Appendice: Oggetto **Response**

Dell'oggetto **Response** abbiamo esaminato solo il metodo **Write** (che scrive sull'output della pagina), quindi è d'obbligo una panoramica sugli altri metodi:

Response.AddHeader Aggiunge un header specifico alla pagina, di cui è possibile settare nome e valore

Response.AppendToLog questo metodo aggiunge una stringa di non più di 80 caratteri al file di log del server cioe' di IIS (Internet Information Server: web server su Windows NT)

Response.BinaryWrite questo metodo permette di visualizzare dei file binari (es. le immagini) cioe' invia all'output il parametro passato senza applicare conversioni di charset

Response.Clear ripulisce l'output, ma solo se bufferizzato

Response.End interrompe l'elaborazione ASP in corso (e restituisce l'output se bufferizzato).

Response.Flush invia all'istante l'ouput bufferizzato

Response.Redirect serve per inviare i clients ad un altro URL cioe' invia un messaggio di redirect al browser, spostandolo su un differente URL. Se il buffer e' attivato ripulirà il buffer e inserirà il comando di redirect al posto dei dati cumulati.

Response. IsClientConnected verifica durante l'esecuzione della pagina se il client è ancora connesso

Response.Pics Aggiunge un valore al PICS label dell'header

Con la proprietà **Response.ContentType** si può specificare il tipo di dato inviato con protocollo http.

La sintassi prevede di impostare sia il tipo del contenuto generale sia di quello specifico.

Per default tale proprietà imposta il tipo ed il sottotipo come text/HTML cioè il contenuto è di tipo generale *text* ed in particolare è scritto in linguaggio *HTML*

Metodi dell'oggetto **Response**

Response.BinaryWrite ()

permette di visualizzare dei file binari (es. le immagini) cioè invia all'output il parametro passato senza applicare conversioni di charset

sintassi: Response.BinaryWrite(Data)

The BinaryWrite method writes nonstring data (binary data) to the current HTTP output without any character set conversion.
This method can use the output of the Request.BinaryRead method. For example, you could retrieve an image from a database with Request.BinaryRead and send it to a client using Response.BinaryWrite.
There is one mandatory argument:

Data The Data argument is the data to send to the HTTP output.

Esempio:

```
----------File1.asp----------
<HTML>
<HEAD>
</HEAD>
<BODY>
<form action="File2.asp" method="POST">
Name:<input type="Text" name="name" maxlength="30"><BR>
Age:<input type="Text" name="age" maxlength="10"><BR>
Sex:<input type="Text" name="Sex" maxlength="10"><BR>
<input type="Submit" name="submit" value="submit"><BR>
</form>
</BODY>
</HTML>
```

```
----------File2.asp----------
<%
bytecount = Request.TotalBytes
binread = Request.BinaryRead(bytecount)
Response.BinaryWrite binread
%>
```

Output:
name=paola&age=49&Sex=F&submit=submit
oppure
name=Jane&age=25&Sex=Female&submit=submit

Response.Clear ()

ripulisce l'output, ma solo se bufferizzato

sintassi: Response.Clear

The Clear method clears (erases) any buffered HTML output. It does not erase the response headers, only the body. If the buffer does not exist, because Response.Buffer was set to False, a runtime error will occur.
Note that in ASP version 2.0, the default for Response.Buffer is set to False, however in ASP version 3.0, the default for Response.Buffer is set to True.

Esempio:

```
<% Response.Clear %>
```

Response.End ()

interrompe l'elaborazione ASP in corso (e restituisce l'output se bufferizzato).

sintassi: Response.End

The End method orders the web server to stop processing the script. The current results are returned and no further processing occurs. If Response.Buffer is set to True, Response.End will flush the buffer and then end. Note that in ASP version 2.0, the default for Response.Buffer is set to False, however in ASP version 3.0, the default for Response.Buffer is set to True.
In the example, the second Response.Write will not be displayed in the output.

Esempio:

```
<%
Response.Write "Hello World"
Response.End
Response.Write "Is this the End?"
%>

<!-- Output: Hello World  -->
```

Response.Flush ()

invia all'istante l'ouput bufferizzato

sintassi: Response.Flush

The Flush method immediately sends all current buffered page content to the client if the Response.Buffer property is set to True.
If the buffer does not exist, because Response.Buffer is set to False, a runtime error will occur.
Note that in ASP version 2.0, the default for Response.Buffer is set to False, however in ASP version 3.0, the default for Response.Buffer is set to True.

