Corso basi di dati
Comunicazione con l'utente

Gianluca Di Tomassi
Email: ditomass@dia.uniroma3.it

Università di Roma Tre

Creare moduli (form) in HTML

Elementi essenziali per lo scambio di dati:

- campi: elementi da riempire con testo, che sostituiscono un modulo cartaceo
- bottoni: almeno un tasto che consenta di spedire i campi riempiti al server
- CGI (Common Gateway Interface): programmi (Java, C++, perl, ASP, ..) lato server in grado di leggere i dati inviati, elaborarli e fornire al client una risposta adeguata.

Il tag <form>

Il tag fondamentale utilizzato per la definizione di form è <form> ... </form> e può essere posizionato in qualsiasi punto della pagina

Gli attributi principali del tag form:

- Action: specifica la risorsa che dovrà farsi carico dell'elaborazione dei dati inviati con il form, ovvero l'indirizzo URL al quale inviare i dati
- Method: seleziona il modo in cui vengono trasferiti i dati (get o post)

Esempio:

```
<form action="/cgi-bin/ordine" method="post">
…
</form>
```


GET e POST

Sottomettendo una form il client invia coppie del tipo nome=valore

La modalità get è indicata per il passaggio di pochi parametri e solo di tipo testuale perché i parametri vengono passati utilizzando l'URL

La modalità post è preferibile in presenza di una grande quantità di dati da inviare al server in quanto i parametri passano nel "corpo" della connessione

Un passo indietro: interazione client-server

Ogni volta che un client richiede al server una risorsa, apre una connessione ed invia sulla stessa un messaggio composto da due parti:

- method field: specifica il metodo HTTP da utilizzare, il protocollo, e l'indirizzo della risorsa. Costituisce la prima riga del messaggio
- request fields: contengono informazioni sulle capacità del client ad interpretare e gestire oggetti

Un esempio di metodo GET

Messaggio inviato dal client nel caso di utilizzo del metodo GET

```
GET /prodotti/index.html HTTP/1.0
Accept: text/plain
Accept: text/html
Accept: audio/*
……
User Agent: NCSA Mosaic for
CRLF (blank line)
```

La risposta del server

Il server a sua volta invia al client un messaggio composto da due parti:

- response header : specifica il risultato dell'operazione, la data, il tipo di server web, la versione mime, il tipo di contenuto trasmesso, la data di ultima modifica della risorsa richiesta, la dimensione dei dati trasmessi
- request data: contiene il file trasmesso dal server al client

Un esempio di risposta del server

```
HTTP/1.0  200  OK
Date: Friday, 28-Aug-2000 16:00:00 GMT
Server: NCSA/1.3
MIME-version: 1.0
Content-type: text/html
Last-modified: Friday, 21-Aug-2000 10:30:00 GMT
Content-lenght: 49
CRLF (blank line)
<html><body>
Questa è la risposta del server
</body></html>
```

La risposta del server

- Il risultato è un numero compreso tra 200 e 599:
  - 200: OK
  - 300: reindirizzamento
  - 400: errore (404 file not found)
  - 500: errore (server)
- MIME: Multipart Independent Mail Extension; pensato per inviare dati "misti" attraverso una mail
- Content-Type: indica il tipo di contenuto,se ad esempio si specifica application/ms-word, la risposta del server viene passata direttamente a MS Word

Un esempio di metodo GET in una form

```
GET /cgi-bin/ordine?cliente=DiTomassi&id=112 HTTP/1.0
Accept: text/plain……
User Agent: NCSA Mosaic for …
CRLF (blank line)
```

- In questo caso la lunghezza della prima riga è limitata: non è quindi possibile inviare una grande quantità di dati con il metodo get
- I parametri passano in "chiaro" (sono facilmente visibili all'utente)

Un esempio di metodo GET in una form

- In questo caso stiamo inviando informazioni al server attraverso una form con metodo GET:

  - dopo l'URL (ovvero il riferimento presente nel campo action) il browser inserisce automaticamente il ? seguito dalla lista dei parametri

  - ogni coppia di parametri è separata dal simbolo &

  - caratteri bianchi (spazi) sono rimpiazzati con il simbolo +

Un esempio di metodo GET in una form

- In questo caso stiamo inviando informazioni al server attraverso una form con metodo GET:

