Asp - Active Server Pages

Nicola Gessa

Introduzione

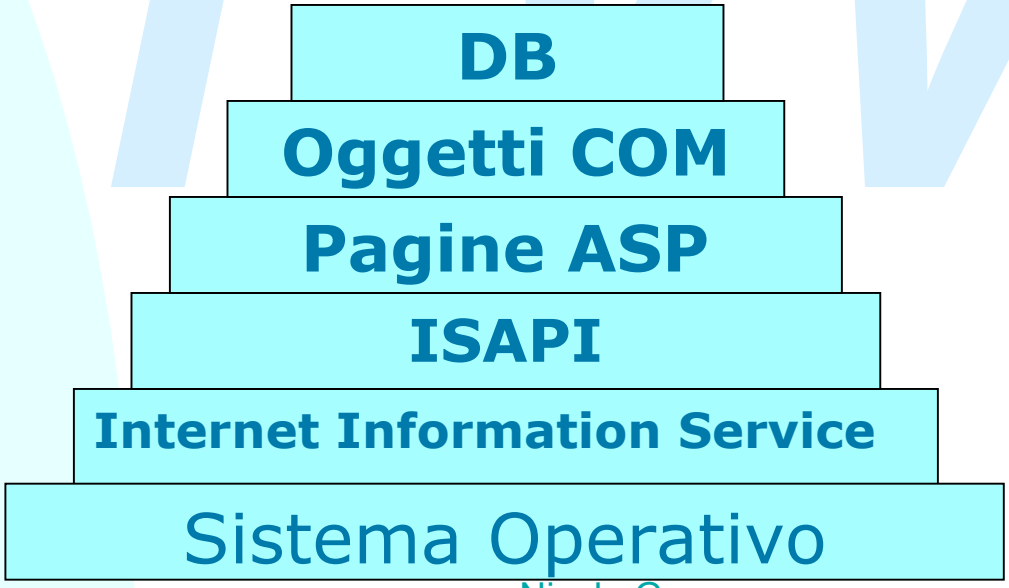
- Con l'acronimo ASP (Active Server Pages) si identifica NON un linguaggio di programmazione ma la tecnologia Microsoft per la creazione di pagine web dinamiche attraverso linguaggi di script come VBScript e Microsoft JScript.
- L'introduzione della tecnologia ASP è stata fatta con il rilascio della versione 3.0 di IIS nel 1997.
- ASP sfrutta non solo la connettivita del web server, ma attraverso oggetti COM si può interfacciare con tutte le risorse disponibili sul server e sfruttare tecnologie diverse in maniera trasparente.

Architettura di funzionamento

Il funzionamento e la programmazione delle pagine ASP si colloca all'interna dell'architettura Microsoft per la programmazione di pagine e applicazioni che rispondono alla richieste ricevute dal Web Server.

Tale architettura è costituita in maniera gerarchica e può comprendere ad esempio:

- IIS
- Active Server Pages
- Componenti COM (Component Object Model)
- DB



ASP - caratteristiche

- Le pagine ASP sono completamente integrate nei file HTML.
- Non necessitano di compilazione.
- Sono orientate agli oggetti.
- Usano componenti server ActiveX.
- Può essere utilizzato un qualunque linguaggio di scripting del quale sia installato sul web server lo *scripting engine*. ASP viene fornito con IIS insieme agli scripting engine per Microsoft VBScript e Microsoft Jscript.
- Un'applicazione ASP non è altro che una directory interna alla home page e per la quale è attivo il permesso di esecuzione.

Come funzionano le pagine ASP

Il procedimento attraverso il quale vengono create delle pagine dinamiche segue questa successione di operazioni:

- Il browser richiede una pagina .asp.
- Il web server avvia un interprete ASP specificando la pagina richiesta. Questo interprete è una DLL caricata dal web server.
- Il risultato dell'elaborazione viene restituito al web server.
- Il web server restituisce il codice HTML creato dall'elaborazione della pagine ASP al browser. Il browser non riceve mai il codice di scripting della pagina, che viene scartato una volta interpretato, ma solo il codice HTML.

