ASP

Web statico e Web interattivo

- In principio il Web era una semplice collezione di pagine HTML statiche collegate tra di loro tramite link ipertestuali.
- La necessità di una maggiore interattività tra l'utente e il server Web, nata soprattutto nel momento in cui grandi aziende hanno visto nella rete delle reti, un nuovo veicolo commerciale, ha indirizzato gli sforzi nello sviluppo di strumenti per rendere il Web sempre più "dinamico" (Es. motori di ricerca)

  ⇒ Nasce lo standard CGI (Common Gateway Interface)

Interazione tra browser e applicazione CGI

- il browser invia al server Web una richiesta facendo riferimento al nome dell'applicazione seguita da una serie di parametri;
- il server Web riconosce che la richiesta deve essere indirizzata all'applicazione specificata e ne attiva un'istanza passandole opportunamente i parametri
- l'applicazione effettua l'elaborazione in base ai parametri acquisiti e crea sullo standard output un flusso di dati che costituisce la pagina di risposta per il browser.
- il server Web cattura il flusso di dati generato dall'applicazione CGI e la trasforma in una risposta HTTP

Client-side scripting

L'interfaccia CGI tuttavia presenta dei limiti (Es. Invio di dati errati)

SOLUZIONE: Netscape prima e Microsoft dopo, hanno pensato di permettere ai loro browser, di interpretare particolari linguaggi, detti linguaggi di scripting (JavaScript, Jscript e VBscript), che permettono al client di effettuare alcune semplici elaborazioni.

Script

- Uno script è composto da una serie di comandi di script
- Un comando di script istruisce la macchina a fare qualcosa
  - Assegnare un valore ad una variabile
  - Eseguire un'operazione
  - Stampare qualcosa

Linguaggi per il web

Per fare chiarezza elenchiamo i 4 tipi di linguaggi con cui si ha a che fare nel web:

- Linguaggi di marcatura (HTML)
- Linguaggi compilati (C)
- Linguaggi semicompilati (Java)
- Linguaggi interpretati (scripts)

Linguaggi di scripting

- I linguaggi di scripting sono una via di mezzo tra un sistema di marcatura ed un linguaggio di programmazione
  - Un *sistema di marcatura* ha il compito di formattare un testo
  - Un *linguaggio di programmazione* dà una serie completa di istruzioni al computer
- Un *linguaggio di scripting* si avvicina di più ad un linguaggio di programmazione che ad un sistema di marcatura
- Le regole di un linguaggio di scripting sono comunque meno rigide e complicate di quelle di un linguaggio di programmazione

Esecuzione di uno script

- Affinchè uno script possa essere eseguito una serie di comandi vengono inviati allo scripting engine
- Script significa "copione", "sceneggiatura": una riga viene letta, interpretata ed eseguita e poi si passa alla successiva, come da copione
- I motori di scripting (scripting engine) sono degli oggetti COM (Component Object Model) che processano e interpretano gli script

Ambienti per lo scripting

- Alcuni sistemi forniscono un host environment per i motori di scripting ossia garantiscono un ambiente dove gli script possono essere processati
- Affinchè uno script possa essere effettivamente processato in un ambiente, il motore di scripting del linguaggio di scripting deve essere installato sul web server
- Al contrario nella maggior parte dei casi il browser client non necessita supporto per lo scripting dato che tutto il carico è lasciato al lato server

ASP

- ASP è l'acronimo per Active Server Pages
- Le pagine sono "attive" poichè al loro interno custodiscono dei comandi che ne possono diversificare il contenuto finale
- Così le pagine inviate al client possono essere differenti da caso a caso a seconda delle scelte fatte dal client stesso o di situazioni contingenti
- Viene fornita una pagina HTML standard che offre di conseguenza il vantaggio, di essere indipendente dal tipo di browser utilizzato.

Introduzione ad ASP (1)

Il successo ottenuto dal client-side scripting ha portato in breve tempo allo sviluppo di linguaggi di scripting anche per il lato server. Nascono:

- Borland IntraBuilder dove è Javascript il linguaggio server-side utilizzato.
- La Microsoft, rilasciando la versione 3.0 di Internet Information
- Server (IIS), ha introdotto sul mercato degli scripting server-side, la tecnologia Active Server Pages (ASP) Ambiente di interpretazione di scripting che gira sul server.

Introduzione ad ASP (2)

- L'idea della tecnologia ASP è quella di sfruttare la tecnologia COM (Common Object Model), cioè sfruttare tutte le risorse che il server Microsoft ha a disposizione e coinvolgendo anche i linguaggi di scripting come Jscript e Vbscript.

A cosa serve ASP e come funziona

Le pagine di un sito non sono più una semplice collezione di documenti HTML ma un insieme di pagine contenenti codice script interpretabile dal server Web.

- Il server Web effettua le elaborazioni specificate prima di inviare la pagina HTML risultante al browser che l'ha richiesta.

ASP un linguaggio?

ASP non è un linguaggio di scripting

Linguaggi di scripting sono:

- Visual Basic Script
- JavaScript
- PERL

VBScript

- Veloce e leggero subset di Visual Basic
- Altamente integrato con i www browser
- Puro interprete: processa il codice sorgente incorporato nell'HTML
- Non produce applets stand-alone ma viene usato per dotare di interattività i documenti HTML

JavaScript

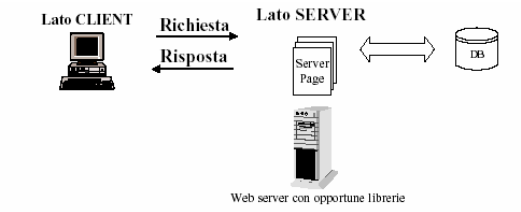
- Nel 1994 la Netscape sviluppa LiveScript
- Nel 1995 nasce la collaborazione con la Sun per sviluppare in parallelo JavaScript
- Nel 1996 Microsoft adotta un linguaggio molto simile: Jscript per problemi di copyright

PERL

Processing Extraction Report Language

- Linguaggio di scripting nato per l'automatizzazione di alcuni task specifici
- Diffusione enorme nell'ambito della manutenzione di attività di un server
- Utilizzato in particolare per la scrittura di procedure CGI
- Il codice non viene compilato in codice macchina

Funzionamento di un'applicazione ASP (1)



- La caratteristica principale di ASP e dei comandi script in generale è di garantire la generazione di codice HTML "on the fly" cioè al volo
- Quando il file di estensione asp viene chiamato dal browser il processo a cui viene dato inizio produce in risposta del codice HTML

Funzionamento di un'applicazione ASP (2)

- Il client richiede una pagina ".asp" al server. Il server invoca ASP per elaborarla
- Vengono effettuati il parsing, il controllo sintattico e la compilazione della pagina
- Il codice viene eseguito
- Il risultato e viene immerso nel codice HTML
- La pagina HTML risultato viene inviata al client

ASP: prime accortezze

Per creare uno script ".asp" disponibile agli utenti del web è necessario:

- Salvare il file in una directory appartenente al web publishing
- Avere il permesso di esecuzione degli script

La sintassi ASP

- Una pagina ASP può essere scomposta in tre parti:
  1) Testo
  2) Marcatori HTML
  3) Comandi script
- In un documento con estensione ".asp" è consentito utilizzare variabili, cicli e istruzioni di controllo.
- Una pagina può essere costituita solamente da codice ASP, oppure avere ASP immerso nel codice HTML (in entrambi i casi l'estensione deve essere ".asp").

