del file all'attributo `action` del `form` incluso in una pagina html,
- assegnando il nome del file all'attributo `Href` di un tag `A`,
- scrivendo l'indirizzo nella barra dell'indirizzo del browser.

Nei documenti di tipo ASP gli script (in VBS) sono racchiusi tra i tag `<%` e `%>`.

Documento `Ncasuali.asp`

```
<%
Function FineCiclo
  Randomize
  FineCiclo = Int(10*Rnd)+1
End Function
%>
<html> <head>
<meta http-equiv="Refresh" content="3">
</head> <body>
<h1>Elenco di numeri generato da un back end </h1>
<%
For i=1 To FineCiclo
  Response.Write i & " "
Next
%>
</body> </html>
```

Impostando nella barra dell'indirizzo `http://localhost/`*`cartella`*`/Ncasuali.asp` il browser riceverà una pagina contenente solo i risultati. Cioè il programma è stato elaborato dal server e, alla risposta http, sono stati allegati solo i valori numerici che il back end ha consegnato al server con il comando `Response.Write i & " "`.

Acquisizione dei parametri con il metodo GET.

Per introdurre una certa interattività nel programma di generazione dei numeri compresi tra 1 e N, si supponga che sia l'utente a specificare gli estremi dell'intervallo tramite un form che si presenta con due caselle di testo e un pulsante di tipo *Submit*.

Valore Iniziale [ ] Valore Finale [ ] Ok

Il documento Html (`Intervallo.htm`) che acquisisce i valori e li passa al server è il seguente:

```
<Html> <Body>
<h1>Interfaccia utente di un'applicazione di generazione di numeri</h1>
<form action="ElencoNumeri.asp" method=”get”>
Valore Iniziale: <input type="Text" name="Inizio">
Valore Finale: <input type="Text" name="Fine"> <input type="Submit" Value="Ok">
</Form> </Body> </Html>
```

La pressione del tasto Ok (submit), in un form avente `method=”GET”`, da parte dell’utente provoca l’invio delle coppie `nome=valore`, dei singoli componenti presenti sul form, subito dopo l’indirizzo. La fine dell’indirizzo è individuata con il carattere “?” e i parametri sono separati tra loro dal carattere “&” come nel seguente esempio:

`http://localhost/`*`cartella`*`/ElencoNumeri.asp?Inizio=1&Fine=25`

Il server rende disponibili i dati, ai back-end ASP, nella *collezione* `QueryString` dell’oggetto `Request`. Una collezione è una struttura dati in cui gli elementi sono accessibili, oltre che con un indice, anche tramite una *chiave* (l’attributo `name` del componente). Il seguente documento viene mandato in esecuzione dal server quando si preme il pulsante submit del form presente nel documento precedente (`Intervallo.htm`)

Il file `ElencoNumeri.asp`

```
<Html> <Body> <h1>Risultato della generazione di numeri</h1>
<%
Iniziale = Request.QueryString("Inizio")
Finale = Request.QueryString("Fine")
For i=Iniziale To Finale
  Response.Write i & " "
Next
%> </body> </Html>
```

Il server memorizza nella collezione QueryString i dati che il browser allega all’indirizzo. Quindi non è necessario far inserire all’utente i dati predefiniti, ma questi possono essere aggiunti all’indirizzo specificato nel collegamento ipertestuale definito con il tag A. Nel file `Intervallo.htm` inserire, prima del tag Form, le seguenti linee:

```
<A HREF=”http://localhost/cartella/ElencoNumeri.asp?Inizio=1&Fine=10”>Elenco da 1 a 10</A>
<A HREF=”http://localhost/cartella/ElencoNumeri.asp?Inizio=11&Fine=20”>Elenco da 11 a 20</A>
```

Gli inconvenienti del metodo Get possono essere così riassunti: 1) I caratteri speciali (&, ?, <. …) devono essere inviati come codici ASCII, 2) Non si può superare la massima lunghezza di 2 KB dell’indirizzo, 3) I valori sono visibili all’utente (una password verrebbe mostrata).

Acquisizione dei parametri con il metodo POST.
Il browser allega i dati (le coppie `name=valore`) di un form avente il metodo POST subito dopo l’intestazione del pacchetto http, e queste coppie sono poi disponibili alle pagine Asp nella collezione `Form` dell’oggetto `Request`.