Linguaggi di script

- ASP supporta in modo nativo due linguaggi di scripting, *VBScript* e *Jscript.*
- E' possibile modificare il linguaggio predefinito sia a livello di Web server, cioè per tutte le applicazioni ASP gestite da IIS, che a livello di singola applicazione, cioè per tutte le pagine che compongono un'applicazione.
- Tramite la direttiva @ *LANGUAGE* è possibile specificare il linguaggio da utilizzare all'interno di una determinata pagina ASP:

```
<%@ LANGUAGE = "JScript" %>
```

- All'interno di una pagina possono essere specificati linguaggi di scripting diversi attraverso il tag <SCRIPT>.

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server" >
Codice script
</SCRIPT>
```

Creare pagine ASP

- Una pagine ASP è un file di testo con estensione .asp che può contenere:
  - Testo
  - Tag HTML
  - Codice di script
- Il codice contenuto nelle pagine ASP consente l'uso di variabili, comandi, istruzioni di ciclo ecc, secondo la sintassi del linguaggio di script adottato.
- Il codice ASP viene identificato nel testo del file delimitandolo con i marcatori <% e %>.

`<%n=1 %>` assegna a n il valore 1;

- La pagina può contenere anche solo codice di script che generi l'HTML della pagina.

Come unire tag HTML e codice

1 - Inserendo il codice HTML in variabili

```
<% If Time >=#6:00:00AM# And Time<#12:00:00PM#
Then
      Saluto = "Buon giorno!"
Else
      Saluto="Ciao!"
End If %>
<%= Saluto%>
```

2 - Inserendo il codice HTML all'interno di blocchi di codice

```
<% If Time >=#6:00:00AM# And Time<#12:00:00PM#
Then%>
      Buon giorno!
<%Else      %>
      Ciao!
<%End If %>
```

Come unire tag HTML e codice

3 - Generando il codice HTML attraverso l'oggetto Response

```
<% If Time >=#6:00:00AM# And Time<#12:00:00PM#
Then
      Response.Write "Buon giorno!"
Else
      Response.Write "Ciao!"
End If %>
```

- Si possono utilizzare le variabili quando parti di codice HTML generato va ripetuto in più punti, mentre si inserisce direttamente il codice negli altri casi.

- La direttiva ASP <%= *espressione* %> mostra il valore della espressione e risulta equivalente ad usare il comando Response.Write.

Direttiva include

Permette di inserire in un file asp, il contenuto di un file esterno che può essere di testo, html, asp, grafica o qualsiasi altro file presente sul server.

```
<!--INCLUDE virtual | file="nomefile" -->
```

Va usata sempre al di fuori dei tag <%,%>, che delimitano gli script ASP.

- Le parole *virtual* e *file* servono ad indicare il tipo di path utilizzato per indicare il file da includere.
  - *virtual* indica un path che inizia con una virtual directory
  - *file* indica un path relativo rispetto al file che esegue l'include
- L'operazione di include viene eseguita prima di eseguire i comandi di script.

Commenti

- **VBScript**

Iniziano con l'apostrofo ‘

```
<% ‘questo linee
‘sono dei commenti VBscript%>
```

- **Jscript**

Iniziano con //

```
<% //questo linee
//sono dei commenti JScript%>
```

- non si possono inserire commenti nella direttiva <%= %>

`<%= nome ‘stampa il nome %>` NON funziona

Oggetti forniti con ASP

- ASP può utilizzare oggetti ActiveX per eseguire alcune operazioni o insiemi di operazioni. Queste componenti possono essere forniti da IIS, da terze parti o programmate in base alle esigenze e poi distribuite.
- Questi oggetti possono essere programmati usando un qualunque linguaggio che supporti il COM (Component Object Model) come C, C++, Java o Visual Basic.
- Per essere eseguiti sul Web Server gli oggetti ActiveX non devono contenere nessun elemento grafico.
- ASP mette a disposizione una lista di oggetti Built-In per la gestione dell'output, dell'input e delle operazioni che il client vuole o deve compiere sul server che vengono caricati in memoria al momento del loro utilizzo.
- Gli oggetti Built-In non devono essere precedentemente creati per il loro utilizzo, ma possono essere usati direttamente tramite i loro metodi e le loro proprietà.

Oggetti forniti con ASP

- Gli oggetti ActiveX sono riusabili: una volta creati possono essere copiati su sistemi windows differenti e registrati per poter usufruire delle loro funzionalità.
- Gli oggetti sono contenuti in librerie dinamiche di Windows (.dll) o in file eseguibili (.exe). Ciascuno di questi file può contenere uno o più oggetti.
- Per utilizzare un oggetto se ne deve creare un'istanza attraverso il metodo Server.CreateObject specificando il nome con cui questo è registrato (PROGID).