La sintassi ASP: esempio

- Il codice ASP è sempre delimitato dai due marcatori <% e %>.

ESEMPIO:

Assegnazione di un valore ad una variabile

Es: <% eta=26 %>

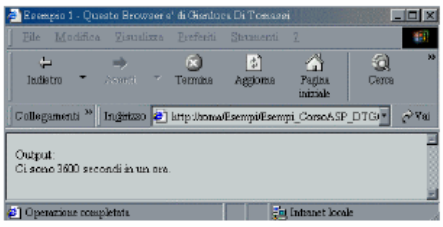
Espressione di output

Es: <%= eta %>

Viene inviato il valore corrente della variabile al browser

Un semplice esempio

```
<HTML>
<BODY>
Output:<BR>
<%
  intS = 60*60
%>
Ci sono
<%
  Response.Write intS
%>
secondi in un'ora.
</BODY>
</HTML>
```



Linguaggi di scripting supportati da ASP

- Possibilità di utilizzare più linguaggi di scripting all'interno di un'applicazione ASP.
- ASP supporta in modo nativo due linguaggi di scripting, VBScript e Jscript.
- Se non diversamente specificato, ASP interpreta il codice script che trova all'interno dei marcatori <% e %> come codice VBScript, il linguaggio predefinito.

Come dichiarare il linguaggio di scripting?

- a livello di sito (quindi di Web server), cioè per tutte le applicazioni ASP gestite da IIS;
- a livello di file (quindi di singola applicazione), cioè per tutte le pagine che compongono un'applicazione;
- a livello di funzione

NOTARE:

La procedura per il settaggio del linguaggio primario dipende dal tipo di linguaggio da introdurre:

- Supporta la sintassi object.method
- Non supporta la sintassi object.method

Specificare il linguaggio a livello di file se object.method" supportato

Tramite la direttiva @ LANGUAGE è possibile specificare il linguaggio da utilizzare all'interno di una determinata pagina ASP.

Accortezze sulla direttiva :

- Questa deve essere la prima linea del file
- Non va inserita in un file incluso
- Inserire uno spazio tra @ e LANGUAGE
- Non inserire altri elementi nella linea

ESEMPIO: <% @ LANGUAGE = "JScript" %>

il motore ASP interpreterà il codice contenuto all'interno di quella pagina come codice JScript.

Specificare il linguaggio al livello di funzione

Si utilizza una versione arricchita del tag <SCRIPT>:

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

L'attributo RUNAT specifica a che livello deve essere interpretato lo script; se non viene impostato, lo script è destinato all'interpretazione da parte del browser.

- **VANTAGGIO** L'uso di RUNAT fa si che il codice sorgente non è mai presente nella pagina html che viene spedita al navigatore dal server.

Inclusione di file

Server-side includes è un meccanismo che consente di inserire informazioni in un file prima che venga processato

Due modi per descrivere il percorso del file:

- Virtual: indica un percorso che inizia in una directory virtuale

  Es: <!—#INCLUDE VIRTUAL="nome"-->
- File: indica un percorso relativo a partire dalla directory che include il file ".asp"

  Es: <!--#INCLUDE FILE="nomefile.est" -->

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

Regole per l'inclusione di file

- Un file incluso può includere altri file
- Si può includere lo stesso file più di una volta
- Attenzione ai cicli infiniti di inclusione
- Un file non può includere se stesso
- Non è possibile aprire un delimitatore in un file ".asp" e chiuderlo in un file incluso
- Non è possibile inserire un delimitatore dentro un altro se il primo non è stato ancora chiuso
- non è possibile usare un comando di script per creare il nome di un file incluso

Es:
```
<% name=(intestazione & ".inc") %>
<!—include file="<% name %>" -->
```

Commenti nelle pagine ASP

- In VBScript le linee possono essere commentate con il REM del BASIC o con gli apostrofi

  ESEMPIO: <% REM Sto per chiamare CiaoVBS
  Call CiaoVBS()
  Call Ciao() 'chiamata Ciao
  %>

- In JavaScript le linee possono essere commentate con //

  ESEMPIO: <% // Sto per chiamare CiaoJS
  Call CiaoJS()
  Call Ciao() %>

Forzare la dichiarazione delle variabili

- Per poter forzare a dichiarare tutte le variabili utilizzate all'interno della pagina ASP si utilizza: **Option Explicit**

Quindi se si aggiunge **<% Option Explicit %>** subito dopo la riga **<%@ Language)…%>,**

si fa in modo che VBScript richieda dichiarazioni esplicite di tutte le variabili

Gli oggetti interni ASP

Gli oggetti sono costituiti dagli elementi che li descrivono e dalle operazioni che possono essere effettuate utilizzandoli.

- Gli elementi che descrivono gli oggetti sono denominati proprietà
- Le operazioni che possono essere effettuate utilizzandoli sono denominate metodi

Oggetti direttamente disponibili in ASP (1)

ASP mette a disposizione per la gestione dell'output, dell'input e delle operazioni che il client vuole o deve compiere sul server diversi oggetti predefiniti.