Modificare il documento `Intervallo.htm` nel seguente modo:

```
<Html> <Body>
<h1>Form con metodo GET</h1>
<form action="ElencoNumeri.asp" method=”get”>
Valore Iniziale: <input type="Text" name="Inizio">
Valore Finale: <input type="Text" name="Fine"> <input type="Submit" Value="Ok">
</Form>
<h1>Form con metodo POST</h1>
<form action="ElencoNumeri.asp" method=”Post”>
Valore Iniziale: <input type="Text" name="Inizio">
Valore Finale: <input type="Text" name="Fine"> <input type="Submit" Value="Ok">
</Form>
</Body> </Html>
```

Per generalizzare l’elaborazione lato server, in modo che i dati possano essere acquisiti sia se inviati con il metodo Get sia se inviati con il metodo Post, si può ricorrere alla collezione `ServerVariables` dell’oggetto `Request`. Questa collezione contiene variabili controllate dal server. In particolare la variabile `REQUEST_METHOD` viene aggiornata dal server con il valore dell’attributo method del form a cui appartengono i dati. Il documento mandato in esecuzione dal server `ElencoNumeri.asp` diventa:

```
<Html> <Body> <h1>Risultato della generazione di numeri</h1>
<%
If Request.ServerVariables(“REQUEST_METHOD”)=”GET” Then
  Iniziale = Request.QueryString("Inizio")
  Finale = Request.QueryString("Fine")
Else
  Iniziale = Request.Form("Inizio")
  Finale = Request.Form("Fine")
End If
For i=Iniziale To Finale
  Response.Write i & " "
Next
%> </body> </Html>
```

Esercizio5

L'oggetto Server di ASP.

Il metodo `CreateObject(Classe di oggetto)` restituisce il riferimento a un'istanza di un oggetto. Si considerino le istruzioni seguenti:

```
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & DbtoOpen
```

L'istruzione `Set` specifica che la variabile `Conn` deve contenere il riferimento a un oggetto. ADODB è una libreria di classi di oggetti. Pertanto la prima istruzione associa alla variabile `Conn` un oggetto di classe `Connection`, che è una classe avente proprietà e metodi per accedere a un data base.

La seconda istruzione usa il metodo `Open` dell'oggetto `Conn`, specificando il motore di accesso al data base e il nome del data base. Con una coppia di istruzioni analoghe si inizializza anche il RecordSet.

Un altro metodo dell'oggetto Server di ASP è `MapPath(nome di File)`, utile per trasformare l'indirizzo relativo di un file nel suo percorso fisico. Ad esempio se il data base db1.mdb si trova nella sottocartella miodb, la seguente istruzione

```
Server.MapPath(miodb/db1.mdb)
```

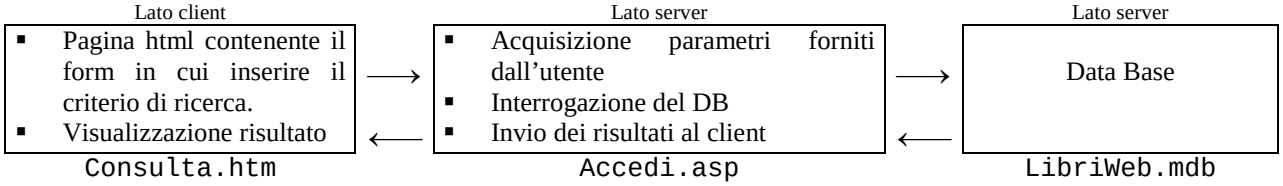
Restituisce il percorso

```
C:\\Inetpub\wwwroot\miodb\db1.mdb
```

Adottando questo metodo si conseguono due risultati positivi:

- L'applicazione può essere trasportata da un server a un altro senza alcuna modifica del codice,
- Alla cartella contenente il data base si può togliere il diritto di lettura, per evitare che un utente usi l'indirizzo del data base per trasferire il file sul proprio computer.

Accesso a data base da un client.



il documento `Consulta.htm`:

```
<html><body>
<h1>Ricerca bibliografica</h1>
<form method="POST" action="Accedi.asp">
<table border="0">
<tr>
      <td>Autore:</td>
      <td><input type="text" name="cAutore"></td>
      <td><input type="checkbox" name="cAutore2">ricerca interna al campo</td>
      <td rowspan="2" valign="bottom"><input type="submit" value="Cerca!"></td>
</tr><tr>
      <td>Titolo:</td>
      <td><input type="text" name="cTitolo"></td>
      <td><input type="checkbox" name="cTitolo2">ricerca interna al campo</td>
</Tr></table>
</form>
</body></html>
```