```
<% Set MyObj =
Server.CreateObject("HelloWorld.UserHandle") %>
Oppure usando il tag <OBJECT>
<OBJECT RUNAT= Server ID=MyObj
            PROGID="HelloWorld.UserHandle"></OBJECT>
```

ASP Built-In object

- ***Request Object*** - per ottenere tutte le informazioni che vengono passate con una richiesta HTTP.
- ***Response Object*** - per gestire le informazioni da spedire verso il client.
- ***Server Object*** - per accedere ai metodi e alle proprietà del server.
- ***Session Object*** - per registrare informazioni relative a sessioni.
- ***ADO Object*** - per l'uso della tecnologia ADO nell'acceso ai DB.
- ***Application Object*** - per gestione di dati comuni a tutte le istanze di esecuzione dello script.
- ***BrowserCap Object*** - per riconoscere il browser usato dall'utente.
- ***FileSystem Object*** - Permette di manipolare files e directories.

Uso degli Oggetti

Gli oggetti possono essere utilizzati sfruttando

- ***Metodi***: sono azioni che possono essere eseguite da un oggetto.

```
Response.Write "Ciao"
```

- ***Proprietà***: sono attributi associati agli oggetti e che possono essere letti o scritti.

```
Response.IsClientConnected
```

- ***Collezioni***: sono insiemi per definire nuovi dati e i loro valori associati a degli oggetti. Sono simili agli array, ma le loro dimensioni sono gestite automaticamente in seguito a inserimenti o cancellazioni di elementi. Gli elementi possono essere riferiti inoltre usando il loro nome o il loro indice.

```
<%Session.Contents("Firstname")="Mario"
Session.Contents("Lastname")="Rossi"%>
<%= Session.Contents("Lastname")%>
```

Uso delle collezioni - 1

- Per accedere a tutti gli elementi di una collezioni in VBScript si può usare il costrutto **For…Each**

```
<%Dim Item
For Each Item in Session.Contents
      Response.Write Session.Contents(Item) &
"<BR>"
Next%>
```

- …. o il costrutto **For..Next**

```
<%Dim Item
For Item  = 1 to Session.Contents.Count
      Response.Write Session.Contents(Item) &
"<BR>"
Next%>
```

Uso delle collezioni - 2

- Per accedere a tutti gli elementi di una collezioni in JScript si può usare il costrutto **for**

```
<% var item, numitems;
numitems = Session.Contents.Count;
for(item=1;item<=numitems;item++){
      Response.Write(Session.Contents(Item)) +
"<BR"
}%>
```

- .... o il costrutto **while**

```
<% var item, numitems;
numitems = Session.Contents.Count; item=1;
while(item<=numitems){
      Response.Write(Session.Contents(item)) +
"<BR";
      item++;
}%>
```

Uso di procedure

- Le procedure possono essere definite nello stesso file .asp che le richiama o in altri file inclusi successivamente con la direttiva #include.
- La definizione delle procedure sono solitamente inserite tra i tag <SCRIPT> e </SCRIPT>. Questi tag non devono contenere codice che non fa parte di procedure, poiché in questo caso l'ordine di esecuzione non è prevedibile.
- Non si può usare la direttiva <% = all'interno di una procedura, ma per generare dell'output si deve utilizzare l'oggetto response (Response.Write).
- Possono essere richiamate procedure definite in linguaggi diversi.

Esempio di chiamata di procedure

```
<%@LANGUAGE=VBSCRIPT%>
<HTML><BODY>
<%Call Echo%><BR>
<%Call PrintDate()%>
</BODY></HTML>
<%Sub Echo
        Response.Write "<P>Ciao Mondo!"
End Sub%>
<SCRIPT LANGUAGE=Jscript RUNAT=Server>
function PrintDate(){
        var x;
        x=NewDate()
        Response.Write("<P>Oggi è il ");
        Response.Write(x.toString())
}</SCRIPT>
```

Il livello sessione e applicazione

Nelle pagine ASP le informazioni di stato sono memorizzate a due livelli:

- **Sessione**: consente di identificare ogni visitatore e collezionare informazioni che consentono di tenere traccia delle scelte fatte dall'utente.

```
<% session("cognome")="Rossi"
session("nome")="Carlo"%>
```

- **Applicazione**: memorizza informazioni globali all'applicazione e visibili quindi da tutti gli utenti.