- Application:utilizzato per condividere informazioni tra numerosi client che esplorano lo stesso gruppo di pagine
- Session: informazioni sul singolo utente che accede a un'applicazione
- Request: utilizzato per recuperare informazioni passate dal client al server dal browser
- Response: utilizzato per inviare manda al client il codice HTML risultante

Oggetti direttamente disponibili in ASP (2)

- Server: mette a disposizione alcune funzionalità del server
- ObjectContext: permette di gestire le transazioni (collega la pagine ASP e Microsoft Transaction Server)
- ASPError: consente di ottenere informazioni relative agli errori degli script all'interno delle pagine

Ogni oggetto, ha una collezione di metodi, proprietà ed eventi

Collection

Rappresenta un insieme di coppie di nomi o valori

La stringa d'interrogazione:
"?nome=Gianluca&cognome=DiTomassi"

Contiene due coppie di nomi e valori, la prima coppia ha il nome "nome" e il valore "Gianluca". La seconda ha il nome "cognome" e il valore "Di Tomassi"

Questi dati (vedremo) vengono memorizzati nella collection QueryString dell'oggetto Request

Altri oggetti utili

Altri oggetti utili nella gestione delle applicazioni:

- Oggetto ADO: Gestisce le collezioni della tecnologia ADO. Nel seguito lo useremo per le operazioni con i database
- Oggetto BrowserCap: Permette di riconoscere il browser e le impostazioni (ad esempio la risoluzione,il numero di colori etc) che il visitatore sta utilizzando, adattando il sito al software dell'utente
- Oggetto FileSystem: Permette di manipolare files e directories
- Oggetto CDONTS : Permette di inviare una mail

L'oggetto Response (1)

Consente di inviare informazioni al browser e di controllare il modo in cui vengono inviate.

Sintassi oggetto Response:

Response.collection/property/metodo

- Le *collection* dell'oggetto Response sono:

  Cookies: Si usano per creare i cookie
- Le *property* per l'oggetto Response sono:

  CacheControl: determina se il proxy riesce a mantenere in cache il contenuto delle pagine ASP

  CharSet: Specifica il set di caratteri utilizzato

L'oggetto Response (2)

Expires: Specifica il tempo che deve trascorrere prima che una pagina presente nella cache del browser venga rimossa

Esempio:

<% Response.Expires = numero %> la pagina scade dopo numero minuti

ExpiresAbsolute: Specifica la data e l'ora quando la pagina deve essere cancellata dalla cache del browser

<% Response.ExpiresAbsolute = Data Time %>

L'oggetto Response (2)

ContentType: E' una stringa che descrive i contenuti supportati dal documento

Esempio: <% Response.ContentType = "text/plain" %>

Buffer: Gestisce la bufferizzazione del codice da inviare all'utente. (occorre settarlo dopo l'Option Explicit)

- Per default è settato a FALSE
- Se viene settato a TRUE tutti gli script saranno processati prima che qualsiasi cosa venga inviata al browser

- I *metodi* dell'oggetto Response sono:

AddHeader: pone il valore tra i tag <Header> ad un valore value

L'oggetto Response (3)

Write: invia info all'utente (scrive una stringa in output). La stringa scritta non può contenere il tag "%>", se è necessario scriverlo utilizzare "%\>"

<% Response.Write(expr) %> è equivalente a **<%=expr%>**

Esempi di utilizzo:

Response.Write("<IMG SRC='banner.gif'>")

Redirect: permette di reindirizzare il browser ad un altro URL n In questo modo è possibile fare dei controlli sulla navigazione degli utenti e pilotarla

- E' possibile effettuare la Redirect solo se il contentuto del file non è già stato inviato al browser oppure utilizzando il buffering

<% Response.Redirect "homepage.asp" %>

L'oggetto Response (4)

- ***AppendToLog***: Aggiunge una stringa al log del Web server per una interrogazione
- ***Clear***: cancella qualsiasi codice HTML nel buffer di trasmissione
- ***End***: Viene interrotta l'esecuzione della pagina ASP e restituisce il risultato corrente (cioè se il buffering è attivato e sono presenti dati nel buffer, questi vengono inviati)
- ***Flush***: Invia immediatamente il contenuto del buffer di trasmissione anche se si sta bufferizzando (utile quando si desidera inviare la maggior parte possibile dell'output al client.

L'oggetto Request (1)

Consente di accedere ai dati che il client ha inviato quando ha richiesto la pagina corrente

Sintassi oggetto Request:

Request.collection/property/metodo

Le collection dell'oggetto Request sono:

ClientCertificate: I valori dei campi memorizzati nei certificati del client, che vengono mandati durante una richiesta via HTTP

Cookies: determina il valore dei cookies

Form: contiene tutte le informazioni che un utente inserisce in una form tramite il metodo POST

L'oggetto Request (2)

Esempio:

```
<form action= "send.asp" method="post">
<p> Sito preferito: <select nome= "sito"> <option>
 ….
Il tuo sito preferito è <% Request.Form("sito") %>
```

QueryString: Contiene tutte le informazioni passate come parametro dopo ? nell'URL (quindi passate con metodo GET)

Esempio:

```
<a href= "saluto.asp?nome=Gianluca&eta=27">
Ciao <%= Request.QueryString("nome") %>
hai <%= Request.QueryString ("eta") %> anni
```

L'oggetto Request (3)

ServerVariables: Fornisce informazioni ricavate dall'intestazione http che poi vengono trattate come variabili d'ambiente del Web server

<% Request.ServerVariables(*nomeVarAmbiente*) %>

Variabili d'ambiente	Descrizione
SERVER_PORT	contiene il numero della porta su cui è stata fatta la richiesta
HTTP_ACCEPT_LANGUAGE	contiene il linguaggio del documento
SCRIPT_NAME	è il nome dello script
HTTP_HOST	è il nome del dominio associato all'indirizzo IP
HTTP_USER_AGENT	è lo user agent del browser utilizzato
CONTENT_LENGTH	informazioni sulla lunghezza del pacchetto

L'oggetto Request (4)

QUERY_STRING	La stringa d'interrogazione (equivalente a `Request.QueryString`)
REMOTE_HOST	fornisce l'indirizzo IP
REMOTE_ADDR	fornisce l'indirizzo dell'host remoto
URL (uguale a PATH_INFO)	L'URL della pagina ASP a partire dalla fine di "http:www.WebServer.it/" fino alla stringa d'interrogazione
PATH_TRASLATED	Il percorso fisico completo della pagina ASP attualmente in esecuzione
SERVER_NAME	Il nome del computer del Web server
SERVER_SOFTWARE	Il nome del SW del Web server
APPL_PHISICAL_PATH	L'indirizzo fisico delle dir principali del Web Server
ALL_RAW	Fornisce tutte le informazioni

L'oggetto Request (5)

Le ***property*** per l'oggetto Request sono:

TotalBytes Read-only: Restituisce il numero di byte spediti da un client durante una richiesta al server

I ***metodi*** dell'oggetto Request sono:

BinaryRead: Restituisce i dati spediti al server da un client come parte di un POST

Il parametro *variabile* è una stringa che specifica il valore da utilizzare in una colection o che deve essere usata come input per un metodo o una property.

