MANUALE DI ASP

Borsoft

2006

Borsoft

Basi di ASP

Introduzione

Cos'è ASP?

ASP è acronimo di Active Server Pages (pagine server attive), risulta chiaro dal nome che è una tecnologia lato server e che viene utilizzata per realizzare pagine che rispondano all'input dell'utente.

Per eseguire delle pagine ASP, abbiamo la necessità di installare un componente di windows chiamato IIS (Internet Information Services), questo componente contiene una libreria chiamata asp.dll che consente al server web di interpretare il codice asp, e fornire in output una pagina html.

Quando un browser richiede un file ASP, IIS passa le richiesta al motore ASP (asp.dll). Il motore ASP, linea dopo linea, legge il file ASP ed esegue gli script del file. Infine il file ASP viene restituito come un HTML.

Sintassi

L'ASP deve essere scritto tra due tag speciali che sono compresi dal server come tag di inizio e fine del codice. Questi tag sono <% e %>.

I server script vengono eseguiti sul server e possono contenere espressioni, istruzione, procedure o operatori validi per il linguaggio di scripting che si preferisce utilizzare.

la direttiva @language

Possiamo usare diversi linguaggi di scripting nei file ASP, il linguaggio di default è VBScript:

Per dichiarare esplicitamente il tipo di linguaggio che vogliamo utilizzare, possiamo usare la direttiva @languge

```
<%@ language="VBScript"%>
<html>
<body>
<%response.write("Il mio primo script!")%>
</body>
</html>
```

Variabili

Cos’è una variabile?

Una variabile è un "contenitore" di informazioni che si vuole memorizzare. Il valore di una variabile può cambiare durante l’esecuzione di uno script. Ci si può riferire ad una variabile utilizzandone il nome per leggerla oppure per impostarne il valore. In VBScript, tutte le variabili sono di tipo variant, questo significa che possono memorizzare diversi tipi di dati.

Regole per i nomi delle variabili:

Devono iniziare con una lettera

Non possono contenere dei punti (.)

Non possono superare i 255 caratteri

Dichiarare le variabili

Si possono dichiarare variabili con le istruzioni Dim, Public o Private. Come nell’esempio:

```
dim name
name=valore
```

Abbianmo creato una variabile chiamata name.

Si possono dichiarare delle variabili usandone il nome nello script, come nell’esempio:

```
name=valore
```

Il metodo appena visto non è consigliabile sia per ragioni di performance che di debug degli script (ad esempio se sbagliando a digitare scriviamo nae invece di name nello script, non è facile accorgersi dell'errore).

Per forzare la dichiarazione delle variabili possiamo usare l'istruzione Option Explicit. In questo caso siamo obbligati ad utilizzare le istruzioni dim, public o private. L'istruzione Option Explicit va all'inizio dello script:

```
option explicit
dim name
name=valore
```

Variabili Array

A volte vogliamo assegnare più di un valore ad una singola variabile. In questo caso creiamo una variabile che contiene una serie di valori. La dichiarazione della variabile include le parentesi tonde () che seguono il nome della variabile. Nel'esempio successivo dichiariamo un array contenente tre valori:

```
dim giorni(7)
```

Il numero mostrato tra parentesi è 2. Partiamo da zero, così abbiamo tre elementi. Questo è un array di dimensioni fisse. Si possono assegnare dei dati agli elementi dell'array nel modo seguente:

```
giorni(0)="domenica"
giorni(1)="lunedì"
giorni(2)="martedì"
giorni(2)="mercoledì"
giorni(2)="giovedì"
giorni(2)="venerdì"
giorni(2)="sabato"
```

possiamo leggere i dati utilizzando l'indice dell'elemento dell'array:

```
myDay=giorni(2)
```

Possiamo avere fino a 60 dimensioni in un array. Le dimensioni multiple sono dichiarate separando i valori con delle virgole. Nell'esempio dichiariamo una matrice con 5 righe e 7 colonne:

```
dim miaTabella(4, 6)
```

Per avere delle variabili accessibili a più file ASP, occorre dichiararle come variabili di sessione o di applicazione.

Variabili di sessione

Le variabili di sessione sono usatre per memorizzare informazioni su UN singolo utente, e sono disponibili a tutte le pagine di un'applicazione.

Variabili di applicazione

Le variabili di applicazione sono delle variabili disponibili a tutte le pagine in un'applicazione. Sono utilizzate per memorizzare informazioni circa TUTTI gli utenti in un'applicazione specifica.

Procedure e funzioni ASP

Le procedure sono dei blochhi di codice a cui è possibile passare dei valori che portanno essere manipolati o utilizzati per eseguire delle operazioni. Abbiamo due tipi di procedure: Sub e Funzioni.

Sub

E' una serie di istruzioni racchiuse tra le parole chiave Sub ed End Sub. Può leggere una serie di variabili passate come argomento e non restituisce un valore.

```
Sub mysub()
istruzioni
End Sub
```

oppure

```
Sub mysub(var1, var2, etc)
istruzioni
End Sub
```

Funzioni

E' una serie di istruzioni racchiuse tra le parole chiave Function ed End Function. Può leggere una serie di variabili passate come argomento e restituisce un valore assegnando ad una variabile il suo nome.

```
Function myFunction()
istruzioni
  ...
  myfunction=valore
End Function
```

oppure

```
Function myFunction(var1, var2, etc)
istruzioni
  ...
  myfunction=valore
End Function
```

Richiamare una Sub o una Function

Quando richiamiamo una funzione utilizziamo un codice del tipo:

```
nomeVariabile = nomeFunzione()
```

Chiamiamo la funzione nomeFunzione() e ne memorizziamo il valore di ritorno nella variabile chiamata "nomeVariabile".

Oppure possiamo fare una cosa del genere:

```
response.write "Il valore della funzione è: " & nomeFunzione ()
```

Quando richiamiamo una Sub, possiamo utilizzare l'istruzione Call e e mettere gli argomenti tra parentesi oppure omettere l'istruzione call e scrivere gli arcgomenti della funzione senza le parentesi:

```
call miaProcedura(var1, var2, etc)
```

Oppure si può omettere l'istruzione Call, come nell'esempio:

```
miaProcedura var1, var2, etc
```

Borsoft

Concetti avanzati

Global.asa

Il file Global.asa è un file opzionale che può contenere dichiarazioni di oggetti, variabili, metodi e può essere acceduto da ogni pagina dell'applicazione ASP.

Il file Global.asa

Il file Global.asa è un file opzionale che può contenere dichiarazioni di oggetti, variabili, metodi e può essere acceduto da ogni pagina dell'applicazione ASP. Tutti gli script (JavaScript, VBScript, JScript, PerlScript, etc.) possono essere usati nel file Global.asa.

Il file Global.asa può contenere solo le seguenti sezioni:

- Eventi Application
- Eventi Session
- Dichiarazioni <object>
- Dichiarazioni TypeLibrary
- La direttiva #include

Nota: Il file Global.asa deve essere memorizzato nella direcrory principale dell'applicazione ASP, ed ogni applicazione ASP può avere soltanto un file Global.asa.

Eventi nel Global.asa

Nel file Global.asa si può dir eagli oggetti application e session cosa fare quando l'applicazione o la sessione inizia o finisce. Il codice viene inserito in dei gestori di eventi.

Il file Global.asa contiene quattro tipi di eventi:

Application_OnStart – quando il PRIMO utente richiama la prima pagina dell'applicazione. Questo evento avviene quando il Web server viene riavviato o dopo che il Global.asa è modificato. L'evento "Session_OnStart" avviene immediatemente dopo.

Session_OnStart – tutte le volte che un NUOVO utente richiede la prima pagina in un'applicazione.

Session_OnEnd – l'evento avviene TUTTE le volte che un utente termina una sessione. U nutente termina una sessione quando non richiede pagine per un certo lasso di tempo (di default 20 minuti).

Application_OnEnd – avviene quando l'ultimo utente termina una sessione (tipicamente quando viene fermato il Web server.

Ecco un esempio di file Global.asa

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

Sub Application_OnStart
    'codice
End Sub

Sub Session_OnStart
    'codice
End Sub

Sub Session_OnEnd
    'codice
End Sub

Sub Application_OnEnd
    'codice
End Sub

</SCRIPT>
```

Nota: Non possiamo usare i delimitatori di script asp (<% e %>) per inserire scripts nel file Global.asa, dobbiamo inserire le subroutines all'interno di tag <script>.

Dichiarazioni <object>

E' possibile creare oggetti con visibilità di sessione o applicazione, utilizzando il tag <object> nel file Global.asa.

Nota: Il tag <object> deve stare al di guori di tag <script>

Sintassi

```
<object runat="server" scope="scope" id="id"{progid="progID"|classid="classID"}>
....
</object>
```

Parametro	Descrizione
scope	Imposta l'ambito dell'oggetto (Session o Application)
id	Specifica un identificatore unico per l'oggetto
ProgID	Un id associato all'id di classe. Il formato è [Fornitore.]Componente[.Versione]
ClassID	Specifica un id unico per un oggetto ProgID e ClassID devono essere specificati.

Esempi

Il primo esempio crea un oggetto di session chiamato "MyAd" usando il parametro ProgID:

```
<object runat="server" scope="session" id="MyAd" progid="MSWC.AdRotator"></object>
```

Il secondo esempio crea un oggetto di applicazione chiamato "MyConnection" utilizzando il parametro ClassID:

```
<object runat="server" scope="application" id="MyConnection" lassid=" Clsid:8AD3067A-
B3FC-11CF-A560-00A0C9081C21"></object>
```

Gli oggetti dichiarati nel Global.asa possono essere usati da qualsiasi script nell'applicazione:

GLOBAL.ASA:

```
<object runat="server" scope="session" id="MyAd"progid="MSWC.AdRotator"></object>
```

Facciamo riferimento all'oggetto in un file:

```
ASP FILE:
<%=MyAd.GetAdvertisement("/banners/adrot.txt")%>
```

Dichiarazioni TypeLibrary

Un TypeLibrary è un contenitore per file DLL corrispondenti ad un oggetto COM. Includendo una chiamata nel file Global.asa, le costanti dell'oggetto COM possono essere accedute, e gli errori possono essere riportati meglio nel codice ASP.

Sintassi

```
<!--METADATA TYPE="TypeLib" file=" filename" uuid="typelibraryuuid"
version="versionnumber" lcid="localeid"-->
```

Parametro	Descrizione
File	Specifica un percorso assoluto per un type library. Richiesto
Uuid	Specifica un identificatore univoco. Richiesto
version	Opzionale. Usato per identificare la versione
localeid	Opzionale.

Errori

Il server può restituire uno dei seguenti messaggi di errore:

Error Code	Descrizione
ASP 0222	Specifica di type library
ASP 0223	Type library
ASP 0224	Non è possible caricare la Type library
ASP 0225	Non è possible spostare il Type library

Restrizioni

Cosa è possibile includere nel Global.asa:

Non si può visualizzare del testo scritto nel Global.asa. Questo file non può visualizzare informazioni

Possiamo usare solo oggetti Server ed Application nelle subroutine Application_OnStart ed Application_OnEnd. Nella subrutine Session_OnEnd subroutine, possiamo usare oggetti Server, Application, e Session. Nella subrutine Session_OnStart subroutine possiamo utilizzare tutti gli oggetti built-in

Come usare le Subroutines

Il Global.asa è usato spesso per inizializzare le variabili.

L'esempio in basso mostra come determinare quando un utente è entrato nell'applicazione (o nel sito). Il tempo è memorizzato nella variabile di sessione chiamata "started", e tale valore può essere usato in qualsiasi pagina asp dell'applicazione:

```
<script language="vbscript" runat="server">
sub Session_OnStart
    Session("started")=now()
end sub
</script>
```

Il Global.asa può anche essere usato per controllare gli accessi alle pagine.

L'esempio in basso mostra come ridirezionare ogni nuovo visitatore ad una pagina chiamata "newpage.asp":

```
<script language="vbscript" runat="server">
sub Session_OnStart
    Response.Redirect("newpage.asp")
end sub
</script>
```

Ne Global.asa si possono includere anche delle funzioni.

Nell'esempio in basso, la subrutine the Application_OnStart viene richiamata quando il webserver viene avviato, quindi la subroutine Application_OnStart chiama un'altra subroutine "getcustomers". La "getcustomers" apre una connessione ad un database e recupera I record dalla tabella "customers" table. Il recordset è assegnato ad un aray, e può errere acceduto dalle pagine dell'applicazionesenza eseguire query sul database:

```
<script language="vbscript" runat="server">
sub Application_OnStart
    getcustomers
end sub
sub getcustomers
    set conn=Server.CreateObject("ADODB.Connection")
    conn.Provider="Microsoft.Jet.OLEDB.4.0"
    conn.Open "c:/webdata/northwind.mdb"
    set rs=conn.execute("select name from customers")
    Application("customers")=rs.GetRows
    rs.Close
    conn.Close
end sub
</script>
```

Esempio di file Global.asa

Nell'esempio seguente creeremo un file Global.asa che conta il numero di utenti collegati al sito.

Application_OnStart imposta la variabile "visitors" a zero quando viene avviato il server

La subroutine Session_OnStart aggiunge 1 alla variabile "visitors" ogni volta che un utente si collega

La subroutine The Session_OnEnd toglie 1 alla variabile "visitors" ogni volta che un utente lascia il sito

Il file Global.asa:

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

Sub Application_OnStart
    Application.Lock
    Application("collegati") = 0
    Application.UnLock
End Sub

Sub Session_OnStart
    Application.Lock
    Application("collegati") = Application("collegati") + 1
    Application.UnLock
End Sub

Sub Session_OnEnd
    Application.Lock
    Application("collegati") = Application("collegati") - 1
    Application.UnLock
End Sub

</SCRIPT>
```

Includes

Inclusioni server-side

Un inclusione lato server non è altro che una porzione di codice (ASP o anche HTML) che risiede in un file esterno e viene richiamata all'interno di un'altro file.

