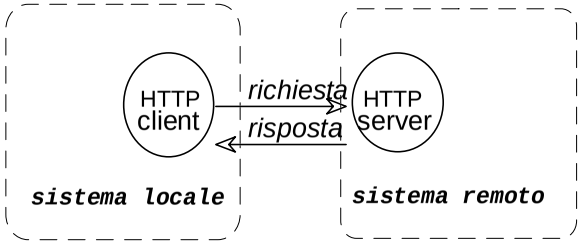
CdL in Ingegneria Elettronica e Informatica
Modulo di Fondamenti di Informatica D

La Programmazione CGI

Prof. Franco Zambonelli
Febbraio 2001

Programmazione client/server in WWW

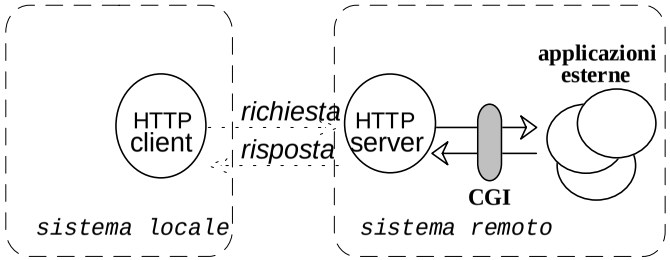


Possibilità di avere ***risposta*** con informazioni dinamiche

Che tipo di elaborazione delle informazioni e
dove viene eseguita

richiesta	risposta	tipo di elaborazione
Documento HTML	Statica (la pagina è un file, non modificabile)	semplice trasferimento file dal server
CGI	dinamica	qualunque elaborazione sul nodo server
Java applet	statica	codice dal server non modificabile, esecuzione sul client
Java applicazione	dinamica	server elabora dinamicamente il codice (in base alla richiesta), esecuzione dinamica sul client

Common Gateway Interface (CGI)



CGI è uno **standard** per interfacciare un server WWW con applicazioni esterne (residenti sulla macchina server)

CGI fornisce all'utente la capacità di eseguire una applicazione sulla macchina server remota

richiesta	***risposta***	***tipo di elaborazione***
CGI	dinamic a	Qualunque elaborazione, **sul nodo server**

La risposta ottenuta dal server è "dinamica", in quanto risultante dalla esecuzione di un programma sul server.

Programmazione CGI

Una applicazione CGI permette agli utenti di eseguire una applicazione sul nodo dove risiede il server www.
Applicazioni CGI possono essere scritte in: C/C++, Fortran, PERL, TCL, Any Unix shell, Visual Basic ….etc etc

Normale attivazione di una CGI:

- Si invia al server un messaggio (ciò avviene tipicamente riempiendo moduli, **FORM HTML**, i cui dati serviranno come input al programma)
- Il messaggio scatena l'esecuzione del programma CGI
- Il programma CGI genera come output una pagina HTML in cui inserisce i risultati della sua esecuzione

Interfaccia tra server WWW e applicazione CGI

- **Variabili di ambiente**
- **Linea di comando**
- **Standard input**: il server manda come standard input alla applicazione CGI i dati ricevuti dal client (browser). Il numero di byte totali è nella varabile di ambiente CONTENT_LENGTH.
- **Coppie Nome-Valore:** L'input della applicazione è costituito da insiemi di stringhe nome=valore. In pratica, l'input è formato da una serie di variabili con un nome e un valore. Le differenti variabili sono separate da ***&***
- **Standard output**: l'applicazione CGI manda il risultato dell-elaborazione sullo standard output (in formato HTML), verso il server, il quale prende i dati e li manda al client.