  - analogamente altri caratteri speciali vengono codificati con il simbolo % seguito dal codice numerico. Questa operazione è anche detta URLEncode. Ad esempio:

Name=Christoph Östrei

⇓

Name=Christoph+%D6Östrei

Un esempio di metodo POST in una form

```
POST  /cgi-bin/ordine  HTTP/1.0
Accept: text/plain……
User Agent: NCSA Mosaic for…
Content-type: application/x-www-form-urlencoded
Content-lenght:21
CRLF (blank line)
cliente=DiTomassi&id=112
```


Un esempio di metodo POST in una form

In questo caso stiamo inviando informazioni al server attraverso una form con metodo POST:

- nella prima riga compare solo l'URL del campo action
- l'elenco dei parametri viene passato nel corpo del messaggio:
  - la lunghezza del corpo non è limitata: è quindi possibile inviare una grande quantità di dati con il metodo post
  - i parametri non passano in "chiaro" (non sono visibili nell'URL)

Un esempio di metodo HEAD

HEAD /prodotti/index.html HTTP/1.0
User Agent: HEAD Test Agent

HTTP/1.0 200 OK
Date: Friday, 28-Aug-2000 16:00:00 GMT
Server: NCSA/1.3
MIME-version: 1.0
Content-type: text/html
Last-modified: Friday, 21-Aug-2000 10:30:00 GMT
Content-lenght: 0

Usato per verificare i file in cache

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
- Vedremo anche i tag textarea (righe di testo) e select (elementi di selezione singola/multipla)

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

Esempi complessivi sulle Form

Sul sito del corso guardare gli esempi:

http://diadema.dia.uniroma3.it/ASP/esempi/form/form_GET.html

http://diadema.dia.uniroma3.it/ASP/esempi/form/form_POST.html

Alla fine di queste slide guardare:

http://diadema.dia.uniroma3.it/ASP/esempi/form/form.asp

Riepilogando sui moduli (form)

Una form ha due compiti:

1. Raccogliere i dati dall'utente
2. inviarle i dati raccolti ad una pagina Web

Per passare le informazioni sono disponibili due metodi:

GET → I dati vengono passati nella stringa d'interrogazione

POST → I dati vengono passati nel body del pacchetto HTTP

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`**

ATTENZIONE:

Se i valori dei campi della form si passano con metodo GET e si utilizza la Request.Form oppure se si passano con metodo POST e si utilizza Request.QueryString allora viene restituita una stringa d'interrogazione di lunghezza 0

SOLUZIONE

Si può utilizzare la sintassi `Request("NomeVariabile")` senza preoccuparsi di quale delle due Collection si debba utilizzare.

Quando non si specifica la collection, l'oggetto Request le controlla entrambe.

COME OPERA L'INTERPRETE

L'oggetto Request effettua la ricerca prima sulla stringa d'interrogazione (metodo GET), se il nome della variabile non viene reperita viene effettuata la ricerca nelle intestazioni HTTP (metodo POST), se anche in questo caso il nome della variabile non viene reperito viene restituita una stringa vuota.

Esempio di lettura campi form passati con metodo GET

```
'Se il metodo utilizzato nella form è GET
 1:If Request.ServerVariables("QUERY_STRING") <> "" Then
 2: Response.write("<b>Query String: " & _
       Request.ServerVariables("QUERY_STRING") & "</b>")
 3: Dim strName
    'Lista dei valori della form nella collection QueryString
 4: For Each strName in Request.QueryString
 5:   If Request.QueryString(strName)= "" Then
 6:    Response.Write strName & “- <i>VALORE NON INSERITO _
                                 </i>"  & Vbcrlf
 7:   Else
 8:    Response.Write strName & “ - ” & _
          Request.QueryString(strName) & "<BR />"  & Vbcrlf
    End If
  Next
End If
```


SPIEGAZIONE:

Se il metodo con cui sono stati passati i campi della form è GET allora vuol dire che la QueryString è non nulla (riga 1) e allora si stampa l’intera QueryString (riga 2).