Il documento `Accedi.asp`.

```
<html><body>
<h1>Risultato della ricerca</h1>
 ---------------------- Acquisizione dei parametri dall'interfaccia utente. -------------------------
<%
lAutore=Trim(Request.Form("cAutore")) 'elimina gli eventuali spazi iniziali e finali
lAutore2=Request.Form("cAutore2") 'acquisisce il valore del checkbox
lTitolo=Trim(Request.Form("cTitolo"))
lTitolo2=Request.Form("cTitolo2")
if lAutore="" And lTitolo ="" Then ' se i campi sono vuoti non inizia la ricerca
  Response.Write "Indicare una condizione di ricerca"
  Response.Write "</Body></Html>"
  Response.End ' il metodo End dell'oggetto Response termina l'esecuzione dello script.
```

```
End If
```
---------- Composizione del comando SQL. ----------
```
w="%"  'carattere jolly per VBS
s=""  'inizialmente la stringa SQL è vuota.
if lAutore<>"" Then
      if lAutore2="on" Then
            s="Autore Like'" & w & lAutore & w & "'"
      Else
            s="Autore LIKE '" & lAutore & w & "'"
      End If
End If
If lTitolo <> "" Then
  if s <> "" Then s = s & " AND "
    if lTitolo2 ="on" Then ‘ on è il valore di un checkbox selezionato
      s=s & "Titolo Like '" & w & lTitolo & w & "'"
    Else
      s=s & "Titolo Like '" & lTitolo & w & "'"
    End if
  End if
s="Select * From TabBiblio Where ("& s & ");"
```
‘ a questo punto con l’istruzione seguente `Response.Write s` è possibile verificare la correttezza della query.
---------- Interrogazione del Data base. ----------
```
Dim oConn
Dim oReco
DBtoOpen = Server.MapPath("LibriWeb.mdb")
set oConn = Server.CreateObject("ADODB.Connection")
oConn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & DBtoOpen
set oReco = Server.CreateObject("ADODB.Recordset")
oReco.Open s, oConn
```
---------- Preparazione e invio del risultato all’interfaccia utente. ----------
```
if oReco.Eof Then
Response.write "Nessun record trovato"
Response.write "</body></Html>"
Response.End
End If
linesep="<p>"
While not oReco.Eof
Response.write oReco("Autore") & ", " & oReco("Titolo") & " - " &
oReco("Commenti") & lineSep
oReco.MoveNext
Wend
```
---------- Rilascio dello spazio di memoria occupato dal oggetti del Server. ----------
```
oReco.Close: Set oReco=Nothing
oConn.Close: Set oConn = Nothing
%>
</body></html>
```

Esercizio 6.

Preparazione del data base.

In MS-Access creare un data base nuovo dal nome *Libri* e salvarlo in una sottocartella di Inetpub\wwwroot. Nel data base creare una nuova tabella con la seguente struttura:

Nome Campo	Tipo dati	Descrizione
Autore	Testo	
Titolo	Testo	
Commenti	Memo	

Salvare la tabella con il nome *TabBiblio*. Aprirla e inserire un certa quantità di record di prova.

La pagina lato client: `Consulta.htm`:

```
<html><body>
<h1>Ricerca bibliografica</h1>
<form method="POST" action="Accedi.asp">
<table border="0">
<tr>
      <td>Autore:</td>
      <td><input type="text" name="cAutore"></td>
      <td><input type="checkbox" name="cAutore2">ricerca interna al campo</td>
      <td rowspan="2" valign="bottom"><input type="submit" value="Cerca!"></td>
</tr><tr>
      <td>Titolo:</td>
      <td><input type="text" name="cTitolo"></td>
      <td><input type="checkbox" name="cTitolo2">ricerca interna al campo</td>
</Tr></table>
</form>
</body></html>
```

La pagina **`Accedi.asp`**

```
<%
Function creaStringaSQL(xAutore, xAutore2, xTitolo, xTitolo2)
w="%"
s=""
if xAutore <> "" Then
      if xAutore2="on" Then
s="Autore Like'" & w & xAutore & w & "'"
Else
s="Autore LIKE '" & xAutore & w & "'"
End If
End If
If xTitolo <> "" Then
  if s <> "" Then s = s & " AND "
    if xTitolo2 ="on" Then
      s=s & "Titolo Like '" & w & xTitolo & w & "'"
    Else
      s=s & "Titolo Like '" & xTitolo & w & "'"
    End if
  End if
creaStringaSQL = s
End Function
%>

<html><body>
<h1>Risultato della ricerca</h1>
<%
lAutore=Trim(Request.Form("cAutore")) 'elimina gli eventuali spazi iniziali e finali
lAutore2=Request.Form("cAutore2") 'acquisisce il valore del checkbox
lTitolo=Trim(Request.Form("cTitolo"))
lTitolo2=Request.Form("cTitolo2")
if lAutore="" And lTitolo ="" Then
  Response.Write "Indicare una condizione di ricerca"
  Response.Write "</Body></Html>"
  Response.End
End If
lOrdine = request.form("cOrdine")
if lOrdine = "" then lOrdine = "Autore"
```

```
s = creaStringaSQL(lAutore, lAutore2, lTitolo, lTitolo2)

s="Select * From TabBiblio Where ("& s & ") ORDER BY " & lOrdine & ";"
Dim oConn
Dim oReco
DBtoOpen = Server.MapPath("Libri.mdb")
set oConn = Server.CreateObject("ADODB.Connection")
oConn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & DBtoOpen
set oReco = Server.CreateObject("ADODB.Recordset")
oReco.Open s, oConn
if oReco.Eof Then
Response.write "Nessun record trovato"
Response.write "</body></Html>"
Response.End
End If
Response.write "<form method = 'post' action='Accedi.asp'>"
Response.write "Ordina per: <select name= 'cOrdine'>"
A = Array("Autore","Autore DESC","Titolo","Titolo DESC")
For i = 0 to 3
Response.write "<option value='" & a(i) & "'>" & a(i) & "</option>"
Next
Response.write "</select>"
Response.write "<input type='hidden' name='cAutore' value='" & lAutore & "'>"
Response.write "<input type='hidden' name='cAutore2' value='" & lAutore2 & "'>"
Response.write "<input type='hidden' name='cTitolo' value='" & lTitolo & "'>"
Response.write "<input type='hidden' name='cTitolo2' value='" & lTitolo2 & "'>"
Response.write "<input type='submit' value='esegui'>"
Response.write "</form>"
linesep="<p>"
While not oReco.Eof
Response.write oReco("Autore") & ", " & oReco("Titolo") & " - " &
oReco("Commenti") & lineSep
oReco.MoveNext
Wend
oReco.Close: Set oReco=Nothing
oConn.Close: Set oConn = Nothing
%>
</body></html>
```