L'oggetto Request (6)

- Se la variabile non è presente quando si usa una delle 5 collection viste, l'oggetto Request restituisce il valore EMPTY
- Tutte le variabili, possono essere visualizzate direttamente senza il nome della collection mediante l'istruzione:

  Request(*variabile*)
- In questo caso, il sever cerca il valore della variabile, analizzando le collection nell'ordine che segue: QueryString, Form, Cookies, ClientCertificate, ServerVariables.

L'oggetto Server (1)

Fornisce l'accesso ad alcuni strumenti di base sul server

Sintassi oggetto Server:

Server.*property*/*metodo*

Le ***property*** per l'oggetto Server sono:

ScriptTimeout: il tempo massimo, in secondi, che uno script può funzionare,prima che venga "disattivato" dal server

I ***metodi*** dell'oggetto Server sono:

CreateObject: crea un istanza di un componente del server

Esempio: **<% Server.CreateObject(*NomeComponente*) %>**

Execute: esegue una pagina ASP (utile rispetto all'include perché il nome del file può essere generato dinamicamente)

L'oggetto Server (2)

HTMLEncoding: Codifica la stringa argomento in modo che il browser non la interpreti come HTML

Esempio: **<% Server.HTMLEncoding(*Stringa*) %>**

MapPath: Stabilisce una corrispondenza fra il percorso virtuale specificato, sia relativo, sia assoluto, e il percorso fisico.

Esempio: **<% Server.MapPath(*path*) %>**

URLEncode: Applica le regole di encoding, inclusi i caratteri di escape, ad una stringa in modo che essa possa esser posta in una stringa d'interrogazione

GetLastError: Restituisce un'istanza dell'oggetto ASPError che descrive l'ultimo errore avvenuto

L'oggetto Session (1)

- Trasporta i valori richiesti da un singolo client nell'intera sessione, che può essere di diverse pagine.
- Il Web server attiva un'istanza dell'oggetto Session, ogni volta che un utente accede ad una pagina.
- Il server poi, distruggerà l'istanza una volta che l'utente si disconnette o dopo un certo tempo di timeout.
- Attraverso l'uso di Session è possibile memorizzare le preferenze di ciascun utente collegato.

L'oggetto Session (2)

Sintassi oggetto Session :

Session.*collection*.*property*/metodo

Le ***collection*** per l'oggetto Session sono:

Contents: Contiene gli ITEMS che sono stati aggiunti alla sessione con i comandi di script e quindi diverse da un oggetto

StaticObjects: Contiene tutti gli oggetti creati con il tag <OBJECT>

L'oggetto Session (3)

Le ***property*** per l'oggetto Session sono:

CodePage: Il codice della pagina che sarà usato per il mapping simbolico del sito

LCID: L'identificatore locale per la sessione

SessionID: Restituisce un ID di sessione per l'utente connesso nella sessione avviata

Timeout: Indica il tempo, in minuti, prima del Timeout della sessione in corso.

L'oggetto Session (4)

I ***metodi*** dell'oggetto Session sono:

Abandon: Questo metodo, distrugge un oggetto Session e rilascia tutte le risorse tenute dall'oggetto fino a quel momento.

Contents.Remove: Rimuove l'elemento dalla collection Contents

Esempio: <% Session.Contents.Remove(*ele*) %>

Contents.RemoveAll: Rimuove tutti gli elementi dalla collection Contents

L'oggetto Session (5)

Session possiede anche la gestione di **eventi**. Essi sono gestibili nel file global.asa attraverso i seguenti script:

- Session_OnStart: Si verifica quando si crea una nuova sessione
- Session_OnEnd: Si verifica quando termina una sessione, a causa di Abandon o per Timeout

L'oggetto Application(1)

Solo un'istanza dell'oggetto Application è creata per una applicazione ed è condivisa fra tutti i client che accedono a quell' Application:

Application.collection/metodo

Le ***collection*** dell'oggetto Application sono:

Contents: Contiene tutte le variabili dell'applicazione diverse da un oggetto

StaticObjects: Contiene tutti gli oggetti dell'applicazione

L'oggetto Application(2)

I ***metodi*** dell'oggetto Application sono:

Contents.Remove: Rimuove l'elemento dalla collection Contents

Contents.RemoveAll: Rimuove tutti gli elementi dalla collection Contents

Lock: Impedisce a tutti gli altri client di modificare i valori dell'oggetto Application

Unlock: rilascia il bloccaggio e consente a tutti gli altri client di modificare i valori nell'oggetto Application

L'oggetto Application(3)

Application, come Session, possiede anche la gestione di **eventi**.

Essi sono gestibili nel file global.asa attraverso i seguenti script:

Application_OnStart: Si verifica quando si avvia un'applicazione prima di avviare la sessione

Application _OnEnd: Si verifica quando termina l'applicazione, dopo che tutte le sessioni sono terminate

CDONTS: Inviare una mail (1)

Il suo utilizzo è semplice, basta definire un mittente, un destinatario e il corpo del messaggio (che può essere HTML o testo semplice) e procedere all'invio del messaggio

Esempio

```
<%
   'Istanziamo l'oggetto CDONTS
   Set objMail =
   Server.CreateObject("CDONTS.NewMail")
```


CDONTS: Inviare una mail (2)

```
'Definiamo nell'ordine il corpo del messaggio, il mittente,
   il destinatario e il formato, (0 HTML, 1 solo testo)
   objMail.body=‘Ciao Ci vediamo da me alle 17
   objMail.from= ‘bpasolin@csr.unibo.it’
   objMail.to=‘lumini@csr.unibo.it’
   objMail.BodyFormat=0
   'Ora possiamo procedere all'invio del messaggio
   objMail.Send

   'Distruggiamo l'oggetto CDONTS
set objMail=nothing

%>
```

Definire procedure

- Le Active Server Pages hanno la capacità di incorporare diverse procedure di linguaggi di scripting in un singolo file ".asp"
- È possibile definire una propria procedura e chiamarla ripetutamente nei propri scripts
- Le definizioni di procedure devono apparire tra i tag <script> e </script>

Esempio:

```
<SCRIPT RUNAT=Server LANGUAGE=VBScript>
    Sub CiaoVB
    Response.write "ciao da VBScript "
    End Sub
</SCRIPT>
```