```
<% application.lock
application("valore")=application("valore")+1
application.unlock%>
```

Gestione delle Sessioni

- Le sessioni sono gestite tramite l'oggetto Session. L'ID di sessione è generato con un algoritmo che ne garantisce l'unicità.
- All'inizio della sessione il server invia al browser dell'utente l'ID della sua sessione usando i cookie.
- L'oggetto session fornisce un array associativo nel quale vengono memorizzate le informazioni di sessione
- Le sessioni hanno una durata fissata di default a 20 Minuti. Un lungo timeout di sessione porta al menteniemento di molte sessioni aperte contemporaneamente e quindi ad un forte utilizzazione delle risorse del web server.
- Le variabili memorizzate in questo oggetto non sono cancellate quando un utente passa da una pagina all'altra.
- Si può evitare l'uso delle sessioni in una pagina usando il tag

```
<%@ EnableSessionState=False%>
```

o impostando la relativa proprietà dell'applicazione ASP.

Gestione delle Sessioni

- L'oggetto Session fornisce dei metodi per terminare esplicitamente una sessione o fissarne un timeout.

```
//Fissare delle variabili di Sessione
<%Session.Contents("Nome")="Mario"%>
// Accedere a variabili di Sessione
<%= Session.Contents("Nome")%>
//Fissare un timeout ( es. 5 minuti)
<%Session.Timeout= 5 %>
//Terminare una sessione
<%Session.Abandon %>
```

Uso dei Cookie

ASP fornisce la collezione **Cookie** negli oggetti Request e Response per gestire i cookie.

- Per impostare un cookie si usa Response.Cookies. Se il cookie non esiste, viene creato.

```
<%Response.Cookies("IDUtente")="123456"%>
```

questa operazione va fatta prima di inviare il tag <HTML>.

- Si può impostare il timeout di un Cookie:

```
<%Response.Cookies("IDUtente").Expires="January 1, 2002"%>
```

- Si può impostare il path di in cui è valido il Cookie:

```
<%Response.Cookies("IDUtente").Path="/MyApp/User"%>
```

- Per leggere un cookie dal browser si usa l'oggetto Request

```
<%=Request.Cookies("IDUtente")%>
```

Inviare dati al client

- E' possibile fissare il tipo MIME dei contenuti inviati al server in maniera da informare il client su come leggere i dati ricevuti

```
<%Response.ContentType="text/plain"%>
```

- Si può reindirizzare il browser verso un nuovo URL utilizzando il metodo Redirect

```
<%If Session("SessionID")=0 Then Response.Redirect
"home.asp"
End If %>
```

Questo comando interrompe la successiva esecuzione dello script.

- Si può richiedere la generazione completa di una pagina prima che questa sia inviata al client fissando l'opzione Response.Buffer = true nel caso in seguito decidere se inviare o meno il contenuto.
- Si può forzare il browser a non memorizzare una pagina nella cache fissando Response.Expires =0.

Utilizzare le form HTML

Utilizzando l'oggetto Request si possono creare delle pagine che raccolgono dati inseriti tramite una form HTML

```
<FORM METHOD="POST" ACTION="myfile.asp">
<INPUT TYPE="text" NAME="firstname">
<INPUT TYPE="text" NAME="lastname">
<INPUT TYPE="text" NAME="age">
<INPUT TYPE="hidden" NAME="userstatus"
VALUE="new">
<INPUT TYPE="submit" NAME="enter">
</FORM>
```

Utilizzare le form HTML

I file .asp possono essere organizzati in 3 modi:

- Un file .htm può contenere una forma che richiama un file .asp
- Un file .asp può creare una form che richiama un'altro file .asp
- Un file .asp può creare una form che richiama lo stesso file .asp che ha creato la form

L'oggetto **Request** contiene due collection: **QueryString** e **Form** possono essere utilizzate per raccogliere i dati inviati con una form.

QueryString collection

- Viene usata per ricavare i valori spediti tramite query string.

Considerando la form vista precedentemente, usando il motodo GET e facendo Submit si può generare una richiesta del tipo

```
http://scripts/myfile.asp?firstname=Mario&lastname=Rossi&age=20&userstatus=new
```

- Per ricavare tali informazioni il codice asp può essere:

```
Salve sig. <%= Request.QueryString("lastname")%>
<%If Request.QueryString("userstatus")="new" then
      Response.Write "Questa è la tua prima visita a questo sito"
EndIf%>
```

Form Collection

Viene usata per ricavare i valori spediti tramite il metodo POST, che rispetto alla query string consente di inviare dati senza limiti di lunghezza.

La collection form funziona in maniera simile a collection QueryString

Es:

```
<%For  i= 1 to Request.Form.Count%>
<%=Request.Form(“names”)(i)%>
<%Next%>
```

Un esempio

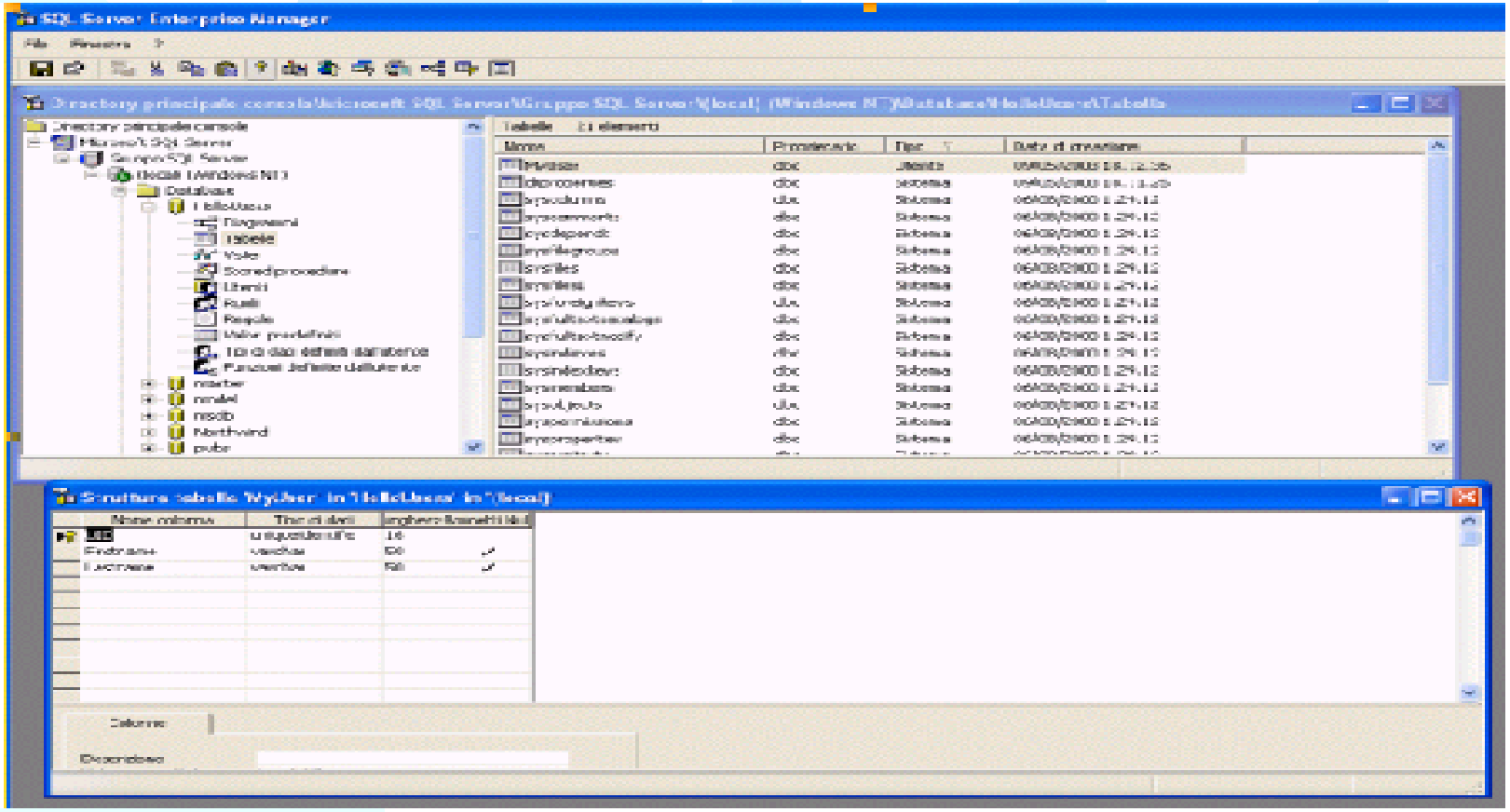
Si vuole creare una applicazione ASP che consenta a degli utenti di inserire il proprio nome da aggiungere ad una lista.

Per fare questo si deve:

- Predisporre un database per la registrazione dei dati ricevuti, quindi creare le tabelle con i campi necessari.
- Creare gli opportuni oggetti COM per eseguire le operazioni sul database.
- Creare una pagina HTML che consenta agli utenti di inserire i dati da registrare.
- Creare la pagina asp che ricevuti i dati via web, utilizza gli oggetti COM creati per registrare i dati sul database.

Predisporre il database

- Si crea il database HelloUser.
- Si crea la tabella MyUser con i campi Firstname, Lastname e ID.



Creazione degli oggetti COM

Si vuole creare una libreria di oggetti che semplifichi l'interazione della nostra pagina asp con il Database.

Questa libreria conterrà allora:

- Un oggetto che rappresenti i dati di un singolo utente: oggetto **User**.
- Un oggetto che rappresenti la lista completa degli utenti: oggetto **Users**. Questa lista sarà quindi una lista di oggetti di tipo User.
- Un oggetto per gestire le operazioni col database: aggiungere un nuovo utente o richiedere la lista degli utenti presenti. Queste due operazioni sono rappresentate da due motodi distinti dell'oggetto **UserHandle**.
- Negli esempi seguenti gli oggetti COM vengono programmati in VB.

Oggetti COM: l'oggetto User