Esercizio 7

Avanzamento del dialogo tra client e server.

Il server si comporta solo da *dispensatore* di pagine Web. Per consentire ad un utente di raffinare la ricerca occorre memorizzare i dati precedenti. Il server potrebbe spedire, accanto al nuovo form, anche il form contenente i dati ricevuti precedentemente dal client, e quindi l'utente li rispedirebbe insieme ai nuovi dati. In pratica i dati, per non essere reinseriti dall'utente, viaggerebbero ripetutamente tra client e server.

Ad esempio se il risultato della ricerca è piuttosto ampio, il server potrebbe inviare al client solo una pagina, consentendo però all'utente di richiedere l'invio di ulteriori pagine (come avviene con i risultati prodotti da un motore di ricerca). In questo caso il server deve conoscere, non solo il numero di pagina desiderato dall'utente, ma anche da quale recordset estrarre la pagina.

La pagina ASP viene modificata per inviare all'utente, oltre al risultato della ricerca, anche quattro link di navigazione:

`Pagina Num di tot - altre pagine:` Prima Precedente Successiva Ultima

Per suddividere il recordset in pagine, ADODB offre i seguenti metodi e proprietà:

`PageSize` - Numero di record da includere in una pagina

`PageCount` - Numero di pagine in cui è suddiviso il recordset, in conseguenza del PageSize scelto.

`AbsolutePage` – Numero di pagina da raggiungere.

Il documento `Accedi.asp`. diventa il seguente:

```
<html><body>
<h1>Risultato della ricerca</h1>
<%
---------- Acquisizione dei parametri dall'interfaccia utente. ----------
    I parametri potrebbero pervenire dal form, avente il metodo POST, o da uno dei link di scorrimento tra le pagine.
If Request.ServerVariables("REQUEST_METHOD")="POST" Then
  lAutore=Trim(Request.Form("cAutore"))
  lAutore2=Request.Form("cAutore2") 'acquisisce il valore del checkbox
  lTitolo=Trim(Request.Form("cTitolo"))
  lTitolo2=Request.Form("cTitolo2")
  if lAutore="" And lTitolo ="" Then
    Response.Write "Indicare una condizione di ricerca"
    Response.Write "</Body></Html>"
    Response.End
  End If
  lPagina = 1 ' se i dati provengono dal form si mostra solo la prima pagina
Else ' Se il metodo non è POST i parametri vengono prelevati dalla parte extra dell'indirizzo
  lAutore = Request.QueryString("cAutore")
  lAutore2= Request.QueryString("cAutore2")
  lTitolo = Request.QueryString("cTitolo")
  lTitolo2= Request.QueryString("cTitolo2")
  lPagina = Cint(Request.QueryString("cPagina"))
End If
 ---------- Composizione del comando SQL. ----------
s=""
if lAutore<>"" Then
  if lAutore2="on" Then
    s="Autore Like '%" & lAutore & "%'"
  Else
    s="Autore LIKE '" & lAutore & "%'"
  End If
End If
If lTitolo <> "" Then
  if s <> "" Then s = s & " AND "
    if lTitolo2 ="on" Then
      s=s & "Titolo Like '%" & lTitolo & "%'"
    Else
      s=s & "Titolo Like '" & lTitolo & "%'"
    End if
  End if
s="Select * From TabBiblio Where ("& s & ");"
    ---------- Interrogazione del Data base. ----------
Dim oConn
Dim oReco
DBtoOpen = Server.MapPath("LibriWeb.mdb")
set oConn = Server.CreateObject("ADODB.Connection")
```

```
oConn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & DBtoOpen
set oReco = Server.CreateObject("ADODB.Recordset")
oReco.Open s, oConn, 3
```