Client HTTP → server HTTP → CGI

Tipicamente, uso di **form**

```
<TITLE>Esempio di Form </TITLE>
<H1>Esempio di Form </H1>

<FORM METHOD="POST" ACTION="http://www-
lia.deis.unibo.it/cgi-bin/post-query">

Inserisci del testo: <INPUT NAME="entry">

e premi per invio: <INPUT TYPE="submit"
                    VALUE="Invio">
</FORM>
```

Visualizzazione **form**

Client HTTP → server HTTP → CGI

Attributi del **form tag**

```
<TITLE>Esempio di Form </TITLE>
<H1>Esempio di Form </H1>

<FORM METHOD="POST" ACTION="http://www-
lia.deis.unibo.it/cgi-bin/post-query">

Inserisci del testo: <INPUT NAME="entry">

e premi per invio: <INPUT TYPE="submit"
                    VALUE="Invio">
</FORM>
```

Dove:

ACTION URL di chi processa la query

METHOD metodo usato per sottomettere il form:

- **POST** il form con i dati è spedito come data body (metodo consigliato), cioè come dati che fanno parte del testo del messaggio HTTP
- **GET** il form con i dati è spedito attaccato all'URL, cioè come parte integrante dell'URL del messaggio (`action?name=value&name=value`)

caso GET

```
http://www-lia.deis.unibo.it
                    /cgi-bin/get-query?entry=testo
```

caso POST

```
http://www-lia.deis.unibo.it
                    /cgi-bin/post-query
```

e come data body:

```
                    entry=testo
```

Applicazione CGI → server HTTP

Applicazione CGI usa **standard output** per mandare al server i dati. I dati sono identificati da un header.

Tipi di dati forniti:

- full document con il corrispondente MIME type (text/html, text/plain per testo ASCII, etc.)

  Esempio: per spedire una pagina HTML

```
Content-type: text/html

<HTML><HEAD>
<TITLE>output di HTML da script CGI</TITLE>
</HEAD><BODY>
<H1>titolo</H1>
semplice <STRONG>prova</STRONG>
</BODY></HTML>
```

- reference a un altro documento

  Esempio: riferisco un documento gopher

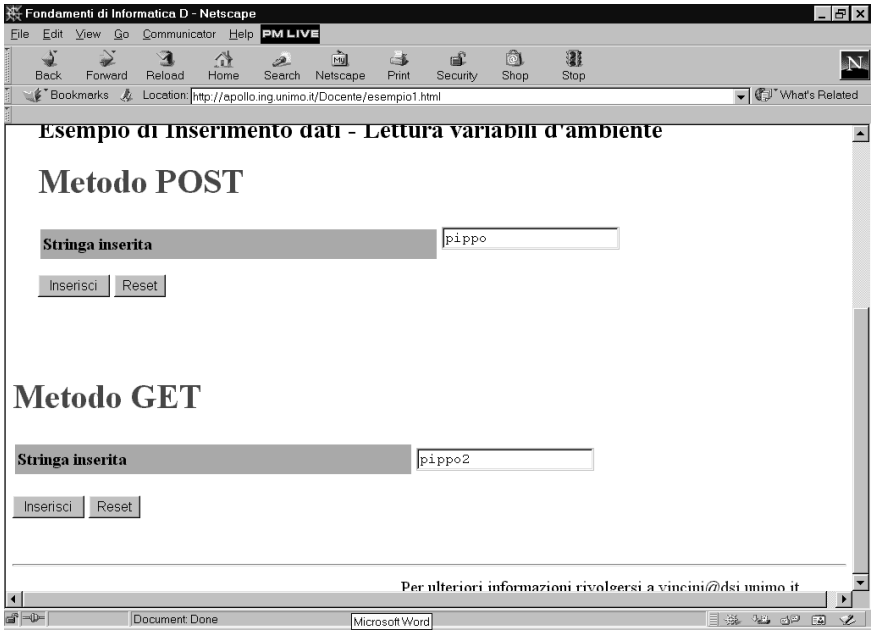
```
Content-type: text/html
Location: gopher://httprules.foobar.org/0

<HTML><HEAD>
<TITLE > Sorry ... it moved </TITLE>
</HEAD><BODY>
<H1>go to gopher </H1>
Now available at
<A HREF="gopher://httprules.foobar.org/0">
         new location</A> of gopher server.
</BODY></HTML>
```


Applicazione CGI

Esempio: generazione della pagina di risposta (caso full document)

```
#include <stdio.h>
..............


main(int argc, char *argv[]) {
  int cl;

generazione di un full document in risposta
  printf("Content-type: text/html");

  cl = atoi(getenv("CONTENT_LENGTH"));

  for(x=0;cl && (!feof(stdin));x++) {
    ...
    elaborazione dell'input (stdin)
    ...
     /* L'input ad esempio sara':
           "entry=ciao&Submit=Invio"
  }

  printf("<H1>Query Results</H1>");
  printf("You submitted ...");

  for(x=0; x <= m; x++)
    printf("...........", ... , ....);
}
```