```
Private mvarFirstname As String 'local copy
Private mvarLastname As String 'local copy

Public Property Let Lastname(ByVal vData As String)
    mvarLastname = vData
End Property

Public Property Get Lastname() As String
    Lastname = mvarLastname
End Property

Public Property Let Firstname(ByVal vData As String)
    mvarFirstname = vData
End Property

Public Property Get Firstname() As String
    Firstname = mvarFirstname
End Property
```

Oggetti COM: l'oggetto Users

```
'variabile locale per memorizzare la collezione
Private mCol As Collection
Public Function Add(Firstname As String, Lastname As String,
Optional sKey As String) As User
    'crea un nuovo oggetto user
    Dim objNewMember As User
    Set objNewMember = New User
    'fissa le proprietà ricevute come parametri dal metodo
    objNewMember.Firstname = Firstname
    objNewMember.Lastname = Lastname
    If Len(sKey) = 0 Then
        mCol.Add objNewMember
    Else
        mCol.Add objNewMember, sKey
    End If
    'ritorna l'oggetto creato
    Set Add = objNewMember
    Set objNewMember = Nothing
End Function
```

Oggetti COM: l'oggetto UserHandle

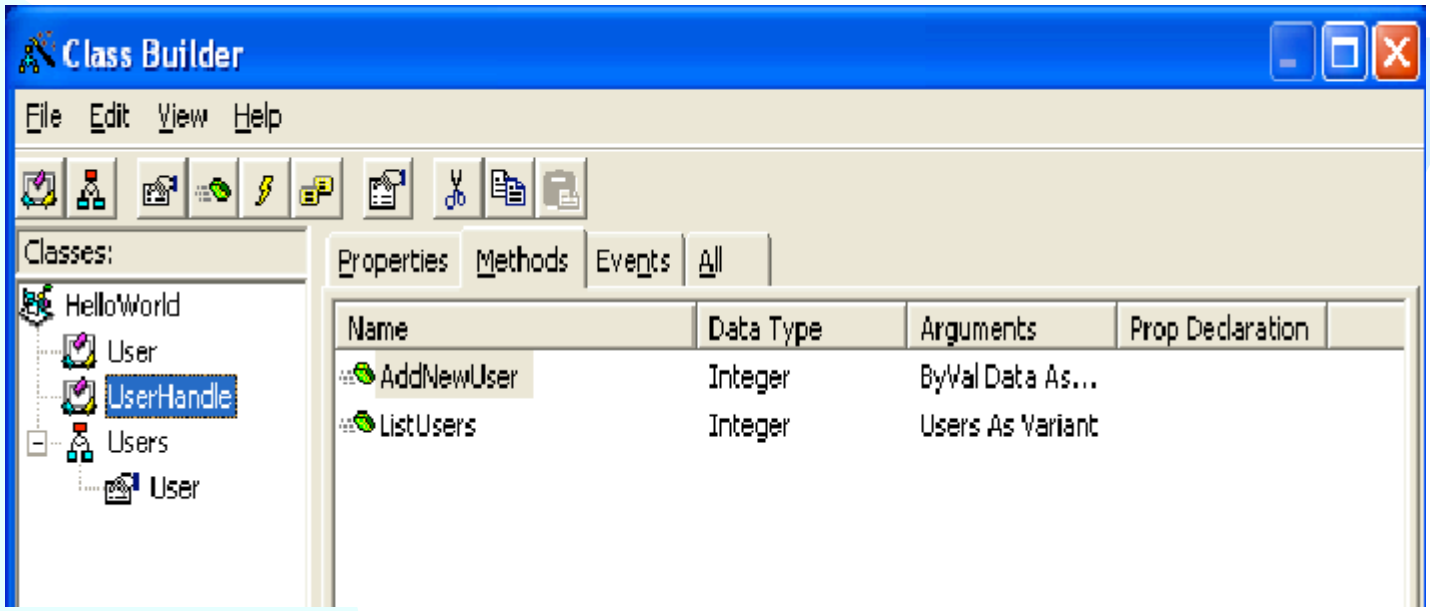
```
Public Function ListUsers(Users As Variant) As Integer
Dim objConn As Connection
Dim objRS As Recordset