Esempio:

```
<% Response.Flush %>
```

Response.Redirect ()

serve per inviare i clients ad un altro URL cioè invia un messaggio di redirect al browser, spostandolo su un differente URL. Se il buffer e' attivato ripulirà il buffer e inserirà il comando di redirect al posto dei dati cumulati.

sintassi: Response.Redirect(URL)

The Redirect method stops processing the current script and attempts to connect the client to a different URL. This is accomplished by adding an HTTP redirection header to the output stream that is being sent from the server to the client. Unfortunately, if page content has already been sent to the client and if a proxy server lies between the server and the client, an error message can be generated. Therefore it is advisable to set Response.Buffer to true and to call Response.Clear just before calling Redirect.

Note in ASP 2.0, the buffering default setting is false and in ASP 3.0, the buffering default setting is true.
There is one mandatory argument:

URL	The URL argument is the Uniform Resource Locator (URL) that the browser is redirected to.

Esempio:

----------File3.asp----------

```
<% Response.Buffer = true %>
<HTML>
<BODY>
<%
Response.Write "This is File3.asp and switching to File4.asp"
rem Response.Clear
Response.Redirect "File4.asp"
%>
</BODY>
</HTML>
```

----------File4.asp----------

```
<HTML>
<BODY>
<%
Response.Write "This is File4.asp"
%>
</BODY>
</HTML>
```

Output:
File3 is written and then the browser will load File4:
----------File3.asp----------
This is File3.asp and switching to File4.asp (se manca Response.Redirect "File4.asp")

----------File4.asp----------

Response.Write(Variant)

The Write method writes any specified string to the HTTP output.
There is one mandatory argument:

Variant The Variant argument is the data to be written as a string. A variant is a VBScript data type and includes characters, integers, and strings. The variant cannot contain the character combination of %>.

Esempio:

```
<% Response.Write "Hello World"
   Response.Write "<BR>"
%>

<!--
Output:
Hello World<BR>
-->
```

Response. IsClientConnected ()

verifica durante l'esecuzione della pagina se il client è ancora connesso

The IsClientConnected property determines if the client has disconnected from the server since the last Response.Write. This property is particularly useful since it will prevent the server from continuing to attempt to pass information after an unexpected disconnect.
In ASP 2.0, you must first attempt to pass information to the client in order to use this
property. However, in ASP 3.0 you can check the response of using this property before passing any content to the client. (Thus, a simple test will prevent the server from performing unneccesary work.)
In the example code, the output will be a list of all the book titles that are in the database. However, if the user stops the connection to the server, then the only a partial list of titles will be created.

Esempio: (da creare **DB** con **nome books** ed anche l'unica **tabella** di nome **books**):

```
<%
set connDB=server.createobject("adodb.connection")
connDB.Open "books", "", ""
mySQL="select * from books"
Set rsBookSrch = Server.CreateObject("ADODB.Recordset")
rsBookSrch.Open mySQL, connDB, adOpenStatic, adLockPessimistic
%>
<HTML><HEAD></HEAD>
<BODY>
<% Do until (rsBookSrch.eof or Response.IsClientConnected=false)
     Response.Write rsBookSrch.Fields("Title") %> <br>
<% rsBookSrch.MoveNext
    Loop
    rsBookSrch.Close
    connDB.Close  %>
</BODY>
</HTML>
```

Proprietà:

PROPERTY: **Response.ContentType**

The ContentType property specifies the HTTP content type, which includes a type/subtype, for the response header. The type is the general content, and the subtype is the specific content. The default is text/HTML.
This example code will produce a 2 X 2 Excel Spreadsheet in a browser, if Excel is installed on the client machine.

Code:

```
<% Response.ContentType = "application/vnd.ms-excel" %>
<HTML>
<HEAD></HEAD>
<BODY>
<TABLE>
<TR>
<TD>2</TD><!--- Cell : A1 --->
<TD>4</TD><!--- Cell : B1 --->
</TR>
<TR>
<TD>5</TD><!--- Cell : A2 --->
<TD>6</TD><!--- Cell : B2 --->
</TR>
</TABLE>
</BODY>
</HTML>   <!-- funziona creando foglio Excel in locale -->
```

Oggetto **Request**

L'oggetto Request, al contrario di Response, è utilizzato per recuperare informazioni di vario genere, vengano esse dal client che richiede una pagina o siano relative al server stesso. Come per l'oggetto Response vediamo il riassunto veloce dei metodi, delle proprietà e delle collection:

Collections:

ClientCertificate
Cookies
Form
QueryString
ServerVariables

Properties: TotalBytes

Methods: BinaryRead

La sintassi generica dell'oggetto Request è: **Request.**[Collection|Property|Method](“variable”)