Richiamare una procedura

- Per richiamare una procedura bisogna includere il nome della procedura in un comando
- Call è la parola chiave per effettuare la chiamata
- Tra parantesi dopo il nome della procedura possono essere inseriti i parametri della procedura stessa
- Se la parola chiave Call viene omessa allora devono essere omesse anche le parantesi attorno ai parametri

Esempio:

```
<SCRIPT RUNAT=SERVER LANGUAGE=JSCRIPT>
   function CiaoJS() {
        Response.write("ciao da JScript")
   }
</SCRIPT>
<SCRIPT RUNAT=Server LANGUAGE=VBScript>
   Sub CiaoVB
        Response.write "ciao da VBScript"
   End Sub
</SCRIPT>
<html><HEAD><TITLE>Esempio </TITLE></HEAD>
<body>
   <% Call CiaoJS%><BR>
    <% Call CiaoVB%>
</body>
</html>
```


Esempio:

```
<SCRIPT LANGUAGE=Jscript RUNAT=Server>
   function PrintDate(){
        var x
        x = new Date()
        Response.Write(x)
   }
</SCRIPT>
<HTML><HEAD><TITLE>Esempio</TITLE></HEAD>
<BODY><% Call PrintDate %></BODY>
</HTML>
```

Perché definire le procedure ?

Ciò si rende necessario ogni qualvolta si ha bisogno di funzioni o procedure, dato che ASP, non è in grado di definirle.

ASP e i Database

- Un ruolo senza dubbio rilevante di ASP è quello di mediatore tra Web server e un sistema di gestione di database
- Grazie ad ActiveX Data Object (ADO) è possibile accedere a qualsiasi database ODBC o OLEDB compatibile, usufruendo dei vantaggi che questa tecnologia offre, tra i quali la semplicità di accesso ai dati e l'indipendenza dell'applicazione dal tipo di database.

ODBC, IDC e ASP

- Introdotta dalla Microsoft nel 1991, l'interfaccia ODBC permette di far interagire DBMS, sistemi operativi e protocolli di rete anche diversi tra loro.

OBIETTIVO: Interazione con i DBMS tramite un browser.

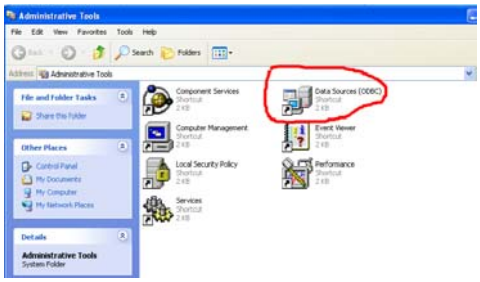
SOLUZIONE MICROSOFT:

MS Web Server, con possibilità di utilizzare:

- L'interfaccia ODBC
- Tecnologia IDC (Internet Database Connector)
- Introduzione di ASP (alternativa ai programmi CGI)

Interfaccia ODBC (1)

All'interno del pannello di controllo occorre selezionare la voce "Administrative Tools" e poi "Sadata Source (ODBC)" (Open Database Connectivity)



Interfaccia ODBC (2)

…e selezionare poi il foglio DSN System

I DSN sono "nomi virtuali" che ADO usa per associare a ogni database utilizzato, un indirizzo fisico sull'hard disk.

I DSN possono essere di tre tipi:

- di sistema; associa un nome a i database presenti sulla macchina con cui si sta lavorando
- di file; permette di associare un nome ad un database, associandolo anche ad un file che può essere presente o meno sulla macchina (ad esempio un database condiviso in una rete intranet)
- di utente; fornisce una lista di tutti i driver ODBC installati

Accedere ai dati con ASP

- Per utilizzare i driver ODBC, ASP si serve della tecnologia ADO (ActiveX Data Objects) sviluppata da Microsoft per il suo linguaggio ActiveX.
- Tramite ADO è comunque possibile connettere una applicazione anche a database non ODBC compatibile quali ad esempio OLE DB (Object Linking and Embedding Database) .
- Per identificare il database su cui lavorare, gli script ADO hanno bisogno che sia specificato un DSN (Data Source Name) che, come già detto, univocamente specifichi il nome e il "luogo fisico" dove il database si trova.

Modello ADO (1)

Il modello ADO contiene sei oggetti:

- Connection: consente di stabilire la connessione all'origine dei dati
- Recordset: consente di operare con i dati contenuti in una tabella (contiene un insieme di righe di una tabella). Può essere usato per leggere o modificare le righe di una tabella o per raccogliere nuovi dati da aggiungere alla tabella
- Error: rappresenta un errore generato dall'origine dati

Modello ADO (2)

- Field: rappresenta una singola colonna in una tabella
- Command: fornisce un altro modo per creare l'oggetto
- Recordset (combina l'oggetto Recordset e Connection)
- Parameters: contiene tutti i parametri necessari al comando

Gli oggetti ADO più importanti

- Connection: la connessione a una sorgente dati
- Recordset: un cursore sul risultato di una query
- Field: i dati di una singola colonna corredati di informazioni

Accedere a un database

1. Aprire una connessione
2. Creare un recordset
3. Estrarre i dati dai campi dei record
4. Chiudere il recordset e la connessione

Aprire una connessione

I metodi più usati e semplici sono quelli che sfruttano il driver ODBC

- perché è facile configurare una sorgente ODBC
- è possibile spostare la posizione del database senza intervenire
- sul codice ASP
- è portabile indipendentemente dall'architettura utilizzata

Connessione a un database

Prima di utilizzare il database è necessario "collegarsi"

L'oggetto Connection è usato per contenere le informazioni del database al quale si vuole accedere

```
Dim objConn
Set objConn =
   Server.CreateObject("ADODB.Connection")
```

Uso di connessione senza DSN (1)

objConn.ConnectionString="DRIVER = Microsoft Access Driver _ (*.mdb) ; " & "DBQ=C:\Cartella\mioDB.mdb"

DRIVER -> Comunica all'oggetto Connection il tipo di database della connessione

DBQ -> Indica il percorso del database sul server. (in alternativa può essere utilizzato Server.MapPath)

UID -> Username per accedere al database

PWD -> Password per accedere al database

Uso di connessione senza DSN (2)

Se si vuole effettuare una connessione diretta OLEDB allora la stringa di connessione sarà la seguente:

"Provider=Microsoft.Jet.OLEDB.3.51; Data Source=path_mio_database"

Uso di connessione con DSN

Dopo aver configurato un nuovo DSN è possibile effettuare una connessione semplicemente facendo riferimento al nome del DSN

objConn.ConnectionString = "DSN=mioDB.dsn"

Come stabilire la connessione

Si utilizza il metodo Open dell'oggetto Connection

```
<%
Dim objConn
Set objConn =
Server.CreateObject("ADODB.Connection")
objConn.ConnectionString = "DSN=mioDB.dsn"
objConn.Open
%>
```

- Solitamente queste istruzioni vengono inserite in un file che viene incluso da tutte le pagine che richiedono accessi al db

Apertura della connessione (1)

La connessione avviene usando l'oggetto ADODB.Connection

```
Set objConn =
   Server.CreateObject("ADODB.Connection")
objConn.Open(SorgenteDatiODBC)
Set objRS = objConn.Execute(QuerySQL)
```

- Con il metodo connessione.Open apriamo materialmente la comunicazione con il database che si trova sul server.
- Con il metodo connection.Execute definiamo il cursore tramite l'interrogazione SQL specificata (sia essa INSERT, UPDATE, DELETE).