La direttiva #include

Si può inserire il contenuto di un file ASP dentro un altro prima che il server lo esegua, mediante la direttiva #include. La direttiva #include è usata per creare funzioni, headers, footers, o elementi che verranno riutilizzati in molte pagine.

Sintassi per l'inclusione dei file

Per includere un file in una pagina ASP, inserire la direttiva #include dentro dei tag <!-- e -->

La parole chiave File

Utilizzare la parola chiave file per indicare un percorso relativo. Un percorso relativo, inizia con la directory che contiene il file che richiama la direttiva

#include. Se abbiamo un file nella directory html ed il file "menu.inc" risiede nella directory fisica "bricks", occorre usare il seguente codice:

Nota:
Si può anche utilizzare la parola chiave file insieme alla sintassi (..\) per includere file da una directory di livello più alto ma NON con windows 2003!.

```
<!-- #include file ="bricks\menu.inc" -->
```

La parola chiave Virtual

Utiizzare la parola chiave virtual per indicare un percorso che inizia con una directory virtuale. Se il file chiamato "dbtools.inc" risiede in una directory virtuale chiamata /includes, il seguente codice inserisce il contenuto di "header.inc":

```
<!-- #include virtual ="/includes/dbtools.inc" -->
```

Forms ed User Input

Abbiamo due metodi per leggere i dati inviati da una Form. Questi sono: Request.QueryString e Request.Form l'utilizzo dell'uno o dell'altro dipende dal metodo con cui vengono inviati i dati del form.

Form di login

L'oggetto Request object può essere utilizzato per ricavare informazioni dalle form:

```
<form method="get" action="login.asp">
Username: <input type="text" name="user"><br />
Password: <input type="text" name="pwd"><br /><br />
<input type="submit" value="Submit">
</form>
```

Se utilizzamo il metodo get per inviare i dati, occorrerà leggerli tramite il metodo Request.QueryString.

Request.QueryString

Tramite il metodo get, le informazioni vengono inviate mediante la querystring, sul browser vedremo un indirizzo del tipo:

http://www.dominio.com/login.asp?user=morpX&pwd=55sGG7

Per leggere i parametri:

```
<%
user = Request.QueryString("user")
pwd = Request.QueryString("pwd")
%>
```

Request.Form

Il comando Request.Form è usato per ricavare informazioni da un form con method="post".

Per leggere i dati dovremo usare una sintassi del tipo:

```
<%
user = Request.Form("user")
pwd = Request.Form("pwd")
%>
```

SQL Injection

Quando leggiamo i dati da una form come quella di una login, siamo a rischio di SQL Injection.

L'SQL Injection è una procedura tramite la quale è possibile "iniettare" del codice SQL in modo illecito.

Esempio

Vediamo un esempio concreto. Supponiamo di avere una tabella Users così fatta:

Username	Password
user1	pass1
user2	pass2
user3	pass3

Ecco la form per il login

```
<form method="post" action="login.asp">
Username: <input type="text" name="user"><br />
Password: <input type="text" name="pwd"><br /><br />
<input type="submit" value="Submit">
</form>
```

Autenticazione

Per leggere i parametri:

```
<%
user = Request.Form("user")
pwd = Request.Form("pwd")

If Not checkUser(user,pwd)
  Response.Redirect("nonautorizzato.asp")
Else
  'utente autenticato
End If
```

```
function checkUser(user,pwd)

  'connessione al database...

  checkUser = false
  SQL = ""
  SQL = SQL & "Select * from utenti "
  SQL = SQL & " where "
  SQL = SQL & "  [username]='" & user & "' and "
  SQL = SQL & "  [password]='" & pwd & "'"

  Res = Connection.Execute(SQL)
  If not(Res.Eof) Then
    checkUser = true
  End If
  Res.Close

end function
%>
```

Ecco che entra in gioco la SQL Injection. se come username e password inseriamo il codice: a' or 'a' = 'a ecco come viene scritta la query che legge l'utente:

```
select * from utenti
where
[username] ='a' or 'a'='a' and
[password] ='a' or 'a'='a'
```

Questo perchè il carattere ' viene interpretato da ASP come carattere di fine stringa.

A questo punto risulta chiaro che la query restituisce il recordset completo di tutti gli utenti del database (essendo la where sempre verificata) e ci fa passare...

Come evitare l'SQL Injection

Ci sono diversi metodi per evitare l'SQL Injection, io vi mostro quello più immediato, e comunque di sicuro effetto.

Quando leggiamo i valori dalla querystring, basta sostituire il carattere di ' con due caratteri '' tramite l'istruzione Replace

```
<%
user = Request.Form("user")
pwd = Request.Form("pwd")

'per prevenire l'sql injection
user = Replace(user,"'","''")
pwd = Replace(user,pwd,"'","''")
%>
```

Il resto del codice non necessita modifiche.

Application

Un gruppo di pagine ASP che cooperano per uno stesso scopo è chiamato applicazione, esiste un oggetto chiamato application che seve a gestire alcune informazioni a livello di applicazione.

Application

Un'applicazione sul web può essere un grupo di file. I file ASP lavorano insieme per uno stesso scopo. L'oggetto application è usato per gestire questi file insieme. Quando parliamo di Application, dobbiamo pensare ad un oggetto che "vive" con l'appplicazione stessa, e non è legato al singolo utente bensì è comune a tutti.

Lock and Unlock

A volte è necessario "bloccare" l'applicazione per eseguire dei comandi, possiamo farlo tramite i metodi lock ed unlock (che seve a sbloccare l'applicazione)

vediamo come fare

```
<%
Application.Lock
'istruzioni
Application.Unlock
%>
```

Le variabili di Applicazione: Un esempio concreto

Per comprendere l'utilizzo delle variabili di applicazione, vi farò vedere come funziona il contatore di utenti collegati che potete vedere sulla homepage di morpheusweb.it

Per farlo ho utilizzato delle variabili di applicazione inizializzate e valorizzate nel file global.asa (esiste un capitolo del manuale apposta su questo particolare file)

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

Sub Application_OnStart
    Application.Lock
    Application("collegati") = 0
    Application.UnLock
End Sub

Sub Session_OnStart
    Application.Lock
    Application("collegati") = Application("collegati") + 1
    Application.UnLock
End Sub

Sub Session_OnEnd
    Application.Lock
    Application("collegati") = Application("collegati") – 1
    Application.UnLock
End Sub

</SCRIPT>
```

Ecco come funziona. Quando viene avviata l'applicazione (all'avvio del web) c'è una variabile di applicazione chiamata application("collegati") che viene inizializzata a zero, in quanto viene eseguito il metodo Application_OnStart. Quando un utente si collega crea una nuova sessione, che scatena l'evento Session_OnStart che incrementa la variabile di 1. In modo analogo quando un utente abbandona il sitola variabile di applicazione viene decrementata.

Per accedere alla variabile di sessione possiamo poi utilizzare il seguente codice:

```
Ci sono <%= Application("collegati") %> utenti collegati.
```

La Collection Contents

La collection Contents contiene tutte le variabili di applicazione, possiamo scorrerne i valori tramite un ciclo For Each (per chi volesse approfondire la sintassi del ciclo c'è un capitolo nel manuale di VBScript)

```
<%
Dim val
For Each val In Application.Contents
    Response.Write(val & "<br />")
Next
%>
```

Una proprietà importante è count, che restituisce il numero di elementi della collection. Come esempio vediamo un metodo analogo al precedente per visualizzare i valori della collection Contents:

```
<%
Dim cont
For cont=1 To Application.Contents.Count
    Response.Write(Application.Contents(cont) & "<br />")
Next
%>
```

Borsoft

Sessioni

La sessione è utilizzata per memorizzare informazioni sulla sessione utente. Le variabili memorizzate nell'oggetto Session tengono informazioni su un singolo utente, e sono disponibili a tutte le pagine dell'applicazione.

Sessioni

HTTP è un protocollo stateless,senza stato, non mantiene cioè informazioni su ciò che avviene quando navighiamo.

Se vogliamo creare delle pagine che "ricordino" delle informazione, dobbiamo memorizzarle da qualche parte. Un modo per farlo è utilizzare le sessioni.

Le sessioni utilizzano i cookie per memorizzare le informazioni, ma lo fanno in modo trasparente al programmatore.

L'oggetto Session è utilizzato per memorizzare o cambiare le informazioni circa la sessione di un singolo utente. Le variabili memorizzate nell'oggetto Session hanno delle informazioni sul singolo utente e sono disponibili per tutte le pagine dell'applicazione.

Timeout della sessione

Una sessione termina se un utente non ha richiesto pagine o fatto il refresh in una pagina dell'applicazione per un determinato lasso di tempo. Di default questo tempo è 20 minuti. Se si vuole cambiare l'intervallo di dafault, occorre impostare la proprità Timeout.

```
<%
Session.Timeout=60
%>
```

Per terminare una sessione immediatamente si può utilizzate il metodo Abandon:

```
<%
Session.Abandon
%>
```

Scrivere e leggere le variabili di sessione

L'esempio in basso spiega come memorizzare delle variabili di sessione. Imposteremo la variabile di sessione username a "Donald Duck" e la variabile age a "50":

```
<%
Session("username")="morphX"
Session("password")="SJj8saD"
%>
```

Queste due variabili possono essere recuperate in qualsiasi parte dell'applicazione:

```
<%
username = Session("username")
password = Session("password")
%>
```

Eliminare le variabili di sessione

La collezione Contents contiene tutte le variabili di sessione. Supponiamo ad esempio di voler fare un logout e pulire la sessione

```
<%
Sub logout()
    Session("username") = ""
    Session("password") = ""
End Sub
%>
```

Per rimuovere tutte le variabili di sessione si può utilizzare il metodo RemoveAll:

```
<%
Sub logout()
    Session.Contents.RemoveAll()
End Sub
%>
```

Potremmo anche utilizzare il metodo session.abandon per fare il logout.

Ciclare il contenuto della collezione Contents

La collezione Contents contiene tutte le variabili di sessione. Si può ciclarne il contenuto per vedere cosa c'è memorizzato:

```
<%
Session("username")="morphX"
Session("password")="SJj8saD"
dim counter
For Each counter in Session.Contents
    Response.Write(counter & "<br />")
Next
%>
```

Risultato:

Se non conosciamo il numero di elementi nella collection, possiamo utilizzare la proprietà Count:

```
<%
dim prop
Response.Write("Variabili di sessione: " & Session.Contents.Count)
For prop=1 toSession.Contents.Count
    Response.Write(Session.Contents(prop) & "<br />")
Next
%>
```

Cookies

Cos’è un cookie?

Un coockie è un file di piccole dimensioni che può essere inviato ad un client che si collega ad una nostra pagina web. Può essere utilizzato in diversi modi, ma lo scopo per cui viene utilizzato più spesso è quello di scrivere delle informazioni sulle preferenze dell'utente o comunque dei dati relativi all'utente che si collega alla pagina

Con ASP è possibili inviare e leggere i cookies.

Come scrivere e leggere un cookie

Per creare un cookie viene utilizzato il comando "Response.Cookies" scritto prima del tag <html>.

Nell’esempio in basso, cereeremo un cookie chiamato "nome" e gli assegneremo il valore "Carmelo", nel contempo impostiamo una data di scadenza, una data cioè dopo la quale il cookie verrà cancellato dal browser dell'utente.