    Set objConn = CreateObject("ADODB.Connection")
    objConn.Open "Provider=sqloledb;Data
Source=NAMTAR;Initial Catalog=HelloUsers;Integrated
Security=SSPI;"
    Set objRS = objConn.Execute("SELECT * FROM MyUser")
    Do While (Not objRS.EOF)
        Users.Add objRS("Firstname"), objRS("Lastname")
        objRS.MoveNext
    Loop
    ListUsers = 0
End Function
```

Oggetti COM: l'oggetto UserHandle

```
Public Function AddNewUser(ByVal Data As User) As
Integer
    Dim objConn As Connection
    Set objConn = CreateObject("ADODB.Connection")
    objConn.Open "Provider=sqloledb;Data
Source=NAMTAR;Initial Catalog=HelloUsers;Integrated
Security=SSPI;"
    objConn.Execute ("INSERT INTO MyUser
(Firstname,Lastname) VALUES ('" & Data.Firstname &
"','" & Data.Lastname & "')")
    objConn.Close
    Set objConn = Nothing
    AddNewUser = 0
End Function
```

La libreria HelloWorld.dll

La libreria **HelloWorld** è quindi così composta:



La compilazione del codice porta alla creazione del file **HelloWorld.dll** che può essere in seguito registrato nel sistema tramite il comando regsrv32 (**regsrv32 HelloWorld.dll**).In tal modo si inserisce la nuova dll nei registri di windows per poter fornire l'accesso agli oggetti che contiene a chi ne fa richiesta

La pagina asp - 1