Attraverso un ciclo For-Each (riga 4) si scandisce l’intera QueryString ottenendo ad ogni ciclo una coppia nome/valore (il nome è `strName` e il valore è If `Request.QueryString(strName)`).

Si controlla se il valore per il campo corrente è nullo (riga 5) e si stampa la coppia (riga 6 oppure 8)

Esempio di lettura campi form passati con metodo POST

```
'Se il metodo utilizzato nella form è POST
 1:If Request.ServerVariables("CONTENT_LENGTH") <> 0 Then
 2: Response.write("<b>Content Lenght: " & _
    Request.ServerVariables("CONTENT_LENGTH") & "</b>")
 3: For i = 1 to Request.Form.Count
 4:    If Request.Form(i) = "" Then
 5:     Response.Write i & " - <i>VALORE NON INSERITO</i>"
 6:    Else
 7:     Response.Write i & " - " & Request.Form(i) & "<BR />"
       End If
    Next
   End If
```


SPIEGAZIONE:

Se il metodo con cui sono stati passati i campi della form è POST allora vuol dire che il ContentLength è non nullo (riga 1) e allora si stampa il ContentLength (riga 2).

Attraverso un ciclo For-Next (riga 3) si scandisce l'intera Form ottenendo ad ogni ciclo il numero del campo e il valore (il numero del campo è `i` e il valore è If `Request.Form(i)`).

Si controlla se il valore per il campo corrente è nullo (riga 4) e si stampa la coppia (riga 5 oppure 7)

Accortezze sul recupero valori dalle caselle di controllo

ATTENZIONE:

Si Ricorda che le caselle di controllo vengono create mediante il tag <input> con la proprietà TYPE impostata a checkbox

Si utilizzano per raggruppare le opzioni correlate quindi dovrebbero avere tutte valore identico per la proprietà NAME

Proprio per questo motivo non si riceve un singolo valore, ma si riceve un elenco delimitato da virgole delle proprietà VALUE delle caselle di controllo selezionate

Esempio:

```
<input type="checkbox" name="interessi" value="Val1"/>Calcio
<input type="checkbox" name="interessi" value="Val2"/>Tennis
<input type="checkbox" name="interessi" value="Val3"/>Karate
```

Per leggere i valori selezionati utilizziamo:

```
<%
  Dim interessiCheckbox
  interessiCheckbox = Request("interessi")
%>
```

Se si sono selezionate la prima e la terza allora il valore di **`interessiCheckbox`** sarà "Val1, Val3" (se nessuna delle 3 caselle fosse selezionata allora si avrebbe la stringa vuota)

Quindi nel caso di caselle di controllo **Request(*NomeCheckBox*)** restituisce un elenco delimitato da virgole di tutte le caselle di controllo selezionate, la cui proprietà NAME è uguale a *NomeCheckBox*

Se si vuole disporre separatamente di tutte le caselle di testo selezionate dall'utente è possibile creare un array

```
<%
   Dim interessiCheckbox
   interessiCheckbox = Request("interessi")
   Dim interessiArray
   interessiArray = split(interessiCheckbox , ",")
%>
```


La funzione **split()** contiene un elenco delimitato e lo suddivide in un array.

Viene letto l'elenco delle caselle di controllo selezionate, delimitato da virgole, nella variabile **interessiCheckbox**.

Il numero di elementi di **interessiArray** dipende dal numero di elementi presenti nell'elenco delimitato da virgole **interessiCheckbox**. Se nessuna casella di controllo fosse selezionata, allora **interessiCheckbox** sarebbe una stringa vuota e la funzione **split()** darebbe come risultato un array con zero elementi e quindi **interessiArray** avrebbe 0 elementi

Per visualizzare il contenuto dell'array

```
<%
 Dim i
 For i = Lbound(interessiArray) to Ubound(interessiArray)
       Response.Write interessiArray (i) & <br>
 Next
%>
```

Sul sito del corso guardare l'esempio:

http://diadema.dia.uniroma3.it/ASP/esempi/form/form_interessi.asp

Accedere alle intestazioni HTTP

Quando il client richiede una pagina Web dal server invia:

1. URL della pagina Web richiesta
2. Informazioni aggiuntive (es. browser utilizzato dal client, SO del client,...)

Ogni parte delle informazioni aggiuntive viene chiamata ***intestazione della richiesta*** (request header).