ESEMPIO CON METODI POST E GET



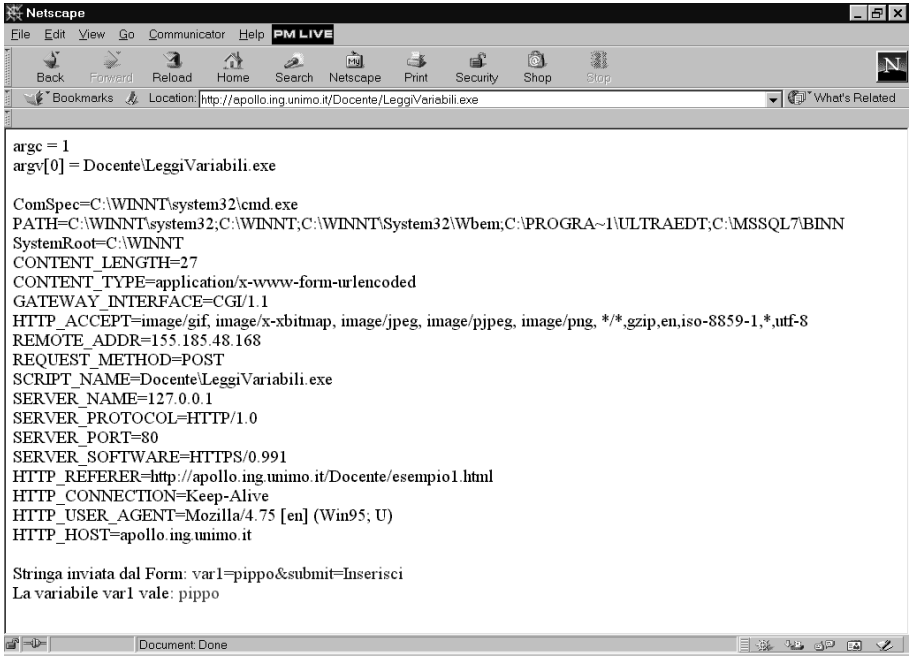
SORGENTE HTML

```
<H1>Fondamenti di Informatica D</H1>
<I><P>Prof. <A
HREF="http://www.dbgroup.unimo.it/Vincini.html"
>Maurizio Vincini</I></A> </P>
<HR>
<br>
<FONT SIZE="5"><B>Esempio di Inserimento dati -
Lettura variabili d'ambiente</B></FONT>
<br>
<H1>Metodo POST</H1>
<FORM ACTION="LeggiVariabili.exe" METHOD=post>
<TABLE border="0" cellpadding="3"
cellspacing="3" WIDTH="100%">
<tr><td colspan="5" align="right">
</td></tr>
<tr>
<P><td BGCOLOR="#76b6f2" width="200"><b>Stringa
inserita</b></td>
<td><INPUT NAME="var1"></td></P>
</tr>
</TABLE>
<BR>
<INPUT TYPE="submit" VALUE="Inserisci"
id=submit name=submit>
<INPUT TYPE="reset" VALUE="Reset"></P>
</FORM>

<br>
```