Il comando di apertura del recordset è stato specificato con un terzo parametro (il valore 3) che indica la modalità di consultazione del recordset. Il valore di default usato nell'esercizio precedente consentiva una lettura sequenziale, dal primo verso l'ultimo record, senza possibilità di ritornare indietro. La modalità 3 invece comprende la possibilità di consultare il recordset in entrambe le direzioni.

---------- Preparazione e invio del risultato all'interfaccia utente. ----------

```
if oReco.Eof Then
  Response.write "Nessun record trovato"
  Response.write "</body></Html>"
  Response.End
End If
RecordPerPagina = 5 ‘ si decide di mostrare 5 record per pagina
OReco.PageSize = RecordPerPagina
NumeroPagine = oReco.PageCount ‘ si legge il numero totale di pagine risultante
‘ composizione della stringa che segue l'indirizzo associato al link.
ExtraURL = ""
If lAutore <> "" Then
  ExtraURL = "cAutore=" & Server.URLEncode(lAutore)
  If lAutore2 ="on" Then ExtraURL = ExtraURL & "&cAutore2=on"
End If
If lTitolo <> "" Then
  If ExtraURL <> "" Then ExtraURL = ExtraURL & "&"
  ExtraURL = ExtraURL & "cTitolo" & Server.URLEncode(lTitolo)
  If lTitolo2 = "on" Then ExtraURL = ExtraURL & "&cTitolo2=on"
End If
‘ il link richiama questo documento asp ripetendogli i parametri di ricerca, ma specificando la pagina da mostrare
u = "Accedi.asp?" & extraURL & "&cPagina="
Response.Write "Pagina " & lPagina & "/" & NumeroPagine & " "
If lPagina > 1 Then
  Response.Write "<A Href='" & u & "1'>Prima Pagina</a>" & " "
  Response.Write "<A Href='" & u & (lPagina-1) & "'>Precedente</A>" & " "
End If
If lPagina < numeroPagine Then
  Response.Write "<A Href='" & u & (lPagina+1) & "'>Successiva</A>" & "nbsp;"
  Response.Write "<A Href='" & u & numeroPagine & "'>Ultima</A>"
End If
Response.Write "<HR>"
linesep="<p>"
oReco.AbsolutePage = lPagina
For i=1 To recordPerPagina
Response.write oReco("Autore") & ", " & oReco("Titolo") & " - " &
oReco("Commenti") & lineSep
oReco.MoveNext
If oReco.EOF Then Exit For
Next

oReco.Close: Set oReco = Nothing
oConn.Close: Set oConn = Nothing
%>
</body></html>
```

Esercizio 8.

I Cookie

Un *Cookie* è un file di testo che il server invia al client nel pacchetto di risposta http. Il client memorizza il cookie sul proprio disco e lo allega ad ogni pacchetto *Request* inviato al server da cui lo aveva ricevuto. In questo modo il *cookie* è usato per mantenere un'informazione relativa allo stato di avanzamento del dialogo. Il server scrive nel cookie con un comando della forma:

```
Response.Cookies(Nome cookie)(nome variabile)=Valore
```

Dove `Nome cookie` e `nome variabile` sono due stringhe (racchiuse quindi tra virgolette) che rappresentano rispettivamente il nome del file a cui associare il cookie e il nome della variabile da inserire nella collezione Cookies. Il server legge dal cookie con un comando della forma

```
Valore = Request.Cookies(Nome cookie)(nome variabile)
```

Il cookie esiste per la durata del collegamento, ma se contiene l'attributo `Expires` viene conservato fino alla data specificata, come nel seguente esempio:

```
Response.Cookies("miosito").Expires=#December 31, 2002#
```

Così, quando si ricollega al server, l'utente viene riportato allo stato in cui si trovava durante la sua ultima visita al sito. L'inconveniente dei cookie è che tra le opzioni del browser è prevista la possibilità di disabilitare la memorizzazione dei cookie.

Per gestire la consultazione del recordset, memorizzando nel cookie le condizioni di ricerca, il documento `Accedi.asp` viene scomposto in due parti. La prima parte prepara un cookie poi, tramite il metodo `Redirect` dell'oggetto `Response`, richiama la seconda parte. Il metodo `Redirect` allega al pacchetto di risposta http un indirizzo a cui il client è invitato a collegarsi. Come conseguenza il cookie viene spedito al client, che lo memorizza, e questo lo rispedisce nuovamente al server nel momento in cui si collega alla seconda parte dell'applicazione.