```
<%
Response.Cookies("nome") = "Carmelo"
Response.Cookies("nome").Expires= #Dec 10,2006#
%>
```

Per leggere un cookie viene usato il comando"Request.Cookies".

```
<%
Dim nome
nome = Request.Cookies("nome")
If (nome <> "") Then
  Response.Write("Ciao, " & nome)
End if
%>
```

Un Cookie con delle chiavi

Se un cookie contiene una collezione di valori, diciamo che il cookie ha delle chiavi (Keys).

Nell’esempio in basso, creiamo un cookie "AuthUser" contenente informazioni sull’utente:

```
<%
Response.Cookies("AuthUser")("nome") = "Carmelo"
Response.Cookies("AuthUser")("username") = "morphX"
Response.Cookies("AuthUser")("password") = "HSu82Jh0"
Response.Cookies("AuthUser")("function") = "Admin"
%>
```

L'esempio visto può ad esempio rivelarsi utile per memorizzare le informazioni di accesso ad un'area risevata del vostro sito.

Error Handling

In ASP non abbiamo dei meccanismi nativi per la gestione degli errori come in altri linguaggi, ad esempio i blocchi try catch di C# o Java.

Esiste però un'istruzione che, se usata nel modo corretto ci consente di gestire gli errori a runtime evitando i classici messaggi di errore di IIS

L'istruzione in questione è

```
<%
On Error Resume Next
%>
```

La quale dice all'interprete ASP di continuare l'esecuzione ignorando l'errore, invece di fermare l'esecuzione e mostrare il messaggio di errore

Una volta detto all'interprete di ignorare l'errore, dobbiamo fare in modo di gestirlo e mostrare all'utente un messaggio. Una volta "trappato" l'errore occorre "gestirlo"

Esiste un oggetto "Err" che può essere utilizzato a tale scopo.

Questo oggetto ha diversi metodi e proprietà che possono essere usati per i nostri scopi.

Metodi

Metodo	Descrizione
Clear	Ripulisce l'oggetto Err, e deve essere usato una volta che abbiamo gestito l'eccezione
Raise	Solleva in modo esplicito un'eccezione

Proprietà

Proprietà	Descrizione
Description	Mostra la descrizione dell'errore
HelpContext	Permette di impostare o recuperare un id di contesto per l'errore
HelpFile	Permette di impostare o recuperare un file di hepl per l'errore
Number	Rappresenta il numero di errore vbscript
Source	Il codice sorgente che ha generato l'errore

Una volta visti metodi e proprietà, vediamo un esempio di gestione di errori in cui in caso di eccezione, inviamo una mail all'amministratore con la descrizione dell'errore e reindirizziamo l'utente in una pagina di spiegazione.

```
<%
On Error Resume Next
'codice...
'la riga che genera l'errore
Response.Write(10/0)

'Gestisco l'errore
If Err.Number <> 0 then
  NumeroErrore = Err.Number
  DescrizioneErrore = Err.Description
  Pagina = Request.ServerVariables("url")
  Call GestisciErrore(NumeroErrore, DescrizioneErrore, Pagina)
End If

'La procedura per la gestione dell'errore
Sub GestisciErrore(NumeroErrore, DescrizioneErrore, Pagina)
  'compongo il messaggio
  Messaggio = "Errore nell'appicazione" & vbCrLf
  Messaggio = Messaggio & "Numero errore: " & NumeroErrore & vbCrLf
  Messaggio = Messaggio & "Descrizione Errore: " & vbCrLf
  Messaggio = Messaggio & DescrizioneErrore & vbCrLf
  Messaggio = Messaggio & "Pagina: " & Pagina & vbCrLf
  'invio la mail
```

```
  Set objMail = Server.CreateObject("CDONTS.NewMail")
  objMail.From = "Erors@myweb.it"
  objMail.To = "webmaster@myweb.it"
  objMail.Subject = "Errore nell'applicazione"
  objMail.Body = Messaggio
  objMail.importance= 0
  objMail.Send
  Set objMail = Nothing

  'ripulisco Err
  Err.Clear
  'redirigo l'utente
  Response.Redirect("PaginaMessaggio.asp")
End Sub
%>
```

Vediamo come funziona.

Viene effettuato un test su Err.Number, se questo è diverso da 0, vuol dire che si sono verificati errori a runtime.

In quel caso lancio una proceura che invia la mail all'amministratore con i dettagli dell'errore.

In GestisciErrore, potrei anche inserire funzioni di log che memorizzano le informazioni su database o file di testo.

Transactions

Introduzione

Le transazioni sono un meccanismo per mantenere l'integrità dei dati nei database transazionali come SQL Server. E' possible utilizzarle in ASP in modo abbastanza semplice.

Cosa sono le transazioni

In SQLServer, Quando eseguiamo una serie di query all'interno di una transazione, abbiamo la possibilità di eseguirle tutte (effettuare il commit) oppure nessuna (un rollback della transazione)

Una transazione SQLServer ha la seguente forma:

```
begin transaction
query 1
query2
...
query n
commit (oppure rollback)
```

E' possibile implementare le transactions in asp utilizzando MTS (Microsoft's answer to transactions).

Utilizzo delle transazioni

Le transactions possono risultare utili per evitare l'inserimento dei dati quando una delle query che stiamo eseguendo genera un errore.

Possiamo utilizzare l'oggetto Err per capire se si sono verificati errori ed eseguire il rollback, oppure, se non si sono verificati errori fare il commit e salvare i dati su database.

Implementare le transazioni in ASP

Per utilizzare le transactions in ASP occorre inserire una direttiva nelle nostre pagine che ne fanno uso.

```
<%@ TRANSACTION = proprieta %>
```

La proprietà può assumere uno dei valori riassunti nella tabella in basso.

Proprietà	Descrizione
Requires_New	La pagina crea sempre una nuova transazione
Required	La pagina crea una transazione se non ne trova una attiva, altrimenti usa quella che trova.

Supported	La pagina utilizza le transazioni attive ma non ne può creare di sue.
Not_Supported	Le transazioni non sono supportate

Anche se in genere utilizzeremo le transazioni in un unica pagina asp, è possibile realizzare transazioni che "durano" per più pagine, pensiamo ad esempio ad un wizard in cui vogliamo salvare i dati su database soltanto se abbiamo compilato correttamente tutti i campi di una serie di form uno dopo l'altro.

Con "Transaction = Supported", possiamo far si che la transazione continui quando si passa da una pagina all'altra.

Commit e Rollback delle transazioni

Creata la transazione abbiamo due metodi per eseguire il commit oppure il rollback della transazione.

```
ObjectContext.SetComplete
```

esegue il commit

```
ObjectContext.SetAbort
```

per il rollback

Abbiamo inoltre due eventi che vengono scatenati dalle due funzioni

```
Sub OnTransactionCommit()
'codice
End Sub
```

viene eseguita in caso di commit

```
Sub OnTransactionAbort()
'codice
End Sub
```

viene eseguita in caso di rollback

Esempio

Visti i comandi base, vediamo come utilizzarli in un caso concreto

```
<%@ LANGUAGE="VBSCRIPT" CODEPAGE="1252" TRANSACTION=Required %>
<%
On Error Resume Next

Set connDB = Server.CreateObject("ADODB.CONNECTION")
connDB.ConnectionTimeout = 0
connDB.Open "DSN=DATASOURCE"

' qui eseguo una serie di operazioni, ad esempio inserimenti su database
connDB.Execute("insert into table (nome,cognome) values ('paloino','paperino')")
connDB.Execute("insert into table (nome,cognome) values ('paperon','de paperoni')")

'controllo se ci sono stati errori
If err.number > 0 Then
  ObjectContext.SetAbort
Else
  ObjectContext.SetComplete
End If

Sub OnTransactionCommit()
  response.write("errore nell'inserimento")
End Sub

Sub OnTransactionAbort()
  response.write("dati inseriti")
End Sub
%>
```

Classi ed Oggetti

Introduzione

In questo capitolo vedremo come creare ed utilizzare le classi VBScript.

Classi

Una classe è un modo per incapsulare i dati, una classe può contenere proprietà (le variabili) e metodi (le funzioni).

Mediante le classi possiamo costruire dei tipi complessi riutilizzabili nelle nostre applicazioni.

Supponiamo di voler rappresentare una persona (ad esempio per l'inserimento in un'anagrafica). Supponiamo di voler memorizzare nome, cognome ed email.

Possiamo utilizzare una classe.

```
Class Persona
  Public Nome
  Public Cognome
  Public Email
End Class
```

Così facendo diciamo che una persona è fatta da un nume, un cognome ed una e-mail.

Se però vogliamo rappresentare una persona in particolare, ad esempio Paolino Paperino, cha ha come mail paolino.paperino@topolinia.dys, quello che ci serve è un "oggetto" di tipo persona, quello che si chiama una "istanza" della classe

La mia prima classe VBScript

La dichiarazione di una classe va fatta mediante la parola chiave "Class", Tutto il contenuto della classe starà all'interno di "Class" ed "End Class"

```
<%
Class clPersona
  Public Nome
  Public Cognome
  Public Email
End Class
%>
```

All'interno dala classe avremo le Proprietà ed i Metodi per manipolare la classe.

Per creare un'oggetto di tipo persona occorre "Istanziare" la classe, questa operazione viene fatta mediante la parola chiave New.

```
<%
Dim persona
Set persona = New clPersona
%>
```

La parola chiave Set, esegue l'operazione di assegnamento alla variabile persona.

Per distruggere l'oggetto una volta utilizzato e liberare le risorse, possiamo utilizzare la seguente sintassi:

```
<%
Set persona = Nothing
%>
```

Proprietà Private e Public

Nell'esempio precedente abbiamo visto che una persona può avere un nome, un cognome ed una mail. Senza saperlo abbiamo definito tre "proprietà" della classe clPersona.

Le proprietà sono delle variabili degli oggetti della nostra classe, e come tali possono essere lette oppure scritte.

Una delle principali caratteristiche della programmazione ad oggetti è l'incapsulamento dei dati, ovvero la possibilità di definire delle proprietà della classe non manipolabili dall'esterno se non attraverso dei metodi della classe stessa. L'incapsulamento di effettua tramite i modificatori di accesso: una proprietà può avere un modificatore di accesso che può essere impostato a Public oppure Private.

Con Public, le proprietà possono essere lette e scritte al di fuori della classe, con Private invece le proprietà sono interne alla classe e possono essere accedute esclusivamente da metodi della classe.

Vediamo un esempio

```
<%
Class clPersona
```

```
  Public Nome
  Public Cognome
  Public Email
End Class
%>
```

Così facendo abbiamo definito la classe clPersona, con tre proprietà Public, le quali possono essere accedute dall'esterno tramite la notazione puntata

variabile = istanza.proprietà 'in lettura

istanza.proprietà = variabile 'in scrittura

Costruttore

Il costruttore di una classe è un metodo che viene richiamato quando creiamo una istanza di una classe. Può essere utilizzato per inizializzare la classe, impostandone i valori di default.

In VBScript occorre creare una procadura chiamata Class_Initialize()

```
Definizione
Private Sub Class_Initialize()
  'corpo del metodo
End Sub
```

Esempio

```
<%
Class clPersona

  'Proprietà
  Public Nome
  Public Cognome
  Public Email

  'Costruttore
  Private Sub Class_Initialize()
    Response.Write "Sto creando un istanza dell'oggetto<br>"
    Nome = "Paolino"
    Cognome = "Paperino"
    Email = "paolino.paperino@paperopoli.dys"
  End Sub