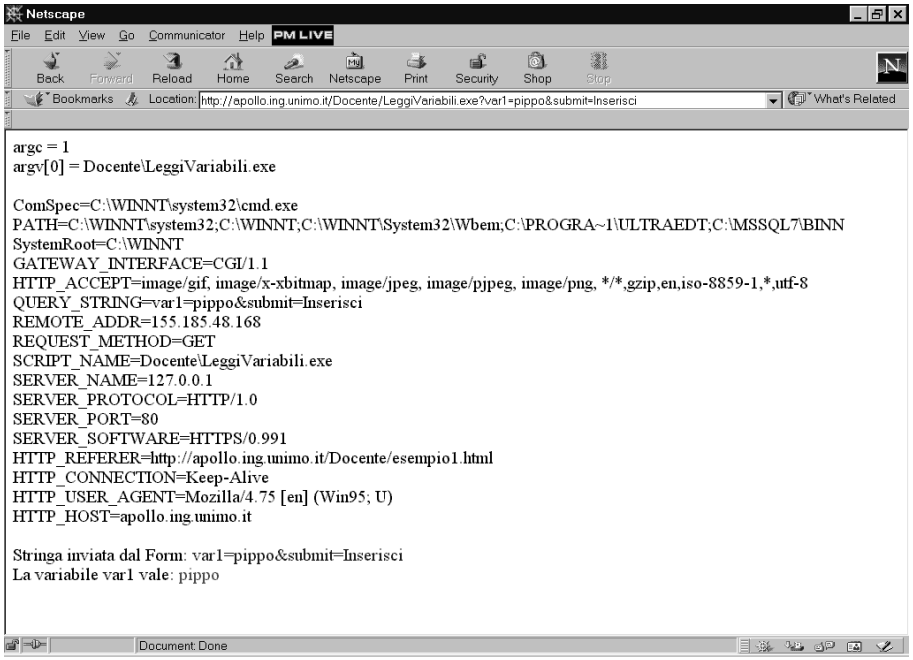
```
<H1>Metodo GET</H1>


<FORM ACTION="LeggiVariabili.exe" METHOD=get>

<TABLE border="0" cellpadding="3"
cellspacing="3" WIDTH="100%">

<tr><td colspan="5" align="right">

</td></tr>

<tr>

<P><td BGCOLOR="#76b6f2" width="200"><b>Stringa
inserita</b></td>

<td><INPUT NAME="var1"></td></P>

</tr>

</TABLE>

<BR>

<INPUT TYPE="submit" VALUE="Inserisci"
id=submit name=submit>

<INPUT TYPE="reset" VALUE="Reset"></P>

</FORM>
```

Supponiamo di inserire “pippo” e sottomettere

CON POST



CON GET



CODICE C DELL'ESEMPIO

```
/* QUI METTEREMO TUTTO L'INPUT */
char InputBuffer[4096];

/*
 *  Programma Principale
 */
int main(int argc, char *argv[]) {
int ContentLength;
int i;
char *p;
char *pRequestMethod;
char *c;

    /* Intestazione della pagina di output */

   printf("Content-type: text/html\n");

    /* Fine dell'header HTML*/
    printf("\n");


    /* Scrive le variabili di input */
    printf("argc = %d\n<BR>\n",argc);
    for (i=0;i<argc;i++) {
     printf("argv[%d] = %s\n<BR>\n",i,argv[i]);
    }
    printf("\n<BR>\n");

    /* Visualizza le variabili d'ambiente */

    DisplayEnvVars();

    printf("\n");
```

VISUALIZZAZIONE VAR. AMBIENTE

```
void DisplayEnvVars(void) {
int i;

    i = 0;
    while (_environ[i]) {
        printf("%s\n<BR>\n",_environ[i]);
/* VISUALIZZA COPPIA NOME=VALORE*/
        i++;
    }
    printf("\n<BR>\n");
}
```

ALTERNATIVA (se si conosce il nome della variabile di interesse)

```
char *valore_var;

  valore_var = getenv("HTTP_HOST");

/* oppure, se rappresenta un numero */
  cl = atoi(getenv("CONTENT_LENGTH"));
```


CONTINUA IL MAIN….

```
   /* Acquisizione metodo di richiesta
     (POST o GET) */
    pRequestMethod = getenv("REQUEST_METHOD");
    if (pRequestMethod==NULL) {
        return 0;
    }

    if (_stricmp(pRequestMethod,"POST")==0) {

/* ADESSO ANDIAMO A RECUPERARE DIM. INPUT */

p = getenv("CONTENT_LENGTH");


/* Trasformiamo la stringa in intero */
        if (p!=NULL) {
            ContentLength = atoi(p);
        } else {
            ContentLength = 0;
        }


/* controlliamo se InputBuffer e'grande
abbastanza */
if (ContentLength>sizeof(InputBuffer)-1) {
     ContentLength = sizeof(InputBuffer)-1;
        }
```