Il documento `Consulta.htm` contenente il form resta invariato, il documento `Accedi.asp` è il seguente (i cookies devono essere usati prima di qualsiasi tag html):

```
<%
lAutore=Trim(Request.Form("cAutore"))
lAutore2=Request.Form("cAutore2")
lTitolo=Trim(Request.Form("cTitolo"))
lTitolo2=Request.Form("cTitolo2")
if lAutore="" And lTitolo ="" Then
  Response.Write "Indicare una condizione di ricerca"
  Response.Write "</Body></Html>"
  Response.End
End If
Response.Cookies("RicBiblio")("Autore") = lAutore
Response.Cookies("RicBiblio")("Titolo") = lTitolo
Response.Cookies("RicBiblio")("Autore2") = lAutore2
Response.Cookies("RicBiblio")("Titolo2") = lTitolo2
Response.Redirect "Accedi2.asp?cPagina=1"
%>
```

Il documento `Accedi2.asp` è:

```
<html><body>
<h1>Risultato della ricerca</h1>
<%
  lAutore = Request.Cookies("RicBiblio")("Autore")
  lAutore2= Request.Cookies("RicBiblio")("Autore2")
  lTitolo = Request.Cookies("RicBiblio")("Titolo")
  lTitolo2= Request.Cookies("RicBiblio")("Titolo2")
  lPagina = Cint(Request.QueryString("cPagina"))
        :
```

La parte relativa alla composizione del comando SQL e quella di connessione al DB restano invariate. Viene modificata la sezione relativa alla formazione dei link:

```
        :
RecordPerPagina = 5
OReco.PageSize = RecordPerPagina
NumeroPagine = oReco.PageCount
kk = "accedi2.asp?cPagina="
Response.Write "Pagina " & lPagina & "/" & NumeroPagine & " "
If lPagina > 1 Then
  Response.Write "<A Href='" & kk & "1'>Prima Pagina</a>" & " "
  Response.Write "<A Href='" & kk & (lPagina-1) & "'>Precedente</A>" & " "
End If
If lPagina < numeroPagine Then
  Response.Write "<A Href='" & kk & (lPagina+1) & "'>Successiva</A>" & "nbsp;"
  Response.Write "<A Href='" & kk & numeroPagine & "'>Ultima</A>"
End If
Response.Write "<HR>"
linesep="<p>"
Ecc ...
%>
</body></html>
```

Esercizio 9.

L'oggetto Session e l'oggetto Application di ASP.

Nell'ambito di ASP un'applicazione è costituita dai documenti contenuti in una cartella o nelle sue sottocartelle. Quando un utente si collega al server richiedendo una pagina contenuta in una cartella, il server crea un oggetto `Application`. Questo oggetto `Application` è visibile a tutti i successivi utenti che accedono alle pagine della stessa cartella o delle sue sottocartelle.

Un oggetto `Session` è creato nel momento in cui un utente si collega a un server. Nella pratica, per una nuova richiesta http, il server genera un identificativo di utente che viene trasferito tra server e client in ogni pacchetto `Response` e `Request`. L'identificativo di sessione vale, per default, al massimo 20 minuti senza essere utilizzato, dopo di che scade. L'oggetto `Session` può essere eliminato dalla memoria del server anche con il metodo `Session.Abandon`, oppure specificando un diverso intervallo di esistenza nella proprietà `Session.Timeout`.

In entrambi gli oggetti si possono memorizzare coppie `(NomeDiVariabile)=Valore`, che quindi sono globali per l'applicazione, cioè per tutti gli utenti di una stessa cartella, o globali per tutte le pagine richieste da uno stesso utente.

`Memoria.htm` chiede all'utente di assegnare un valore a una variabile:

```
<html> <body>
<form method = "post" action="sessione.asp">
Memorizza un valore per la tua sessione: <input type="test" name="Ricordo">
<input type="submit" value = " Ok ">
</form>
</html> </body>
```

Il documento **`Sessione.asp`** richiamato dal form memorizza il valore prelevato dalla casella di testo del form:

```
<%
Session("variabilediStato")=Request.Form("Ricordo")
Session.Timeout = 1
Response.Redirect "Mostra.asp"
%>
```

`Mostra.asp` visualizza il valore della casella di testo accessibile da un altro documento aperto dall'utente.

```
<html> <Body>
Valore memorizzato: <%= Session("variabiledistato") %>
<hr>
<A href="Mostra.asp">Ricarica la Pagina</A>
<A href="Fine.asp">Chiudi la sessione e ricarica la pagina</A>
</body></html>
```

`Fine.asp` chiude la sessione:

```
<%
Session.Abandon
Response.Redirect "Mostra.asp"
%>
```

Il file global.asa.