End Class

'Istanzio la classe
Set persona = New clPersona
Response.Write "Nome: " & persona.Nome & "<br>"
Response.Write "Cognome: " & persona.Cognome & "<br>"
Response.Write "Email: " & persona.Email & "<br>"
Set persona = Nothing
%>
```

Otterremo come output:

Sto creando un istanza dell'oggetto

Nome: Paolino

Cognome: Paperino

Email: paolino.paperino@paperopoli.dys

Nota:Non è possibile creare dei costruttori con argomenti.

Distruttore

Il distruttore di una classe, è un matodo che viene eseguito quando distruggiamo una classe tramite Set istanzaClasse = Nothing

Possiamo utilizzarlo per liberare le risorse ed eseguire le operazioni di finalizzazione.

Definizione

```
Private Sub Class_ Terminate()
  'corpo del metodo
End Sub
```

Esempio

```
<%
Class clPersona

  'Proprietà
  Public Nome
  Public Cognome
  Public Email

  'Costruttore
  Private Sub Class_Initialize()
    Response.Write "Sto creando un istanza dell'oggetto<br>"
    Nome = "Paolino"
    Cognome = "Paperino"
    Email = "paolino.paperino@paperopoli.dys"
  End Sub

  'distruttore
  Private Sub Class_Terminate()
    Response.Write "Sto distruggendo l'istanza dell'oggetto<br>"
  End Sub

End Class

'Istanzio la classe
Set persona = New clPersona
Response.Write "Nome: " & persona.Nome & "<br>"
Response.Write "Cognome: " & persona.Cognome & "<br>"
Response.Write "Email: " & persona.Email & "<br>"
Set persona = Nothing
%>
```

Otterremo come output:

```
Sto creando un istanza dell'oggetto
Nome: Paolino
Cognome: Paperino
Email: paolino.paperino@paperopoli.dys
Sto distruggendo l'istanza dell'oggetto
```

I metodi di accesso

I metodi di accesso sono delle funzioni particolari che permettono di accedere alle proprietà protette di una classe.

Il VBScript abbiamo la proprietà Let e Get

Proprietà Let

Consente di assegnare dei valori alle proprietà private di una classe

```
[Public | Private] Property Let nome ([argomenti,] valore)
  'istruzioni
End Property
```

Proprietà Get

Consente di leggere i valori alle proprietà private di una classe

```
[Public | Private] Property Get nome [(argomenti)]
  [istruzioni]
  [[Set] nome = espressione]
End Property
```

E' possibile interrompere l'esecuzione del codice all'interno di una Proprietà tramite l'istruzione Exit Property

Esempio

```
<%
Class clPersona

  'Proprietà
```

```
  Private prNome
  Public Property Let Nome(strNome)
    prNome = strNome
  End Property
  Public Property Get Nome()
    Nome = prNome
  End Property

  Private prCognome
  Public Property Let Cognome(strCognome)
    prCognome = strCognome
  End Property
  Public Property Get Cognome()
    Cognome = prCognome
  End Property

  Private prEmail
  Public Property Let Email(strEmail)
    prEmail = strEmail
  End Property
  Public Property Get Email()
    Email = prEmail
  End Property

  'Costruttore
  Private Sub Class_Initialize()
    Response.Write "Sto creando un istanza dell'oggetto<br>"
    Nome = "Paolino"
    Cognome = "Paperino"
    Email = "paolino.paperino@paperopoli.dys"
  End Sub

  'distruttore
  Private Sub Class_Terminate()
    Response.Write "Sto distruggendo l'istanza dell'oggetto<br>"
  End Sub

End Class

'Istanzio la classe
Set persona = New Persona

Response.Write "Prima dell'assegnazione"
Response.Write "Nome: " & persona.Nome & "<br>"
Response.Write "Cognome: " & persona.Cognome & "<br>"
Response.Write "Email: " & persona.Email & "<br>"

persona.Nome = "Paperon"
persona.Cognome = "De Paperoni"
persona.Email = "paperon.depaperoni@paperopoli.dys"

Response.Write "Dopo l'assegnazione"
Response.Write "Nome: " & persona.Nome & "<br>"
Response.Write "Cognome: " & persona.Cognome & "<br>"
Response.Write "Email: " & persona.Email & "<br>"

Set persona = Nothing
%>
```

Metodi

Abbiamo due tipi di metodi, Funzioni (che hanno un valore di ritorno) e le procedure (che processano i dati senza restituire un valore di ritorno).

Anche i metodi possono essere pubblici o privati, i metodi privati sono interni alla classe e non possono essere eseguiti dall'esterno, mentre quelli pubblici possono essere richiamati all'esterno.

Funzioni

```
[Public | Private] Sub nomeProcedura [(argomenti)]
  [istruzioni]
End Sub
```

Procedure

```
[Public | Private] Function nomeFunzione [(argomenti)]
  [istruzioni]
  nomeFunzione = valore
End Property
```

Esempio

```
<%
Class clPersona

  'Proprietà
  Private prNome
  Public Property Let Nome(strNome)
    prNome = strNome
  End Property
  Public Property Get Nome()
    Nome = prNome
  End Property

  Private prCognome
  Public Property Let Cognome(strCognome)
    prCognome = strCognome
  End Property
  Public Property Get Cognome()
    Cognome = prCognome
  End Property

  Private prEmail
  Public Property Let Email(strEmail)
    prEmail = strEmail
  End Property
  Public Property Get Email()
    Email = prEmail
  End Property

  'Costruttore
  Private Sub Class_Initialize()
    Response.Write "Sto creando un istanza dell'oggetto<br>"
    Nome = "Paolino"
    Cognome = "Paperino"
    Email = "paolino.paperino@paperopoli.dys"
  End Sub

  'distruttore
  Private Sub Class_Terminate()
    Response.Write "Sto distruggendo l'istanza dell'oggetto<br>"
  End Sub

  Public Sub StampaPersona()
    Response.Write "Nome: " & Nome & ", Cognome:" & Cognome & "<br>"
  End Sub

  Public Function dettaglioPersona()
    dettaglioPersona = "Nome: " & Nome & ", Cognome:" & Cognome & "<br>"
  End Function

End Class

'Istanzio la classe
Set persona = New Persona

Response.Write "Prima dell'assegnazione"
Response.Write "Nome: " & persona.Nome & "<br>"
```

```
Response.Write "Cognome: " & persona.Cognome & "<br>"
Response.Write "Email: " & persona.Email & "<br>"

persona.Nome = "Paperon"
persona.Cognome = "De Paperoni"
persona.Email = "paperon.depaperoni@paperopoli.dys"

Response.Write "Dopo l'assegnazione"
Response.Write "Nome: " & persona.Nome & "<br>"
Response.Write "Cognome: " & persona.Cognome & "<br>"
Response.Write "Email: " & persona.Email & "<br>"

Response.Write "Richiamo il metodo"
persona.StampaPersona()

Response.Write "Richiamo la funzione"
dettaglioPersona = persona.dettaglioPersona()
Response.Write dettaglioPersona

Set persona = Nothing
%>
```

Borsoft

Oggetti ASP

Oggetto Request

Oggetto Request

Quando un browser richiede una pagina, questo atto è chiamato request. Request viene usato per ottenere informazioni dall'utente. Le sue collezioni, proprietà de metodi verranno descritte di seguito:

Collezioni

Collezione	Descrizione
ClientCertificate	Contiene i valori dei campi memorizzati nel certificato del client
Cookies	Contiene i cookie inviati in una richiesta HTTP
Form	Contiene i valori provenienti da una form inviata tramite il metodo post
QueryString	Contiene i valori delle variabili inviate tramite una richiesta HTTP
ServerVariables	Contiene tutti i valori delle variabili del server

La collection Cookies

La collection Cookies è usata per impostare o leggere i valori dei cookies. Se un cookie non esiste, verrà creato e prenderà il valore specificato.

Nota: Il comando Response.Cookies deve essere inserito prima del tag <html>.

Sintassi

```
Response.Cookies(nome)[(chiave)|.attributo]=valore
nomeVariabile=Request.Cookies(nome)[(chiave)|.attributo]
```

Parametro	Descrizione
nome	Il nome del cookie
valore	Il valore del cookie
attributo	Opzionale. Da informazioni sul cookie. Può avere uno dei seguenti valori: Domain - Il cookie è inviato solo a richieste per questo dominio Expires – La data in cui il cookie scade (se non viene specificata, il cookie scade al termine della sessione) HasKeys – Controlla se il cookie ha delle chiavi Path – Se impostato, il cookie è inviato solo per richieste di questo percorso Secure – Indica se il cookie è sicuro
chiave	Opzionale. Specifica le Specifies la chiave a cui è assegnato il valore

La collection Form

La collection Form è usata per recuperare i valori da una form inviata con il metodo POST.

Sintassi

```
Request.Form(elemento)
Request.Form[(indice)]
```

Parametro	Descrizione
elemento	Il nome dell'elemento del form da cui si vogliono leggere i valori
indice	Opzionale. Sepcifica un parametro tra molti

La collection QueryString

E' usata per recuperare i valori delle variabili inviata in una query string HTTP.

Sintassi

```
Request.QueryString(variabile)
Request.QueryString[(indice)]
```

Parametro	Descrizione
variabile	Il nome della variabile da leggere
indice	Opzionale. Specifica uno dei valori per una variabile

La collection ServerVariables

E' usata per recuperare i valori delle variabili del server.

Sintassi

```
Request.ServerVariables (server_variable)
```

Parametro	Descrizione
server_variable	Il nome della variabile del server

Variabili del Server

Variabile	Descrizione
ALL_HTTP	Ritorna tutti gli header HTTP inviati dal server
ALL_RAW	Ritorna tutti gli header HTTP inviati dal server in formato raw (grezzo, nello stato in cui sono state inviate al browser)
APPL_MD_PATH	Ritorna la meta base per la DLL ISAPI
APPL_PHYSICAL_PATH	Ritorna il percorso fisico corrispondente alla meta base (esempio c:\Inetpub\wwwroot)
AUTH_PASSWORD	Restituisce il valore inserito nella maschera di autenticazione utente
AUTH_TYPE	Il metodo di autenticazione usato dal server
AUTH_USER	L'utente autenticato
CERT_COOKIE	Restituisce un ID univoco per il certificato digitale del client sottoforma di stringa
CERT_FLAGS	Il bit0 è 1 se il certificato del client è presente, e il bit1 vale 1 se la certificazione del client non è valida
CERT_ISSUER	Ritorna il campo distributore di certificate
CERT_KEYSIZE	Il numero di bit della chiave per il Secure Sockets Layer (es 64 o 128)
CERT_SECRETKEYSIZE	Il numero di bit della chiave privata dei certificati del server
CERT_SERIALNUMBER	Il numero di serie del cerificato del client
CERT_SERVER_ISSUER	L'autorità di distribuzione del server
CERT_SERVER_SUBJECT	Il soggetto del certificato del server
CERT_SUBJECT	Il soggetto del certificato del client
CONTENT_LENGTH	La lunghezza del contenuto inviato dal client
CONTENT_TYPE	Il tipo MIME inviato dal client
GATEWAY_INTERFACE	La revisione della specifica CGI usata dal server
HTTP_ACCEPT	L' Accept header
HTTP_ACCEPT_LANGUAGE	La lingua usata per visualizzare il contenuto
HTTP_COOKIE	Il cookie incluso nella richiesta
HTTP_REFERER	L'URL da cui proviene la richiesta
HTTP_USER_AGENT	Il tipo di browser che ha inviato la richiesta
HTTPS	E' ON se la richiesta viene da un canale sicuro, OFF altrimenti
HTTPS_KEYSIZE	Il numero di bit della chiave per il Secure Sockets Layer

HTTPS_SECRETKEYSIZE	Il numero di bit della chiave privata dei certificate del server
HTTPS_SERVER_ISSUER	Il distributore del certificato del server
HTTPS_SERVER_SUBJECT	Il soggetto del certificato del server
INSTANCE_ID	L'ID per l'istanza di IIS ion formato testo
INSTANCE_META_PATH	Il percorso della meta base per l'istanza di IIS che risponde alla richiesta
LOCAL_ADDR	L'indirizzo IP del server che risponde alla richiesta
LOGON_USER	L'account di windows con cui l'utente si è loggato
PATH_INFO	Il percorso virtuale della pagina da cui il client invia la richiesta
PATH_TRANSLATED	Il passaggio da virtuale a fisico di PATH_INFO
QUERY_STRING	La query string inviata dal client
REMOTE_ADDR	L'indirizzo IP del client che invia la richiesta HTTP
REMOTE_HOST	Il nome dell'host che effettua la richiesta HTTP
REMOTE_USER	La username inviata dall'utente
REQUEST_METHOD	Il metodo con cui il client ha effettuato la richiesta http (GET, POST...)
SCRIPT_NAME	Il percorso virtual dello script
SERVER_NAME	Il nome del server
SERVER_PORT	La porta del server a cui è inviata la richiesta
SERVER_PORT_SECURE	1 se la porta del server è protetta, 0 altrimenti
SERVER_PROTOCOL	Il protocollo usato dal server
SERVER_SOFTWARE	Nome e versione del software usato dal server
URL	L'url base della richiesta

Proprietà

Proprietà	Descrizione
TotalBytes	Il numero di bytes inviati dal client nel corpo della richiesta

Proprietà TotalBytes

Restituisce il numero di bytes inviati dal client nel corpo della richiesta.

Sintassi