Una request header è una singola riga di testo che il browser invia al server Web quando richiede una pagina.

Un browser può inviare 0 o N intestazioni

Quando il server Web invia la pagina Web richiesta al client, invia anche un insieme di intestazioni, denominate ***intestazioni di risposta*** (response header)

In generale si fa riferimento alle intestazioni di richiesta e di risposta come ***intestazioni HTTP*** (HTTP header)

Quindi un'intestazione HTTP è una singola informazione, inviata dal client al server quando si richiede una pagina, oppure dal server al client quando si risponde alla richiesta di una pagina

NOTA: Le intestazioni HTTP non appaiono nella pagina Web visualizzata nel browser

Intestazioni HTTP personalizzate

Se si deve inviare un'intestazione di risposta personalizzata si utilizza il metodo AddHeader dell'oggetto Response. La sintassi è la seguente:

```
Response.AddHeader NomeIntestazione, ValoreIntestazione
```

Esiste un insieme standard di informazioni che la maggior parte dei browser invia al server Web.

La più utile è l'intestazione **User-Agent** che contiene le informazioni sul browser che è utilizzato e sul sistema operativo del client

Esempio utile:

Netscape Navigator e Internet Explorer gestiscono in maniera differente i fogli di stile (CSS). Si può risalire al browser del client per poi modificare le definizioni dei fogli di stile

Intestazioni HTTP standard

Tutte le intestazioni HTTP standard possono essere lette con **`Request.ServerVariables`**

Nome intestazione HTTP	Descrizione
HTTP_ACCEPT	Un elenco dei tipi MIME accettati dal client
HTTP_ACCEPT_LANGUAGE	contiene il linguaggio del documento
HTTP_CONNECTION	Il tipo di connessione stabilita tra il client e il server Web
HTTP_HOST	Il nome host del computer client
HTTP_USER_AGENT	Il tipo, la versione del browser e info sul SO del client
HTTP_REFERER	L'URL completo della pagina Web contenente il link usato per raggiungere la pagina ASP in esecuzione
HTTP_COOKIE	I cookie inviati dal browser

Leggere tutte le intestazioni HTTP

Si utilizza la Collection ServerVariables dell'oggetto Request.

Le collection vengono usate per memorizzare delle coppie nome/valore, per ottenere il valore è necessario conoscere il nome.

Per visualizzare tutte le intestazioni HTTP si utilizza:

```
<% Request.ServerVariables("ALL_RAW") %>
```

Così si visualizza l'elenco esatto delle intestazioni inviate dal browser al server Web.

Per visualizzare un elenco formattato di intestazioni si utilizza:

```
<% Request.ServerVariables("ALL_HTTP") %>
```

Leggere una specifica intestazione HTTP

Occorre utilizzare la seguente sintassi:

```
<% Request.ServerVariables("HTTP_NomeIntestazione") %>
```

NOTARE:

- Occorre usare il prefisso HTTP_ con il nome dell'intestazione
- Se il nome dell'intestazione contiene dei trattini (-), è possibile sostituirli con caratteri di sottolineatura (_)

Esempio:

Per visualizzare l'intestazione User-Agent si utilizza:

```
<% Request.ServerVariables("HTTP_USER-AGENT") %>
```

Oppure

```
<% Request.ServerVariables("HTTP_USER_AGENT") %>
```


Sul sito del corso guardare l'esempio:

http://diadema.dia.uniroma3.it/ASP/esempi/HTTP/HTTPHeaders.asp

L'intestazione User-Agent

- Varia a seconda del browser
- Esiste un componente standard ASP "Browser Capabilities"

L'intestazione User-Agent segue questo standard:

- Sia Internet Explorer che Netscape Navigator iniziano le loro intestazioni User-Agent con la parola "Mozilla"
- Netscape Navigator fa seguire alla parola "Mozilla" una barra con il numero della versione
- Internet Explorer ha un insieme di parametri in una stringa che segue Mozilla. Questa stringa inizia con una parentesi e un punto e virgola delimita ogni parametro. Il secondo parametro è il gruppo di caratteri MSIE, seguiti dal numero della versione.
- Anche l'intestazione User-Agent di Netscape Navigator contiene una stringa di parametri racchiusa tra parentesi (Il SO è il primo parametro)
- Internet Explorer invia il SO come terzo parametro della stringa di parametri