Per ogni applicazione (quindi all'interno di ogni cartella di pagine Web) si può inserire un file denominato `global.asa`, adibito a contenere i gestori di evento. Gli oggetti `Session` e `Application` posseggono i gestori di evento `Session_OnStart`, `Session_OnEnd`, `Application_OnStart`, `Application_OnEnd`.

Ad esempio il gestore dell'evento OnStart dell'oggetto Session potrebbe contenere il comando di apertura del database, che in tal modo resta valido per tutta la durata della sessione utente, e viceversa il gestore di evento OnEnd potrebbe contenere i comandi per il rilascio degli oggetti DataBase e RecordSet.

Nella cartella di un'applicazione registrare il seguente file `global.asa`, in cui ci sono i gestori di evento che nel loro insieme hanno lo scopo di conteggiare il numero di utenti collegati contemporaneamente all'applicazione:

`Sub Application_OnStart`
`Application("numUtenti")=0`
`End Sub`
	Il gestore di evento `Application_OnStart` memorizza una variabile globale (accessibile a tutti gli utenti di quella applicazione) nell'oggetto `Application`, la denomina `numUtenti` e le assegna il valore iniziale 0.

`Sub Session_OnStart`
`Application("numUtenti") = _`
`Application("numUtenti") + 1`
`End Sub`
	Per ogni utente collegato (compreso il primo che ha avviato l'applicazione) si incrementa la variabile `numUtenti`.

`Sub Session_OnEnd`
`Application("numUtenti") = _`
`Application("numUtenti") – 1`
`End Sub`
	Per ogni utente che si scollega (con `Session.Abandon` o con lo scadere del suo Timeout) la variabile `numUtenti` si decrementa.

Per leggere il numero di utenti collegati all'applicazione si deve solo includere nei documenti della cartella la linea:

`Response.write Application("numUtenti") & "utenti connessi"`

Il primo utente che si collega riceverà: `1 utenti connessi` e non verrà informato che altri utenti si sono connessi fino a che non aggiorna la pagina.

Esercizio 10.

Scrittura nei data base: Cancellazione di record.

Le operazioni ammesse sui record di un data base sono essenzialmente quattro: Ricerca, Cancellazione, Aggiunta e Modifica. La ricerca di record è un'operazione di sola lettura che non comporta l'adozione di particolari precauzioni. Per le altre operazioni invece il programmatore deve prevedere il caso di accesso concorrente a un record.

Per svolgere gli esempi che seguono copiare la tabella `TabBiblio` del data base di esempio nella nuova tabella `Bibliobak`, in modo da ripristinare la tabella originaria dopo ogni operazione di modifica. Il seguente frammento di esempio cancella dalla tabella `TabBiblio` un insieme di record, identificati sulla base di una condizione (del tipo `Autore LIKE 'xx%'` dove xx è una stringa):

```
Set oConn = Server.CreateObject("ADODB.Connection")
oConn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source" _
& Server.MapPath("libriWeb.mdb")
oConn.Execute "DELETE * FROM TabBiblio WHERE (Autore LIKE '"& xx & "%');"
oConn.C1ose: Set oConn = Nothing
```

In questo esempio si è usato il metodo Execute per eseguire un comando SQL.

Una seconda tecnica di cancellazione permette di ricorrere a un recordset. La condizione con cui aprire il recordset identifica esattamente i record che si intendono cancellare, così che occorrerà solo scandire l'intero recordset e cancellarne tutti i record:

```
Set oConn = Server.Createobject("ADODB.Connection")
8et oReco = Server.Createobject("ADODB.Recordset")
oconn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source= " _
& Server.MapPath("LibriWeb.mdb")
oReco.Open "8ELECT * FROM TabBiblio WHERE (Autore LIKE '" & xx & "%');", _
oConn, 1, 2
While Not oReco.EOF
OReco.Delete
oReco.MoveNext
Wend
oReco.Close: Set oReco = Nothing
oconn.Close: Set oConn = Nothing
```

Un dettaglio importante riguarda le opzioni usate nel metodo Open del recordset:

```
recordset.Open istruzioneSQL, connessione, cursorType, lockType
```

Il metodo Open con cui viene aperto un recordset consente di specificare alcuni parametri opzionali: il *tipo di cursore*, con cui si determina quella che potrebbe essere definita la strategia di accesso ai dati presenti nel recordset, e il *tipo di lock* (LockType), che stabilisce le condizioni di aggiornamento dei dati in caso di accesso concorrente.

Il problema generale relativo al *locking* dei dati attiene alle varie strategie finalizzate a garantire la correttezza del risultato delle operazioni di scrittura che diversi utenti potrebbero compiere sul db, per evitare, per esempio, che un utente cerchi di modificare il contenuto del campo di un record mentre un altro utente sta cancellando proprio quello stesso record.