```
bytesInviati=Request.Totalbytes
```

Metodi

Metodo	Descrizione
BinaryRead	Legge un numero di byte dei dati inviati al server dal client dal corpo della richiesta POST

Metodo BinaryRead

Legge un numero di byte dei dati inviati al server dal client dal corpo della richiesta POST. I dati sono memorizzati in un array.

Sintassi

```
Request.BinaryRead(numero)
```

Parametro	Descrizione
numero	Quanti byte devono essere letti dal client

Oggetto Response

Oggetto Response

L'oggetto ASP Response è usato per inviare dell'output all'utente. Le sue collections, proprietà e metodi sono descritti in basso:

Collections

Collection	Descrizione
Cookies	Imposta il valore di un cookie. Se il cookie non esiste, verrà creato, e prenderà il valore specificato

La Collection Cookies

La collection Cookies è usata per impostare o leggere il valore di un cookie. Se il cookie non esiste, verrà creato, e prenderà il valore specificato.

Sintassi

```
Response.Cookies(nome)[(chiave)] = valore
nomeVariabile=Request.Cookies(nome)[(chiave)]
```

Parametri	Descrizione
nome	Richiesto.
valore	Richiesto per il commando Response.Cookies.
attributo	Opzionale. Specifica le informazioni
Può essere uno dei seguenti parametri	
Domain	Sola scrittura. Cookie inviato solo a richieste da questo dominio
Expires	Sola scrittura. La data in cui il cookie espirerà. Se non viee impostata il cookie è temporaneo e verrà cancellato alla fine della sessione
HasKeys	Sola lettura. Se il cookie ha chiavi
Path	Sola scrittura. Se impostato invia i cookies solo da un determinato percorso dell'applicazione
Secure	Sola scrittura. Indica
chiave	Opzionale.

Proprietà

Proprietà	Descrizione
Buffer	Specifica se bufferizzare l'output della pagina
CacheControl	Specifica se il proxy può mettere nella cache l'output della pagina
Charset	Aggiunge il nome del character-set all'header dell'oggetto Response
ContentType	Imposta il content type HTTP per l'oggetto Response
Expires	Per quanto tempo (in minuti) una pagina sarà tenuta nella cache prima che scada
ExpiresAbsolute	Imposta una data ed un'ora in cui la pagina scadrà
IsClientConnected	Se il client è connesso
Pics	Aggiunge un valore alla label PICS
Status	Specifica il valore dello stato restituito dal server

Proprietà Buffer

La proprietà Buffer specifica se bufferizzare l'output oppure no. Quando l'output è bufferizzato, il server trattiene il response finchè tutti gli script lato server sono eseguiti oppure finchè uno script non chiama un metodo Flush o End.

Nota: Se la proprietà viene impostata, occorre farlo prima del tag <html> nel file.asp

Sintassi

```
response.Buffer[=flag]
```

Parametro	Descrizione
flag	Valore booleano che specifica se bufferizzare la pagina oppure no.

False indica che non c'è buffering. Il server invia l'output, mentre questo viene processato.

True indica che c'è buffering. Il server non invia l'output, finchè tutti gli script non sono stati eseguiti (default per IIS 5.0 e successivi).

La proprietà CacheControl

La proprietà CacheControl imposta se un proxy può mettere o meno nella cache l'output generato da ASP oppure no.

Sintassi

```
response.CacheControl[=control_header]
```

Parametro	Descrizione
control_header	Può essere impostato a "Public" o "Private".

Private è default ed indica che solo le cache private possono memorizzare la pagina, e non i proxy.

Con Public indica i proxy metteranno la pagina in cache.

La proprietà Charset

La proprietà Charset aggiunge il nome del set di caratteri all'header dell'oggetto Response. Il default è ISO-LATIN-1.

Sintassi

```
response.Charset(nome)
```

Parametro	Descrizione
nome	Una stringa che specifica il set di caratteri per la pagina

La proprietà ContentType

Imposta il content type HTTP per l'oggetto response.

Sintassi

```
response.ContentType[=contenttype]
```

Parametri	Descrizione
contenttype	Una stringa ce descrive il content type.

Per una lista completa vedere la documentazione del browser o le specifiche http.

Esempi

Ecco alcuni dei valori più comuni. Se una pagina non ha la proprietà impostata, il default è: text/html

```
<%response.ContentType="text/HTML"%>
<%response.ContentType="text/plain"%>
<%response.ContentType="image/GIF"%>
<%response.ContentType="image/JPEG"%>
<%response.ContentType="application/vnd.ms-excel"%>
```

La proprietà Expires

Imposta per quanto tempo in minuti, la pagina sarà tenuta nella cache del browser.

Sintassi

```
response.Expires[=numero]
```

Parametri	Descrizione
numero	Il tempo in minuti prima dell'expire della pagina

La proprietà ExpiresAbsolute

La proprietà ExpiresAbsolute imposta una data de un ora in cui la pagina espirerà.

Sintassi

```
response.ExpiresAbsolute[=[data][ora]]
```

Parametri	Descrizione
data	La data in cui la pagina espira.
ora	L'ora in cui la pagina espira.

Proprietà IsClientConnected

Indica se il client si è disconnesso dal server.

Sintassi

```
response.IsClientConnected
```

La proprietà Status

Specifica il valore dello stato restituito dal server.

Sintassi

```
response.Status=descrizione
```

Parametri	Descrizione
descrizione	Un numero di tre cifre ed una descrizione. (Ad esempio 404 Not Found oppure 200 OK)

I valori sono definiti nelle specifiche HTTP.

Metodi

Metodo	Descrizione
AddHeader	Aggiunge un nuovo header HTTP ed un valore alla risposta HTTP
AppendToLog	Aggiunge una sringa alla fine del log nel server
BinaryWrite	Scrive dati direttamente all'output senza convertire I caratteri
Clear	Ripulisce ogni output bufferizzato
End	Ferma la procesazione dello script e restituisce il risultato corrente
Flush	Invia immediatamente al browser l'output HTML bufferizzato
Redirect	Redireziona l'utente ad un URL diverso
Write	Scrive una stringa sull'output

Metodo AddHeader

Aggiunge un nuovo header HTTP ed un valore alla risposta HTTP.

Sintassi

```
response.AddHeader nome,valore
```

Parametri	Descrizione
nome	Il nome dell'header (non può contenere underscores)
valore	Il valore iniziale della variabile

Metodo AppendToLog

Aggiunge una sringa alla fine del log nel server. .

Sintassi

```
response.AppendToLog stringa
```

Parametri	Descrizione

stringa	La stringa da aggiungere al log (non può contenere virgole)

Metodo BinaryWrite

Fa una scrittura binaria senza convertire i caratteri.

Sintassi

```
response.BinaryWrite dati
```

Parametri	Descrizione
dati	L'informazione binaria da inviare

Il metodo Clear

Ripulisce ogni output bufferizzato.

Sintassi

```
response.Clear
```

Metodo End

Ferma la procesazione dello script e restituisce il risultato corrente.

Sintassi

```
Response.End
```

Metodo Flush

Invia immediatamente al browser l'output HTML bufferizzato.

Sintassi

```
Response.Flush
```

Metodo Redirect

Redireziona l'utente ad un URL diverso.

Sintassi

```
Response.Redirect URL
```

Parametri	Descrizione
URL	L'indirizzo a cui redireziono il browser

Metodo Write

Scrive una stringa sull'output.

Sintassi

```
Response.Write dato
```

Parametri	Descrizione
dato	Il dato da scrivere (variabili, stringhe...)

Oggetto Application

Un gruppo di file che lavorano insieme per uno stesso scopo viene chiamato applicazione. L'oggetto Application in ASP è usato per legare tutti questi file insieme.

Oggetto Application

Un'applicazione su web può essere un gruppo di file ASP. I file ASP lavorano insieme per lo stesso scopo. L'oggetto application viene usato per legare tutti questi file. L'oggetto Application è usato per memorizzare e leggere variabili da qualsiasi pagina, proprio come l'oggetto session. La differenza è che l'oggetto Application viene condiviso da tutti gli utenti, mentre l'oggetto session vale per il singolo utente.

Collezioni

Collezione	Descrizione
Contents	Contiene tutti gli elementi aggiunti all'applicazione tramite script
StaticObjects	Contiene tutti gli elementi aggiunti all'applicazione tramite il tag HTML<object>

La collezione Contents

Contiene tutti gli elementi aggiunti all'applicazione tramite script.

Suggerimento: Per eliminare degli elementi dalla collezione Contents, si possono usare i metodi Remove e RemoveAll.

Sintassi

```
Application.Contents(chiave)
```

Parametro	Descrizione
chiave	Il nome dell'elemento da recuperare

Metodi

Metodo	Descrizione
Contents.Remove	Elimina un elemento dalla collection
Contents.RemoveAll()	Elimina tutti gli elementi della collection
Lock	Impedisce ad altri utenti di modificare le variabili dell'oggetto Application
Unlock	Abilita gli utenti amodificare le variabili dell'oggetto Application (dopo che l'applicazione è stata bloccata con il metodo lock)

Metodo Contents.Remove

Elimina un elemento dalla collection.

Sintassi

```
Application.Contents.Remove(nome|indice)
```

Parametro	Descrizione
nome	Il nome dell'elemento da rimouvere
indice	L'indice dell'elemento da rimouvere

Metodo Contents.RemoveAll

Elimina tutti gli elementi della collection.

Sintassi

```
Application.Contents.RemoveAll()
```

Eventi

Evento	Descrizione
Application_OnEnd	Quando tutte le sessioni utente sono terminate e l'applicazione termina.
Application_OnStart	La prima volta che una nuova sessione viene creata

Eventi Application_OnStart

La prima volta che una nuova sessione viene creata. E' inserito nel Global.asa.

Eventi Application_OnEnd

Quando tutte le sessioni utente sono terminate e l'applicazione termina (quando si ferma il webserver). E' inserito nel Global.asa.

Oggetto Session

Oggetto Session

L'oggetto Session è utilizzato per memorizzare informazioni sulla sessione utente. Le variabili memorizzate nell'oggetto Session tengono informazioni su un singolo utente, e sono disponibili a tutte le pagine dell'applicazione.

Collezioni

Collezione	Descrizione
Contents	Contiene tutti gli elementi aggiunti alla sessione tramite script
StaticObjects	Contiene tutti gli elementi aggiunti alla sessione tramite il tag HTML<object>

La collezione Contents

Contiene tutti gli elementi aggiunti alla sessione tramite script.

Sintassi

```
Session.Contents(chiave)
```

Parametro	Descrizione
chiave	Il nome dell'elemento da recuperare

La collezione StaticObjects

Contiene tutti gli elementi aggiunti alla sessione tramite il tag HTML<object>.

Sintassi