Esempi:

(1) Mozilla/4.0 (compatibile; MSIE 4.01; Windows 98)

(2) Mozilla/4.0 (compatibile; MSIE 5.01; Windows NT 5.0)

(3) Mozilla/4.5 (WinNT; I)

(1) Intestazione User-Agent di un client che utilizza Internet Explorer 4.01 su SO Windows 98

(2) Intestazione User-Agent di un client che utilizza Internet Explorer 5.01 su SO Windows NT 5.0

(3) Intestazione User-Agent di un client che utilizza Netscape Navigator 4.5 su SO Windows NT

Sul sito del corso guardare l'esempio:

http://diadema.dia.uniroma3.it/ASP/esempi/HTTP/UserAgent.asp

Il componente Browser Capabilities

Il componente standard ASP “Browser Capabilities” risponde alle domande:

Quale browser sta usando il client ?
Quale è la versione del browser ?
Il browser supporta i frame ?
Il browser supporta le tabelle ?
Il browser supporta i suoni in sottofondo ?
Il browser supporta VBScript ?
Il browserr supporta JavaScript ?

Sul sito del corso guardare l’esempio:

http://diadema.dia.uniroma3.it/ASP/esempi/HTTP/UsoBrowserCapabilities.asp

Esempio *UsoBrowserCapabilities.asp*

```
<%@ Language=VBScript %>
<% Option Explicit %>
<HTML>
<BODY>
<%
    Dim objBC
    Set objBC = Server.CreateObject("MSWC.BrowserType")
%>
    Browser Name: <%= objBC.Browser %><P>
    Browser Version: <%= objBC.Version%><P>
<%  If (objBC.Frames) then %>
      Your browser supports frames<P>
<%  else %>
      Your browser does not support frames<P>
<%  end if %>
```

```
<%  If (objBC.Tables) then %>
      Your browser supports tables<P>
<%  else %>
      Your browser does not support tables<P>
<%  end if %>
<%  If (objBC.BackgroundSounds)then %>
      Your browser supports background sounds<P>
<%  else %>
      Your browser does not support background sounds<P>
<%  end if %>
<%  If (objBC.VBScript) then %>
      Your browser supports VBScript<P>
<%  else %>
      Your browser does not support VBScript<P>
<%  end if %>
```



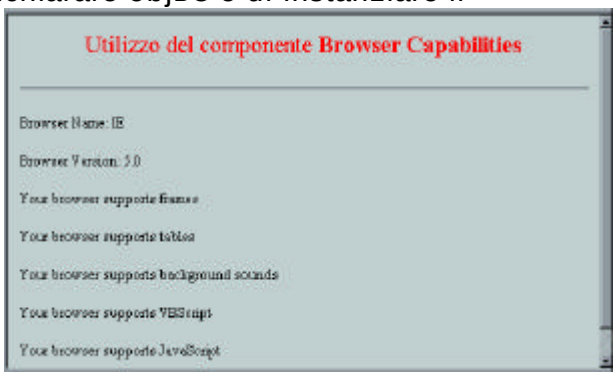
```
<%  If (objBC.Javascript) then %>
      Your browser supports JavaScript<P>
<%  else %>
     Your browser does not support JavaScript<P>
<%
    End if
    Set objBC = nothing
%>
  </BODY>
</HTML>
```

Le righe

```
<%   Dim objBC
     Set objBC = Server.CreateObject("MSWC.BrowserType")
%>
```

consentono di dichiarare objBC e di instanziare il componente.

Il risultato sarà:



Non è tutto così facile...

Quando viene distribuito un nuovo browser, questo viene identificato come "sconosciuto" dal componente Browser Capabilities

SOLUZIONE:

Per questo viene utilizzato **Browscap.ini** che è un file che può essere aggiornato con le nuove versioni del browser e con nuove capacità. Il file permette di controllare proprietà quali il numero della versione, il supporto dei cookie e il supporto ActiveX.