Vi possono essere varie collection, ed infatti negli esempi si può osservare l'utilizzo della collection **QueryString**, che accetta le stringhe provenienti da un form oppure inserite manualmente, formate da un punto interrogativo seguito da un testo oppure l'utilizzo della collection **Form** che recupera i dati del form inviati con metodo post

La collezione Request.Cookies

Serve a recuperare un cookie, la sintassi è semplice

```
Request.Cookies("NOME_DEL_COOKIE")
```

La collezione Request.ServerVariables

La collezione Request.ServerVariables si occupa del fornire delle informazioni relative all'utente connesso ad una pagina o del restituire i valori di alcune variabili d'ambiente. L'utilizzo è classico, basato sulla nozione puntata e sulle chiavi:

variabile = Request.ServerVariables("nome_chiave");

Le possibili chiavi sono molteplici ed alcune di esse dipendono strettamente dal server che elabora la pagina.

Per identificare il nome del navigatore che il client sta usando in quel momento si è usato il metodo ServerVariables che permette di richiedere al server una delle variabili di sistema come ad esempio HTTP_USER_AGENT.

Ecco altre **variabili del server**:

HTTP_REFERER	URL della pagina indicato dall'utente al documento del sito
HTTP_ACCEPT_LANGUAGE	Lingua accettata dal browser
REMOTE_ADDR	IP del client.
ALL_HTTP	Tutte le variabili HTTP_*

Le **chiavi** più ricorrenti e più frequentemente utilizzate:

Chiave	Utilizzo
ALL_HTTP	Restituisce tutti gli header HTTP associati alla richiesta del client
HTTP_<HEADER>	Sostituendo <HEADER> con il nome di uno specifico header appartenente alla richiesta HTTP è possibile conoscerne il contenuto
HTTP_CONTENT_TYPE	Restituisce il tipo di dato dei contenuti inviati dal client. Si utilizza con le richieste che trasmettono informazioni a partire dal client, come con i form HTML inviati con metodo POST o GET
HTTPS	Restituisce ON se la richiesta arriva attraverso canali sicuri (SSL), OFF in caso contrario
LOCAL_ADDR	Restituisce l'indirizzo del Server
QUERY_STRING	Restituisce l'intera query string associata all'URL
REMOTE_ADDR	Restituisce l'indirizzo IP dell'host che ha inviato la richiesta
REMOTE_HOST	Restituisce il nome dell'host che ha inviato la richiesta
SCRIPT_NAME	Restituisce il percorso virtuale dello script in esecuzione
SERVER_NAME	Restituisce il nome del server, il suo alias DNS o il suo indirizzo IP
SERVER_PORT	Restituisce il numero della porta alla quale la richiesta è stata inviata
SERVER_SOFTWARE	Restituisce il nome e la versione del server software che ha risposto alla richiesta

NB: altre variabili non implementabili su alcuni server *free*

HTTP_UA_PIXELS	Risoluzione del monitor con il browser del client
HTTP_UA_COLOR	Palette dei colori del monitor dell'utente
HTTP_UA_OS	Sistema Operativo del browser dell'utente
HTTP_CONTENT_LENGTH	Restituisce la lunghezza dei contenuti inviati dal client
DATE_LOCAL	Data corrente nella fascia oraria locale

Metodo dell'oggetto **Request**

METHOD **BinaryRead (Count) Method**
The BinaryRead method retrieves the data that was sent to the server from the browser as part of a POST request, and returns the number of bytes read.

Da ***Quick Refs – ASP*** http://www.devguru.com/technologies/asp/9112

e tutorial ***ASP: la guida introduttiva*** di Carlo Pelliccia
http://www.aspcode.it/tutorials/tutorials.asp?action=show_tut&idx=6

vedi anche gli ***Esempi*** proposti:
http://www.w3schools.com/asp/asp_ref_response.asp 

Write text with ASP How to write text with ASP
Format text with HTML tags in ASP How to combine text and HTML tags with ASP
Redirect the user to a different URL How to **redirect** the user to a different URL
Show a random link How to create a random link
Controlling the buffer How to control the buffer
Clear the buffer How to clear the buffer
End a script in the middle of processing and return the result How to end a script in the middle of processing.
Set how many minutes a page will be cached in a browser before it expires How to specify how many minutes a page will be cached in a browser before it expires
Set a date/time when a page cached in a browser will expire How to specify a date/time a page cached in a browser will expire
Check if the user is still connected to the server How to check if a user is disconnected from the server
Set the type of content How to specify the type of content
Set the name of the character set How to specify the name of the character set