Apertura della connessione (2)

La proprietà ConnectionString può essere evitata passando le informazioni direttamente al metodo Open secondo la sintassi:

```
objConn.Open
strConnectionInfo,strUsername,strPassword
```

Come chiudere la connessione

Come tutti gli oggetti al termine del loro utilizzo si deve liberare la memoria ma solo dopo aver chiuso la connessione

```
objConn.Close
Set objConn = Nothing
```


Proprietà della connessione

Esiste una collection nell'oggetto Properties che contiene un'istanza dell'oggetto Property per ogni proprietà supportata dalla connessione

Esempio:

```
<%
For Each objProp in objConn.Properties
        Response.Write objProp.Name & ": " &
        objProp.Value       & "<BR>"
Next
objConn.Close
Set objConn = Nothing
%>
```

Apertura della connessione (1)

```
<%
Response.Expires = 0
Option Explicit
1. Dim objConn, objRS, strQuery, strConnection
2. Set objConn =
   Server.CreateObject("ADODB.Connection")
3. strConnection =
   "DSN=Northwind;Database=Northwind;”
4. strConnection = strConnection &
   "UID=sa;PWD=;”
5. objConn.Open strConnection …%>
```


Apertura della connessione (2)

Nel punto 1. Vengono dichiarate le variabili che si tilizzeranno;

Nel punto 2. Viene creato un oggetto ADODB.Connection

Nel punto 3. e 4. viene definita la stringa per la connessione

Nel punto 5. viene aperta materialmente la connessione

L'oggetto Recordset

- Può essere usato per contenere un sottoinsieme dei record di una tabella o anche tutti i record di una tabella
- Per prima cosa occorre istanziare l'oggetto Recordset

```
Dim objRS
Set objRS =
   Server.CreateObject("ADODB.Recordset")
```

- La prima istruzione dichiara la variabile che conterrà l'oggetto e la seconda lo crea.
- L'oggetto Recordset appartiene al pacchetto ADODB

Il metodo Open (1)

- L'oggetto RecordSet viene popolato di record con l'uso del metodo Open.
- Open accetta diversi gruppi di argomenti:

```
Recordset.Open source, connection, cursortype,
   locktype,
Commandtype
```

Il metodo Open(2)

objRS.Open "miatabella", objConn, , , adCmdTable

adCmdTable vuol dire che la stringa sorgente dovrà assumere il valore del nome della tabella (Quindi se nel db esiste una tabella "miatabella" indicata dall'oggetto Connection chiamato objConn quella tabella sarà copiata nell'oggetto Recordset chiamato objRS). All'interno di objRS ci si potrà spostare in avanti è possibile eseguire solo letture sul db (Non sono specificati gli argomenti per il tipo di cursore e il tipo di blocco

Connessioni implicite

- Si effettuano mediante l'oggetto Recordset
- Con l'oggetto Recordset e con il metodo Open è possibile ottenere una connessione al db senza creare una variabile per contenere un oggetto Connection

```
Dim objRS
Set objRS =
Server.CreateObject("ADODB.Recordset")
objRS.Open = "miatabella", strConnect, , ,
adCmdTable
```

Utilizzo di “adovbs.inc” (1)

- Le costanti definite per l’oggetto Recordset non sono incluse in ASP
- È possibile definirle ogni volta che sono necessarie oppure includere il file “adovbs.inc” che contiene le definizioni per tutte le costanti ADO
- Viene fornito con IIS e PWS si trova (solitamente) nella cartella c:\Program Files\Common Files\System\ado\
- Copiare il file nella directory principale del server Web e includerlo nelle pagine dove si utilizzano le costanti

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


Utilizzo di “adovbs.inc” (2)

Se non lo trovate non disperate…si può creare da zero:

```
Const adOpenForwardOnly = 0
Const adLockOptimistic = 3
Const adLockPessimistic = 2
Const adLockReadOnly = 1
Const adCmdTable = 2
```

Naturalmente si potrà utilizzare:

```
objRS.Open "miatabella", objConn, 0, 1, 2
```

Oppure

```
objRS.Open "miatabella",
   objConn,adOpenForwardOnly,
adLockReadOnly, adCmdTable
```

Apertura di un Recordset

```
Set objConn
   =Server.CreateObject("ADODB.Connection")
objConn.Open(SorgenteDatiODBC)
Set objRS =
   Server.CreateObject("ADODB.Recordset")
objRS.ActiveConnection = objConn
….
objConn.Open(QuerySQL)
```

Definiamo un oggetto objRS e definiamo i valori per le sue proprietà (ActiveConnection, LockType, CursorType, CursorLocation).

Esempio creazione del recordset

```
<%
...
'Definisce l'apertura del cursore
1. strQuery = "SELECT NomeProdotto,
   PrezzoUnitario FROM Prodotti"
2. strQuery = strQuery & "ORDER BY
   NomeProdotto"
' Esegue la query e genera il cursore
3. Set objRS = objConn.Execute(strQuery)
%>
```

- Nel punto 1. e 2. Viene definita quella che sarà la query da eseguire
- Nel punto 3. Viene eseguita la query e generato il cursore

Leggere record dal database (1)

Notazione per accedere alle colonne del recordset:
NomeRecordSet.Fields.Item("NomeColonna").Value
NomeRecordSet("NomeColonna")
NomeRecordSet(n)
Nei nostri esempi NomeRecordSet è " objRS "

Leggere record dal database (2)

Metodi Per accedere alle righe del recordset

- **MoveNext**: Consente di spostarsi al record successivo
- **MovePrevious**: Consente di spostarsi al record precedente
- **MoveFirst**: spostamento al primo record
- **MoveLast**: spostamento sull'ultimo record
- **BOF**: indica se è posizionato all'inizio del recordset (True)
- **EOF**: indica se si è giunti alla fine del recordset (True)
- **Fields.Count**: numero di colonne presenti nel cursore