STAMPIAMO TUTTO L'INPUT

```
fgets(InputBuffer,ContentLength+1,stdin);
i = strlen(InputBuffer);
printf("Stringa inviata dal Form:
<FONT COLOR=""blue"">
            %s</FONT>\n<BR>\n",InputBuffer);
/* stampiamo tutto l'input come unica stringa
*/


if(strcmp(p,
 "application/x-www-form-urlencoded")==0)
{
        strcpy(p, InputBuffer);
        /* copiamo input per non modificare */
        /* la copia originale */

        /* Trova valore variabile var1 */
        c = GetField("var1", p);

        if (c != NULL) {
           printf("La variabile var1 vale:
            <FONT COLOR=""red""> %s </FONT>", c);
        } else {
           printf("La variabile var1
                           &egrave; NULL");
        }
       } else
        {
           /* Visualizza i dati */
           printf("Input = %s\n",InputBuffer);
       }
```


FUNZIONI PER RECUPERARE LE COPPIE NOME VALORE DALL'INPUT (1)

```
char *ValueField(char *Item) {
char *p;

    p = strchr(Item,'=');
/* torna la sottostringa che parte da '='  */

    if (p==NULL) return NULL;
    *p='\0';
/*  ci  mette  il  terminatore,  per  spezzare  la
coppia nome&valore in due stringhe separate  */

/* incrementa, e punta alla stringa valore */
     p++;

/* ritorna la stringa che rappresenta il valore
della variabile */
   return p;
}
```

FUNZIONI PER RECUPERARE LE COPPIE NOME VALORE DALL'INPUT (2)

```
/*  Ritorna il valore della variabile Item
contenuta nella stringa Field  */
char *GetField(char *Item, char *Field) {
char *p;
char *s;

    p = strtok(Field,"&");
/* la funzione strtok spezza una stringa in
diverse stringhe, assumendo come punto di
separazione il secondo parametro*/ /* se il
primo parametro e' NULL, si intende che usiamo
a spezzattare la stessa stringa di prima –
ritorna la stringa separata dal resto */
/* nel nostro caso, una stringa nome=valore */
/*  quando  non  e'  piu'possibile continuare  a
spezzettare, ritorna NULL */

  while (p!=NULL) {
        s = ValueField(p);
/*  andiamo  a  estrarre  il  valore  dalla  stringa
nome valore */

printf("Valore di variabile
            (%s) = %s <BR>\n",p, s);

      if (strcmp(p, Item) == 0) return s;
        p = strtok(NULL,"&");
/*  se  non  ha  trovato  la  variabile  cercata,
continua a spezzattare la stringa di input… */
    }

   return NULL;
}
```


RECUPERO VALORI VARIABILI: ATTENZIONE!!

Il Web server tratta in modo particolare i caratteri speciali inviati dalle CGI.

Caratteri speciali quali: 'à' (lettere accentate), simboli, etc…

Esempio: la 'à' diventa '%E5' (cioè il numero esadecimale che rappresenta il codice ASCII della a accentata).

Esempio: lo spazio ' ' diventa '+'.

Nella CGI, bisogna effettuare la conversione inversa!

```
/*
 *  Ritorna il valore della variabile
 */
char *ValueField(char *Item) {
char *p;

    p = strchr(Item,'=');
    if (p==NULL) return NULL;
    *p++='\0';
/* la funzione URL decode ri-trasforma la
codifica dei caratteri speciali, p.e.,
trasforma i '+' in spazi.

    urlDecode(Item);
    urlDecode(p);
   return p;
}
```

FUNZIONE URLDECODE

```
/*
 *  Decodifica la stringa in input che utilizza il %
 */
void urlDecode(char *p) {
char *pD;

    pD = p;
    while (*p) {
        if (*p=='%') {
            /* i 2 char successivi sono la
rappresentazione hex del carattere in esame */
            p++;
          // isxdigit ritorna un valore != 0 nel caso
in cui il char sia di tipo esadecimale
            if (isxdigit(p[0]) && isxdigit(p[1])) {
                *pD++ = (char) TwoHex2Int(p);
                p += 2;
            }
        } else {
          if (*p=='+') {   /* SPAZIO DIVENTATO '+' */
             *pD++ =  ' ';
             p++;
          }  else
             *pD++ = *p++;
        }
    }
    *pD = '\0';
}
```