```
Session.StaticObjects(chiave)
```

Parametro	Descrizione
chiave	Il nome dell'elemento da recuperare

Proprietà

Proprietà	Descrizione
CodePage	Specifica il set di caratteri che sarà utilizzato nella visualizzazione di contenuti dinamici
LCID	Imposta o restituisce un intero che specifica un luogo o una regione. Contenuti come la data, l'ora e la moneta verranno visualizzati in relazione alla regione scelta.
SessionID	Restituisce un identificativo univoco per ciascun utente. L'ID viene generato dal server
Timeout	Restituisce oppure imposta il timeout per l'oggetto sessione nell'applicazione.
Proprietà CodePage	Specifica il set di caratteri che sarà utilizzato nella visualizzazione di contenuti dinamici.

Proprietà CodePage

Specifica il set di caratteri che sarà utilizzato nella visualizzazione di contenuti dinamici.

Sintassi

```
Session.CodePage(=Codepage)
```

Parametro	Descrizione
codepage	Definisce il code page (il set di caratteri) per il sistema

Proprietà LCID

Imposta o restituisce un intero che specifica un luogo o una regione. Contenuti come la data, l'ora e la valuta verranno visualizzati in relazione alla regione scelta.

Sintassi

```
Session.LCID(=LCID)
```

Parametro	Descrizione
LCID	Identificatore locale

Proprietà SessionID

Restituisce un identificativo univoco per ciascun utente. L'ID viene generato dal server

Sintassi

```
Session.SessionID
```

Proprietà Timeout

Restituisce oppure imposta il timeout per l'oggetto sessione nell'applicazione. Se l'utente non richiede pagine entro il timeout, sa sessione termina.

Sintassi

```
Session.Timeout[=nMinuti]
```

Parametro	Descrizione
nMinuti	La durata in minuti della sessione Il default è 20 minuti

Metodi

Metodo	Descrizione
Abandon	Termina una sessione e ne distrugge le variabili
Contents.Remove	Elimina un elemento dalla collection
Contents.RemoveAll()	Elimina tutti gli elementi della collection

Metodo Abandon

Termina una sessione e ne distrugge le variabili.

Sintassi

```
Session.Abandon
```

Metodo Contents.Remove

Elimina un elemento dalla collection.

Sintassi

```
Session.Contents.Remove(nome| indice)
```

Parametro	Descrizione
nome	Il nome dell'elemento da rimouvere
indice	L'indice dell'elemento da rimouvere

Metodo Contents.RemoveAll

Elimina tutti gli elementi della collection.

Sintassi

```
Session.Contents.RemoveAll()
```

Eventi

Evento	Descrizione
Session_OnEnd	Quando termina una sessione
Session_OnStart	Quando termina una sessione

Oggetto Server

Oggetto Server

L'oggetto Server è utilizzato per per accedere alle proprietà ed ai metodi del server. Le proprietà ed i metodi sono descritti di seguito:

Proprietà

Proprietà	Descrizione
ScriptTimeout	Imposta o restituisce il numero massimo in secondi per completare uno script

Proprietà ScriptTimeout

Imposta o restituisce il massimo numero in secondi utili alla terminazione di uno script.

Sintassi

```
Server.ScriptTimeout[=NumSecondi]
```

Parametro	Descrizione
NumSecondi	Il numero massimo in secondi per completare uno script, prima che il server lo termini. Il valore di default è 90 secondi

Metodi

Metodo	Descrizione
CreateObject	Crea un'istanza di un oggetto
Execute	Esegue un file ASP, all'interno di un altro
GetLastError()	Restituisce un oggetto ASPError che descrive la condizione di errore accorsa
HTMLEncode	Applica la codifica HTML ad una stringa specifica
MapPath	Mappa un percorso specifico in un percorso fisico
Transfer	Trasferisce tutta l'informazione create in una pagina ASP ad una seconda pagina ASP
URLEncode	Applica le regole di codifica URL ad una stringa

Metodo CreateObject

Crea un'istanza di un oggetto.

Sintassi

```
Server.CreateObject(progID)
```

Part	Descrizione
progID	Richiesto: il tipo di oggetto da creare

Metodo Execute

Esegue un file ASP, all'interno di un altro. Dopo l'esecuzione, il controllo ritorna al file chiamante.

Sintassi

```
Server.Execute(percorso)
```

Parametro	Descrizione
percorso	La posizione del file da eseguire

Metodo GetLastError()

Restituisce un oggetto ASPError che descrive la condizione di errore accorsa.

Sintassi

```
Server.GetLastError()
```

Metodo HTMLEncode

Applica la codifica HTML ad una stringa specifica.

Sintassi

```
Server.HTMLEncode(stringa)
```

Parametro	Descrizione
stringa	La stringa da codificare

Metodo MapPath

Mappa un percorso specifico in un percorso fisico.

Sintassi

```
Server.MapPath(percorso)
```

Parametro	Descrizione
percorso	Un percorso virtuale relative che si vuole mappare in un percorso fisico.

Metodo Transfer

Trasferisce tutta l'informazione create in una pagina ASP ad una seconda pagina ASP .

Sintassi

```
Server.Transfer(percorso)
```

Parametro	Descrizione
percorso	La posizione del file a cui trasferire il controllo

Metodo URLEncode

Applica le regole di codifica URL ad una stringa.

Sintassi

```
Server.URLEncode(stringa)
```

Parametro	Descrizione
stringa	La stringa da codificare

Oggetto AspError

E' usato per visualizzare informazioni dettagliate su qualsiasi errore in uno script.

Oggetto ASPError

E' usato per visualizzare informazioni dettagliate su qualsiasi errore in uno script. L'oggetto viene creato quando si richiama il metodo Server.GetLastError.

Proprietà

Proprietà	Descrizione
ASPCode	Il codice d'errore generato da IIS
ASPDescription	Una descrizione dettagliata dell'errore
Category	La provenienza (linguaggio di scripting, oggetto...)
Column	La colonna che ha generato l'errore
Description	La descrizione dell'errore

File	Il file che ha generato l'errore
Line	Il numero di riga dell'istruzione che ha generato l'errore
Number	Il codice COM standard per l'errore
Source	Il codice della rig ache ha generato l'errore

Oggetto TextStream

Proprietà

Proprietà	Descrizione
AtEndOfLine	Da true se il puntatore è posizionato immediatamente prima della fine del marcatore end-of-line in uno stream di testo, e false altrimenti
AtEndOfStream	Da true se il puntatore è posizionato alla fine dello stream di testo, e false altrimenti
Column	Restituisce il numero di colonna del carattere corrente
Line	Restituisce il numero di riga

Proprietà AtEndOfLine

Da true se il puntatore è posizionato immediatamente prima della fine del marcatore end-of-line in uno stream di testo, e false altrimenti.

Sintassi

```
TextStreamObject.AtEndOfLine
```

Proprietà AtEndOfStream

Da true se il puntatore è posizionato alla fine dello stream di testo, e false altrimenti.

Sintassi

```
TextStreamObject.AtEndOfStream
```

Proprietà Column

Restituisce il numero di colonna del carattere corrente.

Sintassi

```
TextStreamObject.Column
```

Proprietà Line

Restituisce il numero di riga.

Sintassi

```
TextStreamObject.Line
```

Metodi

Metodo	Descrizione
Close	Chiude uno stream di testo aperto
Read	Legge un numero di caratteri da uno stream di testo
ReadAll	Legge un intero stream di testo
ReadLine	Legge una riga da uno stream di testo
Skip	Salta un certo numero di caratteri dalla lettura di uno stream di testo
SkipLine	Salta una riga dalla lettura di uno stream di testo
Write	Scrive del testo in un fle
WriteLine	Scrive del testo ed alla fine inserisce un carattere di new-line
WriteBlankLines	Scrive un certo numero di caratteri di of new-line in un file di testo

Metodo Close

Chiude uno stream di testo aperto.

Sintassi

```
TextStreamObject.Close
```

Metodo Read

Legge un numero di caratteri da uno stream di testo e restituisce una stringa come risultato.

Sintassi

```
TextStreamObject.Read(numchar)
```

Parametro	Descrizione
numchar	Il numero di caratteri da leggere

Metodo ReadAll

Legge un numero di caratteri da uno stream di testo.
Nota: Non opportune per file di grosse dimensioni perchè spreca parecchia memoria.

Sintassi

```
TextStreamObject.ReadAll
```

Metodo ReadLine

Legge una riga da uno stream di testo e restituisce una stringa come risultato.

Sintassi

```
TextStreamObject.ReadLine
```

Metodo Skip

Salta un certo numero di caratteri dalla lettura di uno stream di testo.

Sintassi

```
TextStreamObject.Skip(numCaratteri)
```

Parametro	Descrizione
numCaratteri	Il numero di caratteri da saltare

Metodo SkipLine

Salta una riga dalla lettura di uno stream di testo.

Sintassi

```
TextStreamObject.SkipLine
```

Metodo Write

Scrive del testo in un fle.
Nota: Scrive senza spazi o interruzioni di riga.

Sintassi

```
TextStreamObject.Write(text)
```

Parametro	Descrizione
text	Il testo da scrivere nel file

Metodo WriteLine

Scrive del testo ed alla fine inserisce un carattere di new-line.

Sintassi

```
TextStreamObject.WriteLine(text)
```

Parametro	Descrizione
text	Opzionale. Il testo da scrivere, se non si inserisce nulla viene scritta una riga vuota

Metodo WriteBlankLines

Scrive un certo numero di caratteri di of new-line in un file di testo.

Sintassi

```
TextStreamObject.WriteBlankLines(numero)
```

Parametro	Descrizione
numero	Richiesto. Il numero di caratteri di new-line da inserire nel file

Oggetto Drive

Oggetto Drive

L'oggetto Drive è usato per restituire informazioni su dischi locali o di rete.

Proprietà

Proprietà	Descrizione
AvailableSpace	Restituisce lo spazio disponibile in un drive locale o di rete
DriveLetter	Restituisce la lettera che identifica un disco
DriveType	Restituisce il tipo di disco
FileSystem	Restituisce il tipo di filesystem di un disco
FreeSpace	Restituisce lo spazio disponibile per un utente
IsReady	Se un drive è pronto oppure no
Path	Restituisce il nome del path per un drive specifico
RootFolder	Restituisce la cartella di root di un drive
SerialNumber	Restituisce il numero seriale di un drive
ShareName	Restituisce il nome della condivisione di rete del drive
TotalSize	Restituisce la dimensione di un dive
VolumeName	Restituisce il nome del volume del drive

Proprietà AvailableSpace

Restituisce lo spazio disponibile in un drive locale o di rete.

Sintassi

```
OggettoDrive.AvailableSpace
```

Proprietà DriveLetter

Restituisce la lettera che identifica un disco.

Sintassi

```
OggettoDrive.DriveLetter
```

Proprietà DriveType

Restituisce il tipo di disco.

Può restituire uno dei seguenti valori:

0 = unknown

1 = removable

2 = fixed

3 = network

4 = CD-ROM

5 = RAM disk

Sintassi

```
OggettoDrive.DriveType
```

Proprietà FileSystem

Restituisce il tipo di filesystem di un disco.
Può restituire uno dei seguenti valori:
FAT – per dischi rimovibili
CDFS – per drive CD-ROM
FAT, FAT32 or NTFS – per hard disks con Windows 2000 or Windows NT
FAT or FAT32 - per hard disks con Windows 9x

Sintassi

```
OggettoDrive.FileSystem
```

Proprietà FreeSpace

Restituisce lo spazio disponibile per un utente.

Sintassi

```
OggettoDrive.FreeSpace
```

Proprietà IsReady

Se un drive è pronto oppure no.

Sintassi

```
OggettoDrive.IsReady
```

Proprietà Path

Per recuperare il percorso di una cartelle o un file.

Sintassi

```
OggettoDrive.Path
OggettoFile.Path
OggettoFolder.Path
```

Proprietà RootFolder

Restituisce la cartella di root di un drive.

Sintassi

```
OggettoDrive.RootFolder
```

Proprietà SerialNumber

Restituisce il numero seriale di un drive.

Sintassi

```
OggettoDrive.SerialNumber
```

Proprietà ShareName

Restituisce il nome della condivisione di rete del drive.

Sintassi

```
OggettoDrive.ShareName
```

Proprietà TotalSize

Restituisce la dimensione di un dive.

Sintassi

```
OggettoDrive.TotalSize
```

Proprietà VolumeName

Restituisce il nome del volume del drive.

Sintassi