```
<%@     Language=VBScript        %>
<% Response.Buffer = True%>
<%
Dim MyHelloWorld
Dim MyUsers
Dim MyUser
Dim MyError
   Set MyHelloWorld = Server.CreateObject("HelloWorld.UserHandle")
   Set MyUsers = Server.CreateObject("HelloWorld.Users")
         If (Request("Firstname") <> "") Then
               Set MyUser = CreateObject("HelloWorld.User")
               MyUser.Firstname = Request("Firstname")
               MyUser.LastName = Request("Lastname")
               MyError = MyHelloWorld.AddNewUser(MyUser)
               Set MyUser = Nothing
         End If
%>
```

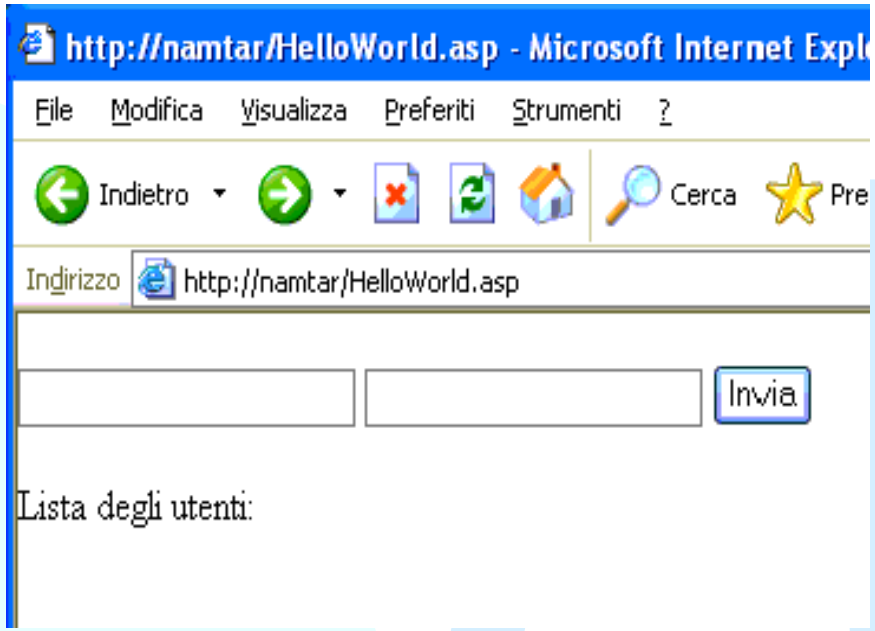
La pagina asp - 2

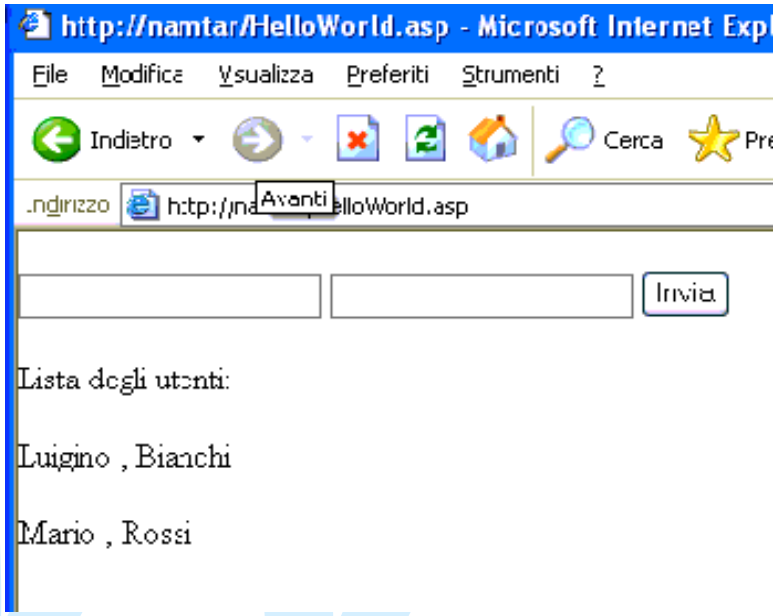
```
<HTML><HEAD></HEAD>
<BODY leftmargin="0" topmargin="0" marginwidth="0" marginheight="0" >
        <FORM action=HelloWorld.asp method=POST>
                <INPUT TYPE="text" name="Firstname">
                <INPUT TYPE="text" name="Lastname">
                <INPUT TYPE="submit" name="Invio" value="Invia"><BR>
        </FORM>
        <p>Lista degli utenti:<p>
        <% MyError = MyHelloWorld.ListUsers(MyUsers) %>
        <% For Each MyUser In MyUsers %>
        <p>
                <%= MyUser.Firstname %> , <%= MyUser.LastName %>
        </p><% Next %>
</BODY></HTML>
<%   Set MyHelloWorld = Nothing
Set MyUsers = Nothing  %>
```

Un paragone con un'applicazione VB

```
Private Sub Command2_Click()
Dim MyHelloWorld As UserHandle
Dim MyUsers As Users
Dim MyUser As User
Dim MyError As Integer
    List1.Clear
    Set MyHelloWorld = CreateObject("HelloWorld.UserHandle")
    Set MyUsers = CreateObject("HelloWorld.Users")
    MyError = MyHelloWorld.ListUsers(MyUsers)
    For Each MyUser In MyUsers
        List1.AddItem MyUser.Firstname & " " & MyUser.LastName
    Next
    Set MyHelloWorld = Nothing
    Set MyUser = Nothing
End Sub
```

Il risultato





Le due immagini mostrano la form di inserimento mostrata agli utenti e il risultato dopo l'avvenuto inserimento, con la lista di tutti i nomi inseriti nel Database

Il file Global.asa

Il file **Global.asa** serve per definire alcune funzioni che vengono avviate in alcune circostanze (avvio di un'applicazione, inizio di una nuova sessione).

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>

Sub Application_OnStart

End Sub

Sub Session_OnStart

End Sub

Sub Session_OnEnd

End Sub
</SCRIPT>
```

Link Utili

- http:// www.asp.net
- http:// msdn.microsoft.com
- http://www.w3schools.com/asp/
- http://www.aspitalia.com/