Esempio di lettura dei record dal database

```
<HTML>
<BODY>
….Tutte le password memorizzate nel database:
   <BR>
<%
   While Not objRS.EOF
        Response.Write Cursore(”UserID”) & “:” &
   _
        objRS(”Password”) & “<BR>”
        objRS.MoveNext
   Wend
   ……
```


Aggiungere record al database (1)

Utilizzare il metodo **Execute**, passandogli una Query SQL

```
query= “INSERT INTO STUDENTE VALUES “ & _
“(82498,’Gianluca’, ‘Di Tomassi’, ‘6stosenso’)”
objRS.Execute(query)
```

Utilizzare il metodo **AddNew** per aggiungere una riga al recordset. In questo caso, i campi del record devono essere impostati con adUseServer e con adOpenDynamic o adOpenKeyset.

AddNew crea il nuovo record vuoto e lo definisce come record corrente. Dopo aver terminato di impostare i valori, si deve aggiungere il record alla tabella del db con Update

Aggiungere record al database (2)

Esempio:

```
objRS.AddNew
objRS("Nome") = "Gianluca"
…
objRS.Update
```

In alternativa si può utilizzare la seguente sintassi:

objRS.AddNew *campo*, *valore*

Campo e valore sono valori singoli o array con lo stesso numero di elementi.

Esempio:

```
objRS.AddNew "Nome", "Gianluca" objRS.AddNew
  Array("Nome",
  "Cognome"),Array("Gianluca","Di Tomassi")
```


Modificare un record del database(1)

Utilizzare il metodo **Execute**, passandogli una Query SQL

```
query= "UPDATE STUDENTE SET Password
  =`canone^-1" & _
" WHERE Matricola=82498"
objRS.Execute(query)
```

- Utilizzare il metodo **Update** per modificare una riga al recordset. In questo caso, i campi del record devono essere impostati con adUseServer. Ci si deve portare sul record che si desidera modificare e poi

```
objRS("Password") = "canone^-1"
objRS.Update
```

Modificare un record del database(2)

Utilizzando CancelUpdate si può operare sulle modifiche effettuate sui record esistenti

```
objRS("Nome") = "Gianluca"  objRS("Cognome") = "Di Tomassi"      objRS.CancelUpdate
```

In questo modo si annullano le due righe precedenti

Esempio:

```
objRS("password") = Request("Pass")
objRS("email") = Request("Email")
If objRS("password") = "" Then
     objRS.CancelUpdate
Else
     objRS.Update
End If
```


Elimina un record del database

Utilizzare il metodo **Execute**, passandogli una Query SQL

```
query= "DELETE FROM STUDENTE WHERE Matricola=82498"
objRS.Execute(query)
```

Utilizzare il metodo **Delete** per modificare una riga al recordset. In questo caso, i campi del record devono essere impostati conadUseServer.

Esempio:

```
objRS.MoveFirst
objRS.Delete
```

Cancella il primo record

Esempio di estrazione dei dati

```
<HTML>
<BODY>

<%
trovati = objRS.RecordCount
Response.Write "Il cursore è composto
   da"&trovati&"righe"
While Not objRS.EOF
   Response.Write objRS("NomeProdotto")
   Response.Write objRS.MoveNext
Wend ...
```


Chiusura del recordset e della connessione

```
   ...
   objRS.close
   objConn.close
   Set objRS = Nothing
   Set objConn = Nothing
%>
</BODY>
</HTML>
```

Chiusura anticipata della connessione (1)

- Al fine di risparmiare risorse del sistema è possibile chiudere la connessione con il server, ma avere la possibilità di continuare ad utilizzare l'oggetto recordset.
- Ciò è possibile solo se il cursore è di tipo adUseClient, in quanto il contenuto del cursore è trasferito nell'area di memoria del server Web ed è quindi separato dal database server.

Chiusura anticipata della connessione (2)

```
<% …
   Set objRS =
   Server.CreateObject("ADODB.Recordset")
   objRS.ActiveConnection = objConn
   objRS.CursorLocation = adUseClient
   …
   objRS.Open(query)
   objRS.ActiveConnection = Nothing
   objConn.Close
   Set objConn = Nothing
   While NOT objRS.EOF
        ….%>
```

Chiusura anticipata della connessione

Approfondimenti sulla connessione

Tre metodi per la connessione:

I METODO

La connessione avviene usando l'oggetto ADODB.RecordSet:

```
Set objRS =
   Server.CreateObject("ADODB.recordset")
objRS.Open strQuery, strProvider
```

Con il metodo *objRS.Open* apriamo materialmente la comunicazione con il database che si trova sul server, nel percorso individuato attraverso la stringa *strProvider* e definiamo anche l'interrogazione (in questo caso statica) che si vuole effettuare sul database e che viene letta attraverso la stringa *strQuery*.

Approfondimenti sulla connessione

```
strProvider = "DRIVER=Microsoft Access Driver
   (*.mdb);
DBQ=" & Server.MapPath("/")& "\asp\Login.mdb;"
strQuery = "SELECT user, password FROM LOGIN
   WHERE user='"&user&""'
```

Per definire il path dove si trova il database, utilizziamo il metodo dell'oggetto Server, *MapPath* in questo modo rendiamo il percorso "relativo" e non assoluto (Metodo 1a).

SVANTAGGIO: Problemi di carico sul server. Se il programma viene sviluppato solo per una macchina e non si prevede di dover in futuro, esportare le pagine realizzate su altri server, è preferibile usare la seguente modalità per definire il path

Approfondimenti sulla connessione

strProvider = "DRIVER=Microsoft Access Driver (*.mdb);
DBQ="&"c:\inetpub\wwwroot\asp\Login.mdb;"

In questo caso specifichiamo il path assoluto sulla macchina che stiamo utilizzando (Metodo 1b).

II METODO

Set objConn = Server.CreateObject("ADODB.Connection")
objConn.Open strProvider

In questo caso ci limitiamo a connettere il database con la nostra pagina

NOTA: non abbiamo ancora definito nessuno strumento per lavorare sui record. à definiamo il seguente oggetto:

Approfondimenti sulla connessione

Set objConn =server.createobject("ADODB.Connection")
objConn.Open strProvider
Set objRS = Server.CreateObject("ADODB.Command")
Set objRS.ActiveConnection = objConn

Ora objRS, ci permetterà di svolgere update, delete e insert, sul database (attraverso objRS.Open(query_definita)).