```
OggettoDrive.VolumeName[=nuovoNome]
```

Parametro	Descrizione
nuovoNome	Opzionale. Imposta un nuovo nome per il drive

Oggetto OggettoFileSystem

Oggetto OggettoFileSystem

E' usato per accedere al file system del server. Di seguito vediamo metodi e proprietà dell'oggetto OggettoFileSystem:

Proprietà

Proprietà	Descrizione
Drives	Restituisce un collection contenente tutti gli oggetti drive del computer

Proprietà Drives

Restituisce un collection contenente tutti gli oggetti drive del computer.

Sintassi

```
myDrives = OggettoFileSystem.Drives
```

Metodi

Metodo	Descrizione
BuildPath	Aggiunge un nome ad un percorso esistente
CopyFile	Copia uno o più file da una directory ad un'altra
CopyFolder	Copia una o più cartelle da una directory ad un'altra
CreateFolder	Crea una nuova cartella
CreateTextFile	Crea un file di testo e restituisce un oggetto TextStream che può essere usato per leggere dal file o scriverci
DeleteFile	Cancella uno o più file
DeleteFolder	Cancella una o più directory
DriveExists	Controlla l'esistenza di un drive
FileExists	Controlla l'esistenza di un file
FolderExists	Controlla l'esistenza di una cartella
GetAbsolutePathName	Restituisce il percorso complete dalla root del drive al file specificato
GetBaseName	Restituisce il nome base di un file o una cartella
GetDrive	Restituisce un oggetto Drive corrispondente al drive in un percorso specifico
GetDriveName	Restituisce nome di uno specifico percorso
GetExtensionName	Restituisce l'estensione di un file
GetFile	Restituisce un oggetto File per un percorso specifico
GetFileName	Restituisce nome del file o di una cartella per l'ultimo componente del percorso specificato
GetFolder	Restituisce una cartella per il percorso specificato
GetParentFolderName	Restituisce il nome della cartella contenitore dell'ultimo componente in un percorso specifico
GetSpecialFolder	Restituisce il percorso di alcune cartelle speciali di windows

GetTempName	Restituisce una cartella temporanea generata in modo random
MoveFile	Sposta uno o più file da una cartella ad un'altra
MoveFolder	Sposta una o più cartelle da una cartella ad un'altra
OpenTextFile	Apre un file e restituisce un oggetto TextStream che può essere utilizzato per accedere al file

Metodo BuildPath

Aggiunge un nome ad un percorso esistente.

Sintassi

```
OggettoFileSystem.BuildPath(percorso,nome)
```

Parametro	Descrizione
percorso	Il percorso a cui aggiungere il nome
nome	Il nome da aggiungere al percorso

Metodo CopyFile

Copia uno o più file da una directory ad un'altra.

Sintassi

```
OggettoFileSystem.CopyFile sorgente,destinazione[,sovrascrivi]
```

Parametro	Descrizione
sorgente	I file da copiare
destinazione	Dove copiare i file
sovrascrivi	Opzionale. Valore booleano che indica se si devono o meno sovrascrivere i file.

Metodo CopyFolder

Copia una o più cartelle da una directory ad un'altra.

Sintassi

```
OggettoFileSystem.CopyFolder sorgente,destinazione[,sovrascrivi]
```

Parametro	Descrizione
sorgente	Le cartelle da copiare
destinazione	Dove copiare le cartelle
sovrascrivi	Opzionale. Valore booleano che indica se si devono o meno sovrascrivere le cartelle.

Metodo CreateFolder

Crea una nuova cartella.

Sintassi

```
OggettoFileSystem.CreateFolder(nome)
```

Parametro	Descrizione
nome	Il nome della cartella da creare

Metodo CreateTextFile

Crea un file di testo e restituisce un oggetto TextStream che può essere usato per leggere dal file o scriverci.

Sintassi

```
OggettoFileSystem.CreateTextFile(nomeFile[,sovrascrivi])
FolderObject.CreateTextFile(nomeFile[,sovrascrivi])
```

Parametro	Descrizione
nomeFile	Il nome del file da creare
sovrascrivi	Opzionale. Valore booleano che indica se si devono o meno sovrascrivere il file

(il default è true).

Metodo DeleteFile

Cancella uno o più file.

Sintassi

```
OggettoFileSystem.DeleteFile(nomeFile[,forza])
```

Parametro	Descrizione
nomeFile	Il nome del file da cancellare
forza	Opzionale. Valore booleano che indica se anche i file di sola lettura devono essere cancellati. Il valore di default è False

Metodo DeleteFolder

Cancella una o più cartelle.

Sintassi

```
OggettoFileSystem.DeleteFolder(nomeCartella[,forza])
```

Parametro	Descrizione
nomeCartella	Cancella una o più directory
forza	Opzionale. Valore booleano che indica se anche le cartelle di sola lettura devono essere cancellate. Il valore di default è False

Metodo DriveExists

Controlla l'esistenza di un drive.

Sintassi

```
OggettoFileSystem.DriveExists(drive)
```

Parametro	Descrizione
drive	Lettera di drive

Metodo FileExists

Controlla l'esistenza di un file.

Sintassi

```
OggettoFileSystem.FileExists(nomeFile)
```

Parametro	Descrizione
nomeFile	Il file di cui si vuole controllare l'esistenza

Metodo FolderExists

Controlla l'esistenza di una cartella.

Sintassi

```
OggettoFileSystem.FolderExists(nomeCartella)
```

Parametro	Descrizione
nomeCartella	La cartella di cui si vuole controllare l'esistenza

Metodo GetAbsolutePathName

Restituisce il percorso complete dalla root del drive al file specificato.

Sintassi

```
OggettoFileSystem.GetAbsolutePathName(percorso)
```

Parametro	Descrizione

percorso	Il percorso da convertire in un path completo

Metodo GetBaseName

Restituisce il nome base di un file o una cartella.

Sintassi

```
OggettoFileSystem.GetBaseName(percorso)
```

Parametro	Descrizione
percorso	Il percorso del file di cui si vuole estrarre il nome base

Metodo GetDrive

Restituisce un oggetto Drive corrispondente al drive in un percorso specifico.

Sintassi

```
OggettoFileSystem.GetDrive(tipoDrive)
```

Metodo GetDriveName

Restituisce nome di uno specifico percorso.

Sintassi

```
OggettoFileSystem.GetDriveName(percorso)
```

Parametro	Descrizione
percorso	Il percorso di cui si vuole avere il nome del drive

Metodo GetExtensionName

Restituisce l'estensione di un file.

Sintassi

```
OggettoFileSystem.GetExtensionName(percorso)
```

Parametro	Descrizione
percorso	Il percorso del file di cui si vuole avere l'estensione

Metodo GetFile

Restituisce un oggetto File per un percorso specifico.

Sintassi

```
OggettoFileSystem.GetFile(percorso)
```

Parametro	Descrizione
percorso	Il percorso del file

Metodo GetFileName

Restituisce nome del file o di una cartella per l'ultimo componente del percorso specificato.

Sintassi

```
OggettoFileSystem.GetFileName(percorso)
```

Parametro	Descrizione
percorso	Percorso di un file o di una cartella

Metodo GetFolder

Restituisce un oggetto cartella per il percorso specificato.

Sintassi

```
OggettoFileSystem.GetFolder(percorso)
```

Parametro	Descrizione
percorso	Il percorso della cartella

Metodo GetParentFolderName

Restituisce il nome della cartella contenitore dell'ultimo componente in un percorso specifico.

Sintassi

```
OggettoFileSystem.GetParentFolderName(percorso)
```

Parametro	Descrizione
percorso	Il percorso del fie o della cartella di cui vogliamo estrarre il contenitore

Metodo GetSpecialFolder

Restituisce il percorso di alcune cartelle speciali di windows.

Sintassi

```
OggettoFileSystem.GetSpecialFolder(nomeCartella)
```

Parametro	Descrizione
nomeCartella	La cartella che vogliamo. 0= La cartella di Windows 1= La cartella di sistema 2= Per i file temporanei

Metodo GetTempName

Restituisce un file o una cartella temporanea generati in modo random.

Sintassi

```
OggettoFileSystem.GetTempName
```

Metodo MoveFile

Sposta uno o più file da una cartella ad un'altra.

Sintassi

```
OggettoFileSystem.MoveFile sorgente,destinazione
```

Parametro	Descrizione
sorgente	I file da spostare
destinazione	Dove spostare i file

Metodo MoveFolder

Sposta una o più cartelle da una cartella ad un'altra.

Sintassi

```
OggettoFileSystem.MoveFolder sorgente,destinazione
```

Parametro	Descrizione
sorgente	Le cartelle da spostare
destinazione	Dove spostare le cartelle

Metodo OpenTextFile

Apre un file e restituisce un oggetto TextStream che può essere utilizzato per accedere al file.

Sintassi

```
OggettoFileSystem.OpenTextFile(nomeFile,modalità,crea,formato
```

Parametro	Descrizione
nomeFile	Il nopme del file da aprire

Modalità	Come aprire il file 1=Sola lettura. 2=In scrittura. 8=Aggiunge del testo in fondo al file.
crea	True indica che il file verrà creato se non esiste. Il default è False
formato	Il formato del file 0=ASCII (default). 1= Unicode. 2=Usa il formato di default del sistema operativo

Oggetto File

Oggetto File

E' usato per ottenere informazioni su uno specifico file.

Proprietà

Proprietà	Descrizione
Attributes	Imposta o restituisce gli attributi di un file
DateCreated	Restituisce data ed ora in cui un file è stato creato
DateLastAccessed	Restituisce data ed ora in cui un file è stato acceduto per l'ultima volta
DateLastModified	Restituisce data ed ora in cui un file è stato modificato per l'ultima volta
Drive	Restituisce la lettera del drive in cui è memorizzato un file
Name	Imposta o restituisce il nome di un file
ParentFolder	Restituisce la cartella in cui è memorizzato un file
Path	Restituisce il percorso di un file
ShortName	Restituisce l'equivalente ad 8 caratteri del nome di un file
ShortPath	Restituisce l'equivalente ad 8 caratteri di un percorso
Size	Restituisce la dimensione di un file in byte
Type	Restituisce il tipo di un file

Proprietà Attributes

Imposta o restituisce gli attributi di un file o di una cartella.

Sintassi

```
OggettoFile.Attributes[=attributi]
```

Parametro	Descrizione
attributi	Specifica gli attributi del file. Può prendere uno o una combinazione dei seguenti valori: 0 = Normale 1 = Sola lettura 2 = Nascosto 4 = File di sistema 16 = Cartella

Proprietà DateCreated

Restituisce data ed ora in cui un file è stato creato.

Sintassi

```
OggettoFile.DateCreated
```

Proprietà DateLastAccessed

Restituisce data ed ora in cui un file è stato acceduto per l'ultima volta.

Sintassi

```
OggettoFile.DateLastAccessed
```

Proprietà DateLastModified

Restituisce data ed ora in cui un file è stato modificato per l'ultima volta.

Sintassi

```
OggettoFile.DateLastModified
```

Proprietà Drive

Restituisce la lettera del drive in cui è memorizzato un file.

Sintassi

```
OggettoFile.Drive
```

Proprietà Name

Imposta o restituisce il nome di un file.

Sintassi

```
OggettoFile.Name[=nuovonome]
```

Proprietà ParentFolder

Restituisce la cartella in cui è memorizzato un file.

Sintassi

```
OggettoFile.ParentFolder
```

Proprietà Path

Restituisce il percorso di un file.

Sintassi

```
OggettoFile.Path
```

Proprietà ShortName

Restituisce l'equivalente ad 8 caratteri del nome di un file

Sintassi

```
OggettoFile.ShortName
```

Proprietà Size

Restituisce la dimensione di un file in byte.

Sintassi

```
OggettoFile.Size
```

Proprietà Type

Restituisce il tipo di un file.

Sintassi

```
OggettoFile.Type
```

Metodi

Metodo	Descrizione
Copy	Copia un file da una posizione ad un'altra
Delete	Elimina un file

Move	Sposta un file da una posizione ad un'altra
OpenAsTextStream	Apre un file e restituisce un oggetto di tipo TextStream per accedere al file

Metodo Copy

Copia un file da una posizione ad un'altra.

Sintassi

```
OggettoFile.Copy(percorso[,sovrascrivi])
```

Parametro	Descrizione
percorso	Dove copiare il file
sovrascrivi	Opzionale. Valore booleano. True indica che un file esistente, in caso di stesso nome, può essere sovrascritto. Il valore di default è true.

Metodo Delete

Elimina un file.

Sintassi