Ecco i valori possibili per la proprietà LockType..

valore:	tipo:
1	*adLockReadonly*
2	*adLockPessimistic*
3	*adLockOptimistic*
4	*adLockBatchOptimistic*

Il primo corrisponde alla condizione di recordset ad accesso in sola lettura *(read only* appunto), e come tale non può essere usato nel caso di script in cui sia necessario anche scrivere nel recordset.

Il secondo e il terzo tipo di lock, detti rispettivamente "pessimistico" e "ottimistico", forniscono un servizio di *aggiornamento immediato*: nel caso pessimistico il lock viene attivato al momento in cui comincia la fase di modifica del recordset, mentre nel caso ottimistico il lock viene attivato solo durante l'esecuzione del metodo Update. Il bloccaggio pessimistico garantisce una migliore protezione in fase di scrittura ma mantiene il record bloccato per un periodo maggiore, aumentando la probabilità che altri utenti che tentano di intervenire sullo stesso record lo trovino appunto bloccato. Il contrario accade per il lock ottimistico.

La proprietà Status del recordset fornisce l'informazione sulle condizioni del record corrente dal punto di vista della sua aggiornabilità. Tra i valori possibili di questa costante troviamo `adRecOK` (record aggiornabile o record correttamente aggiornato), corrispondente alla costante 0. Un tipico frammento di codice che include la gestione del lock potrebbe dunque essere:

```
On Error Resume Next
While Not oReco.EOF
istruzioni per l'aggiornamento del record
oReco.Update
If oReco.Status <> 0 Then
istruzioni per la gestione della condizione di record in stato di lock
End If
oReco MoveNext
Wend
```

Il quarto tipo di lock, corrispondente alla costante adLockBatchOptimistic, fornisce un servizio di *aggiornamento differito*, in cui il metodo Update scrive non direttamente sul db ma in un'area di buffer temporanea, il cui contenuto viene trasferito al db eseguendo il metodo UpdateBatch, e può invece essere cancellato con Cance1Batch. Comunque questo tipo di bloccaggio introduce una notevole complessità di gestione dell'accesso concorrente.

Una variante dei metodi di cancellazione di record, appena più complessa ma anche più versatile, prevede invece che il recordset venga aperto sull'intera tabella, che ad esso *si applichi un filtro* corrispondente ai record da cancellare, e che quindi si scandisca per la cancellazione il recordset così filtrato:

```
Set oConn = Server.CreateObject("ADODB.Connection")
8et oReco = Server.CreateObject("ADODB.Recordset")
oConn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source="
& Server.MapPath("db.mdb")
oReco.Open "TabBib1io", oConn, 1, 2 'apri direttamente la tabella
oReco.Filter = "Autore LIKE '" & xx & "%'"
While Not oReco.EOF
      oReco.Delete
      oReco.MoveNext
Wend
OReco.C1ose: Set oReco = Nothing
OConn.Close: Set oConn = Nothing
```

Scrittura nei data base: Inserimento e modifica di record.

Creazione di un nuovo record assegnando un valore ai suoi campi:

```
INSERT INTO nometabella (campol, campo2, ...) VALUES (valorecampol, valorecampo2, ...);
```

per esempio:

```
INSERT INTO TabBiblio (Autore, Titolo, Commenti) VALUES ("Feyerabend P.", "Contro il metodo", "1975");
```

Modifica del valore dei campi di un record esistente:

```
   UPDATE nometabella SET campo = valore WHERE (condizione);
```

per esempio:

```
UPDATE TabBiblio SET Autore = "Popper K.R." WHERE (Autore="Popper K.");
```

Per quanto riguarda la soluzione basata sull'apertura di un recordset, due sono i metodi di particolare importanza per la scrittura in un db:

`recordset.AddNew` che crea un nuovo record nel recordset, e:

`recordset.Update` che trasferisce il contenuto del record corrente dal recordset al db, memorizzandolo in modo permanente nel db stesso.

L'esempio seguente applica i metodi appena descritti:

```
creazione di un nuovo record, soluzione con gestione mediante recordset
Set oConn = Server.CreateObject("ADODB.Connection")
oConn.Open      "Provider=Microsoft.Jet.OLEDB.4.0;      Data      Source="      &
Server.MapPath("db.mdb")
Set oReco = Server.CreateObject("ADODB.Recordmet")
oReco.Open "TabBiblio", oConn, 1, 2
oReco.AddNew
oReco("Autore")   = "Feyerabend P."
oReco("Titolo")   = "Contro il metodo"
oReco("Commenti") = "1975"
oReco.Update
oReco.Close: Set oReco = Nothing
oConn.Close: Set oConn = Nothing
```