COSA ACCADE:

Sul server Web è presente il file Browscap.dll che contiene il componente ed effettua il confronto fra i dati nell'intestazione del browser con i dati in Browscap.ini

Per scaricare le versioni aggiornate del file **Browscap.ini**

http://207.176.17.138/browscap.zip

Accedere alle variabili d'ambiente

Le intestazioni HTTP sono **utili** per ottenere informazioni sul visitatore attuale ma **non forniscono** informazioni sul server Web e sulla pagina Web richiesta dal client

RICORDA: Le intestazioni HTTP forniscono *SOLO* informazioni sul client

Per ottenere informazioni sul server Web si utilizzano *le variabili d'ambiente* (*environment variable*) del server Web stesso

Le variabili d'ambiente sono delle informazioni che il server Web rende disponibili a qualsiasi programma che le richiede.

Forniscono informazioni quali il nome del server Web, l'URL della pagina ASP che viene attualmente elaborata.

Come accedere alle variabili d'ambiente

In maniera analoga a come si accede alle intestazioni HTTP si utilizza la sintassi:

```
Request.ServerVariables(nomeVariabileAmbiente)
```

Essendo `Request.ServerVariables` una collection, si possono visualizzare i contenuti con un ciclo `For Each…Next`

NOTA

Poiché la collection `Request.ServerVariables` contiene sia le intestazioni HTTP che le variabili d'ambiente saranno visualizzate entrambe quando si elenca il suo contenuto

Sul sito del corso guardare l'esempio:

http://diadema.dia.uniroma3.it/ASP/esempi/HTTP/ListAllServerVariables.asp

Utili variabili d'ambiente

Variabile d'ambiente	Descrizione
URL	URL della pagina ASP a partire dalla fine di http://www.*ProprioServerWeb*.com/ fino alla stringa di interrogazione
PATH_INFO	Come URL
PATH_TRASLATED	Il percorso fisico completo della pagina ASP attualmente in esecuzione
APPL_PHYSICAL_PATH	Indirizzo fisico delle directory principale del Web
QUERY_STRING	Equivale a Request.QueryString
SERVER_NAME	Il nome del computer del server Web
SERVER_SOFTWARE	Il nome del software del server Web (per esempio Microsoft-IIS/5.0)

ATTENZIONE:

Quando si richiede una singola variabile d'ambiente o intestazione HTTP dalla collection `Request.ServerVariables`, tutte le variabili d'ambiente e le intestazioni HTTP sono caricate nella collection.

CONSEGUENZA

Il caricamento di TUTTE le variabili d'ambiente e delle intestazioni HTTP può provocare una diminuzione delle prestazioni sul proprio sito Web.

CONCLUSIONE

La diminuzione non è elevata pertanto va bene il suo utilizzo ma solo quando è veramente necessario

NOTA: Molte variabili d'ambiente non contengono un valore.

Esempio:

Le variabili cha hanno prefisso CERT contengono tutte delle stringhe vuote come valori perché sono usate solo quando il client e il server usano certificati

Quando un browser e un server Web comunicano attraverso un canale sicuro vengono utilizzati i certificati per garantire l'identità di entrambi.

I certificati vengono utilizzati quando si usa una SSL (Secure Socket Layer) sul server Web

Sul sito del corso guardare l'esempio:

http://diadema.dia.uniroma3.it/ASP/esempi/HTTP/MyASPPage.asp

NOTE SULL'ESEMPIO

Se si desidera ottenere l'URL della pagina ASP attualmente in esecuzione si devono utilizzare le due variabili d'ambiente *URL* e *SERVER_NAME*.

URL percorso virtuale completo e nome del file

SERVER_NAME contiene l'hostname attuale del server Web

In definitiva

La collection `Request.ServerVariables` serve per:

1. Leggere le intestazioni HTTP inviate dal browser al server Web
2. Leggere le variabili d'ambiente del server Web

- *Le intestazioni HTTP* forniscono informazioni su quale browser e quale sistema operativo hanno gli utenti

- *Le variabili d'ambiente* forniscono informazioni sul server Web