```
OggettoFile.Delete[(forza)]
```

se "forza" è impostato a true, il file è cancellato anche se di sola lettura

Metodo Move

Sposta un file da una posizione ad un'altra.

Sintassi

```
OggettoFile.Move(destinazione)
```

Parametro	Descrizione
destinazione	Dove spostare il file

Metodo OpenAsTextStream

Apre un file e restituisce un oggetto di tipo TextStream per accedere al file.

Sintassi

```
OggettoFile.OpenAsTextStream(modo,formato)
```

Parametro	Descrizione
modo	Come aprire il file. 1 = In lettura 2 = In scrittura 8 = Per scrivere alla fine del file
formato	Il formato del file. 0 = Default. Formato ASCII 1 = Formato Unicode 2 = Usa il default del sistema operativo

Oggetto Folder

Collezioni

Collezione	Descrizione
Files	Restituisce una collezione dei file presenti in una cartella
SubFolders	Restituisce una collezione delle sottocartelle presenti in una cartella

Collezione Files

Restituisce una collezione dei file presenti in una cartella.

Sintassi

```
OggettoFolder.Files
```

Collezione SubFolders

Restituisce una collezione delle sottocartelle presenti in una cartella.

Sintassi

```
OggettoFolder.SubFolders
```

Proprietà

Proprietà	Descrizione
Attributes	Imposta o restituisce gli attributi di una cartella
DateCreated	Restituisce data ed ora in cui una cartella è stata creata
DateLastAccessed	Restituisce data ed ora in cui una cartella è stata acceduta per l'ultima volta
DateLastModified	Restituisce data ed ora in cui una cartella è stata modificata per l'ultima volta
Drive	Restituisce la lettera del drive in cui è memorizzata una cartella
IsRootFolder	Dice se una cartella è root oppure no
Name	Imposta o restituisce il nome di una cartella
ParentFolder	Restituisce la cartella in cui è memorizzata una cartella
Path	Restituisce il percorso di una cartella
ShortName	Restituisce l'equivalente ad 8 caratteri del nome di una cartella
ShortPath	Restituisce l'equivalente ad 8 caratteri di un percorso
Size	Restituisce la dimensione di una cartella in byte
Type	Restituisce il tipo di una cartella

Proprietà Attributes

Imposta o restituisce gli attributi di un file o di una cartella.

Sintassi

```
OggettoFolder.Attributes[=attributi]
```

Parametro	Descrizione
attributi	Specifica gli attributi dela cartella. Può prendere uno o una combinazione dei seguenti valori: 0 = Normale 1 = Sola lettura 2 = Nascosto 4 = File di sistema 16 = Cartella

Proprietà DateCreated

Restituisce data ed ora in cui una cartella è stata creata.

Sintassi

```
OggettoFolder.DateCreated
```

Proprietà DateLastAccessed

Restituisce data ed ora in cui una cartella è stata acceduta per l'ultima volta.

Sintassi

```
OggettoFolder.DateLastAccessed
```

Proprietà DateLastModified

Restituisce data ed ora in cui una cartella è stata modificata per l'ultima volta.

Sintassi

```
OggettoFolder.DateLastModified
```

Proprietà Drive

Restituisce la lettera del drive in cui è memorizzata una cartella.

Sintassi

```
OggettoFolder.Drive
```

Proprietà IsRootFolder

Dice se una cartella è root oppure no.

Sintassi

```
OggettoFolder.IsRootFolder
```

Proprietà Name

Imposta o restituisce il nome di una cartella.

Sintassi

```
OggettoFolder.Name[=nuovonome]
```

Parametro	Descrizione
nuovonome	Opzionale. Specifica il nome del file

Proprietà ParentFolder

Restituisce la cartella in cui è memorizzato una cartella.

Sintassi

```
OggettoFolder.ParentFolder
```

Proprietà Path

Restituisce il percorso di un file.

Sintassi

```
OggettoFolder.Path
```

Proprietà ShortName

Restituisce l'equivalente ad 8 caratteri del nome di una cartella

Sintassi

```
OggettoFolder.ShortName
```

Proprietà ShortPath

Restituisce l'equivalente ad 8 caratteri di un percorso

Sintassi

```
OggettoFolder.ShortPath
```

Proprietà Size

Restituisce la dimensione di una cartella in byte.

Sintassi

```
OggettoFolder.Size
```

Proprietà Type

Restituisce il tipo di una cartella.

Sintassi

```
OggettoFolder.Type
```

Metodi

Metodo	Descrizione
Copy	Copia una cartella da una posizione ad un'altra
Delete	Elimina una cartella
Move	Sposta una cartella da una posizione ad un'altra
OpenAsTextStream	Apre un file in una cartella specifica e restituisce un oggetto di tipo TextStream per accedere al file

Metodo Copy

Copia una cartella da una posizione ad un'altra.

Sintassi

```
OggettoFolder.Copy(destinazione[,sovrascrivi])
```

Parametro	Descrizione
destinazione	Dove copiare la cartella
sovrascrivi	True indica che una cartella esistente, in caso di stesso nome, può essere sovrascritta. Il valore di default è true.

Metodo Delete

Elimina una cartella.

Sintassi

```
OggettoFolder.Delete[(forza)]
```

Parametro	Descrizione
forza	Opzionale. Se impostato a true, indica che verranno cancellati le cartelle anche di sola lettura. Il valore di default è false.

Metodo Move

Sposta una cartella da una posizione ad un'altra.

Sintassi

```
OggettoFolder.Move(destination)
```

Parametro	Descrizione
destinazione	Dove spostare la cartella

Metodo OpenAsTextStream

Apre un file in una cartella e restituisce un oggetto di tipo TextStream per accedere al file.

Sintassi

```
OggettoFolder.CreateTextFile(nome[,sovrascrivi])
```

Parametro	Descrizione
sovrascrivi	Come aprire il file. 1 = In lettura 2 = In scrittura 8 = Per scrivere alla fine del file

Oggetto Dictionary

Oggetto Dictionary

E' usato per memorizzare delle informazioni nel formato: nome/valore (referenziati come key ed item).

Proprietà

Proprietà	Descrizione
CompareMode	Imposta o restituisce un metodo di comparazione per le chiavi di un oggetto Dictionary
Count	Restituisce il numero di coppie key/item in un oggetto Dictionary
Item	Imposta o restituisce il valore di un oggetto Dictionary
Key	Inserisce un nuovo valore per una chiave in un oggetto Dictionary

Proprietà CompareMode

Imposta o restituisce un metodo di comparazione per le chiavi di un oggetto Dictionary.

Sintassi

```
DictionaryObject.CompareMode[=compare]
```

Parametro	Descrizione
compare	Specifica la modalità di confronto. Può prendere 0 = confronto binario 1 = confronto testuale 2 = confronto database

Proprietà Count

Restituisce il numero di coppie key/item in un oggetto Dictionary.

Sintassi

```
DictionaryObject.Count
```

Proprietà Item

Imposta o restituisce il valore di un oggetto Dictionary.

Sintassi

```
DictionaryObject.Item(chiave)[=nuovoElemento]
```

Parametro	Descrizione
chiave	La chiave associate all'elemento
nuovoElemento	Il valore associate alla chiave

Proprietà Key

Imposta un nuovo valore per una chiave in un oggetto Dictionary.

Sintassi

```
DictionaryObject.Key(chiave)=nuovaChiave
```

Parametro	Descrizione
chiave	Il nome della chiave da cambiare
nuovaChiave	Il nuovo nome della chiave

Metodi

Metodo	Descrizione
Add	Aggiunge una coppia key/item ad un oggetto Dictionary

Exists	Restituisce true se una specifica chiave esiste in un oggetto Dictionary
Items	Restituisce un array contenente tutti gli elementi di un oggetto Dictionary
Keys	Restituisce un array contenente tutte le chiavi di un oggetto Dictionary
Remove	Rimuove una coppia key/item da un oggetto Dictionary
RemoveAll	Rimuove tutte le coppie key/item da un oggetto Dictionary

Metodo Add

Aggiunge una coppia key/item ad un oggetto Dictionary.

Sintassi

```
DictionaryObject.Add(chiave,elemento)
```

Parametro	Descrizione
chiave	Il valore della chiave
elemento	L'elemento associato alla chiave

Metodo Exists

Restituisce true se una specifica chiave esiste in un oggetto Dictionary.

Sintassi

```
DictionaryObject.Exists(chiave)
```

Parametro	Descrizione
chiave	Il valore della chiave da cercare

Metodo Items

Restituisce un array contenente tutti gli elementi di un oggetto Dictionary.

Sintassi

```
DictionaryObject.Items
```

Metodo Keys

Restituisce un array contenente tutti gli elementi di un oggetto Dictionary.

Sintassi

```
DictionaryObject.Keys
```

Metodo Remove

Rimuove una coppia key/item da un oggetto Dictionary.

Sintassi

```
DictionaryObject.Remove(chiave)
```

Parametro	Descrizione
chiave	La chiave associata all'elemento da rimuovere

Metodo RemoveAll

Rimuove tutte le coppie key/item da un oggetto Dictionary.

Sintassi

```
DictionaryObject.RemoveAll
```

Appendice A: errori ASP

HTTP 500 internal server error

Da Internet Explorer andare in Strumenti > Opzioni Internet ...> Avanzate e deselezionate "Mostra messaggi di errore HTTP brevi". A questo punto dovreste vedere l'esatto messaggio di errore e comportarvi di conseguenza.

Microsoft OLE DB Provider for ODBC Drivers error '80004005' [Microsoft][ODBC Microsoft Access Driver] Operation must use an updateable query.

Per risolvere questo problema devono essere abilitati i permessi di scrittura nella directory dove è contenuto il database Access per l'utente IUSER_<severname>.

Microsoft OLE DB Provider for ODBC Drivers error '80004005' [Microsoft][ODBC Microsoft Access Driver] Cannot update. Database or object is read-only.

Per risolvere questo problema devono essere abilitati i permessi di scrittura nella directory dove è contenuto il database Access per l'utente IUSER_<severname>.

Microsoft OLE DB Provider for ODBC Drivers error '80040e09' [Microsoft][ODBC Microsoft Access Driver] Cannot modify the design of table 'Tbl_Anagrafica'. It is in a read-only database.

Per risolvere questo problema devono essere abilitati i permessi di scrittura nella directory dove è contenuto il database Access per l'utente IUSER_<severname>.

Microsoft OLE DB Provider for ODBC Drivers error '80004005' [Microsoft][ODBC Microsoft Access Driver] General error Unable to open registry key 'Temporary (volatile) Jet DSN for process 0x4d4 Thread 0x748 DBC 0x84dbc84 Jet'.

Controllate il percorso del vostro database nella Stringa di connessione.

Microsoft JET Database Engine error '80040e14' Syntax error in INSERT INTO statement.

Controllare la sintassi nella stringa SQL che fa la Insert.

Verificare anche la vostra stringa di connessione: Access: Driver Di Accesso Di DRIVER={Microsoft (* mdb) }; DBQ =... MSSQL: DRIVER={SQL Server};Server =... "

ADODB.Connection error '800a0e7a' ADO could not find the specified provider.

Questo problema deriva dall'MDAC (Microsoft Data Access Components) installato sul Web-Server.

I driver sono inesistenti o danneggiati, scaricate l'ultima versione dell'MDAC.

Microsoft VBScript runtime error '800a01b6' Object doesn't support this property or method: 'objRs.fields(...).AppendChunk'

Questo problema deriva dall'MDAC (Microsoft Data Access Components) installato sul Web-Server.

I driver sono inesistenti o danneggiati, scaricate l'ultima versione dell'MDAC.

Microsoft OLE DB Provider for ODBC Drivers error '80004005' [Microsoft][ODBC Microsoft Access Driver] Cannot open database '(unknown)'. It may not be a database that your application recognizes, or the file may be corrupt.

Se il tuo DataBase risulta essere danneggiato agisci in questo modo. Fai una copia di Backup del database. Apri il db Access e vai in Strumenti > Utilità Database > Compatta e Ripristina Database Se il database risultasse ancora danneggiato, riparti dall'ultimo backup... a proposito lo hai fatto il backup ???

Borsoft

Indice