III METODO

In questo caso la connessione sfrutta le proprietà del driver odbc.

Set objConn =Server.CreateObject("ADODB.Connection")
objConn.Open "nome_Sorgente_dati_ODBC"

Interazione con l'utente

Form - tag <input>

- <input /> è il tag HTML che consente di realizzare una vasta collezione di elementi per l'inserimento dei dati in una form:
  - text (riga di testo), password, hidden (testo nascosto)
  - radio, checkbox, submit, reset (bottoni invio/cancellazione)
- Ad ogni elemento è possibile associare un nome (unico per distinguerlo dagli altri) ed un valore di default

Inserimento di testo - text

- text è la tipologia più comune di immissione dati un modulo.
- Usato per inviare una riga di testo al server

<input type="text" name="NomeElemento" />

- È possibile associare un valore predefinito mediante l'attributo value="valore"
- È possibile definire la dimensione del campo di testo (il numero di caratteri della casella di input) mediante l'attributo size="30".
- Se l'input è troppo lungo, il browser fa scorrere automaticamente il testo verso sinistra

Inserimento di testo - text

- È possibile definire la dimensione massima del campo di testo (il numero massimo di caratteri che può essere inserito nel campo) mediante l'attributo maxlength="50". È molto utile nell'interazione con i database

Inserimento di testo - password

- È esattamente uguale al tipo text; l'unica differenza è che il testo immesso dall'utente non viene visualizzato in chiaro, ma sostituito da caratteri * (asterischi)

Inserimento di testo - hidden

- hidden è utilizzato per inviare con un modulo una coppia nome/valore che non compare nella pagina web (non produce alcun effetto nella pagina web)
- Utilizza unicamente gli attributi type, name e value (non ha attributi di dimensione)

```
<input type="hidden" name="address"
  value="web@dom.com"/>
```

Inserimento di testo - texarea

- textarea è utilizzato per inviare un testo corposo (costituito da più righe e più colonne)
- È un tag e non un attributo:

<textarea name=”memo"> Testo ...</textarea>

- È possibile definire la dimensione della casella di inserimento del testo utilizzando gli attributi rows (numero di righe) e cols (numero di colonne)
- Tutto ciò che viene inserito tra <textarea> e </textarea> viene visualizzato come se fosse racchiuso tra i tag <pre> e </pre>

Caselle di controllo - checkbox

- Casella di controllo: piccolo quadratino selezionabile e deselezionabile
- Possono essere definite più caselle di controllo che fanno parte di una stessa lista. Si possono selezionare una o più caselle di una lista od anche nessuna

<input type=“checkbox” name=“hobbies” value=”calcio” />

<input type=“ checkbox” name=“hobbies” value=”musica” />

- In questo caso i valori verranno inviati come coppie hobbies=calcio o hobbies=musica

Caselle di controllo - checkbox

- Nel caso in cui non siano definiti gli attributi value i valori inviati al server saranno del tipo calcio=on/off

```
<input type="checkbox" name="calcio" />
```

- è possibile definire uno o più celle selezionate per default utilizzando l'attrbibuto cheched (checked="checked" in XHTML)

```
<input type=" checkbox" name="musica"
  checked />
```

- Non ci sono attributi per definire la dimensione della casella di controllo

Pulsanti di scelta - radio

- Questa tipologia di campo garantisce la possibilità di scegliere una sola tra più opzioni

```
<input type="radio" name="so" value="win95"
  />
<input type="radio" name="so" value="winNT"
  />
```

- In questo caso viene inviata la coppia nome/valore, ad esempio so=win95
- Con checked si può scegliere il pulsante selezionato di default
- Non ci sono attributi per definirne la dimensione

Liste di selezione - select

- select è utilizzato per creare liste di selezione a scomparsa e a scorrimento.
- È un tag e non un attributo e può contenere solo tag <option> al suo interno:

```
<select name=”so” size=“3” multiple>
   <option value=“95” selected>Windows
       95</option>
   <option value=“98” >Windows 98</option>
   <option value=“nt” >Windows NT</option>
</select>
```


Liste di selezione – select

- Con l’attributo size viene definita l’ampiezza della finestra di scorrimento (numero di elementi che possono essere visualizzati contemporaneamente). Per default size è pari ad 1
- L’attributo multiple abilità la possibilità di selezionare più elementi della lista (multiple=“multiple” in XHTML). Simile ai checkbox
- Se il valore di size è pari ad 1, viene creata una lista si selezione a scomparsa: per queste liste non è possibile selezionare più di un valore (simile ai radio)

Liste di selezione - select

- Il tag <option> specifica gli elementi della lista
- Con l'attributo selected (selected="selected" in XHTML) è possibile specificare l'elemento selezionato di default nella lista.
- I valori verranno inviati al sever come coppie: so=95 so=98

Invio

- Il campo submit serve per effettuare l'invio dei dati contenuti nella form

```
<input type="submit" value="premi qui" />
```

- Con l'attributo value si specifica il messaggio da visualizzare sul bottone di invio
- È possibile inserire un'immagine al posto del bottone

```
<input type="image" src="premi.gif" />
```

Cancella

- Il campo reset serve per cancellare tutti i dati inseriti nei campi della form, lasciando solo quelli definiti per default nei campi stessi

<input type="reset" value="Cancella" />

- Con l'attributo value si specifica il messaggio sul bottone di reset
- Non è possibile inserire un'immagine al posto del bottone (si può realizzare un collegamento alla pagina corrente)

Invio dei dati

Premendo il tasto submit:

1. il client apre una connessione con il server per passare i dati della form
2. Il server riconosce il tipo di chiamata per mezzo del metodo dichiarato dall'attributo method del tag form e riconosce il programma chiamato all'esecuzione per mezzo dall'attributo action
3. Il server HTTP passa al programma tutti i parametri e redirige l'output del programma verso lo stesso client

Utilizzo dell'oggetto Request per leggere "form"

Si utilizzano due collection dell'oggetto Request per leggere I campi restituiti da un campo di una form

1. **QueryString** si utilizza quando i campi della form vengono passati con metodo GET
2. **Form** si utilizza quando i campi della form vengono passati con metodo POST

La sintassi per il loro utilizzo è la seguente:

Request.QueryString(*NomeVariabile*)

Request.Form(*NomeVariabile*)

NomeVariabile è sempre la proprietà NAME del campo della form

- Per leggere l'intera stringa d'interrogazione occorre specificare il nome della collection:
- Esempio: **Request.QueryString**