FUNZIONE TwoHex2Int

```
/*
 *  The string starts with two hex characters.  Return
an integer formed from them.
 */
static int TwoHex2Int(char *pC) {
int Hi;
int Lo;
int Result;

    Hi = pC[0];
    if ('0'<=Hi && Hi<='9') {
        Hi -= '0';
    } else
    if ('a'<=Hi && Hi<='f') {
        Hi -= ('a'-10);
    } else
    if ('A'<=Hi && Hi<='F') {
        Hi -= ('A'-10);
    }
    Lo = pC[1];
    if ('0'<=Lo && Lo<='9') {
        Lo -= '0';
    } else
    if ('a'<=Lo && Lo<='f') {
        Lo -= ('a'-10);
    } else
    if ('A'<=Lo && Lo<='F') {
        Lo -= ('A'-10);
    }
    Result = Lo + 16*Hi;
    return Result;
}
```

BASIC FORM TAGS

I forms sono i moduli per richiedere informazioni all'interno delle pagine html.

Tag/attribute	Use
<FORM>	Indica un form all'interno di un documento HTML. All'interno di <FORM></FORM> si devono mettere tutti i campi che servono per generale l'input e sottmetterlo al server
<INPUT TYPE="SUBMIT" VALUE="…">	Fornisce un pulsante submit per il form. La pressione di questo pulsante scatena l'invio del form al server. L'attributo value produce il testo sul pulsante.
<INPUT TYPE="IMAGE" NAME= "POINT" SRC= "…" BORDER=0>	Fornisce un pulsante **submit** grafico. L'attributo SRC indica l'immagine che è originata dal file indicato, e l'attributo bordo = compone il bordo dell'immagine.
<INPUT TYPE="RESET" VALUE= "…">	Fornisce un pulsante reset per il form, che annulla tutti i dati inseriri. L'attributo VALUE produce un testo sul pulsante.

Input Field Tag and Attribute

Tag/attribute	Use
<INPUT>	Fissa un'area in un form per l'input di dati dall'utente
TYPE= "…"	Fissa il tipo di campo di input. Possibili valori sono TEXT, PASSWORD, CHECKBOX, RADIO, FILE, HIDDEN, IMAGE, SUBMIT, e RESET.
NAME= "…"	Nome della variabile
VALUE= "…"	Fornisce un contenuto associato con NAME= "…" . Bisogna usare questo attributo con i pulsanti RADIO e CHECKBOX perché non accettano altri input. Si può anche usare con text fields per avere un input iniziale.
SIZE= "n"	Fissa la dimensione visibile per un field. E' possibile usare questo attributo **solamente con i text input**.
MAXLENGTH= "n"	Fissa il più lungo set di caratteri che possono essere sottomessi. Usare questo attributo con text fields.
SELECTED	Indica la selezione di default che deve essere presentata quando il form è inizialmente caricato o resettato.
ACCEPT= "…"	Specifica i tipi accettabili di MIME per file caricati. Wildcards sono accettabili come in image/*.

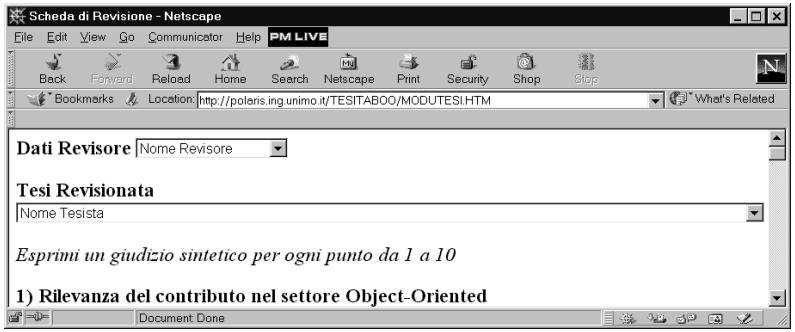
Text Area Tags and Attributes

Tag/attributes	Use
<TEXTAREA>	Fissa un'area in un form per un inserimento dati testuale dell'utente. Il contenuto iniziale del textarea va inserito tra l'opening e il closing tag.
NAME= "…"	Stabilisce una no me per un input field. L'attributo NAME è usato nel caso di CGI
ROWS= "…"	Fissa il numero di linee per l'area
COLS= "…"	Fissa il numero di colonne per l'area.

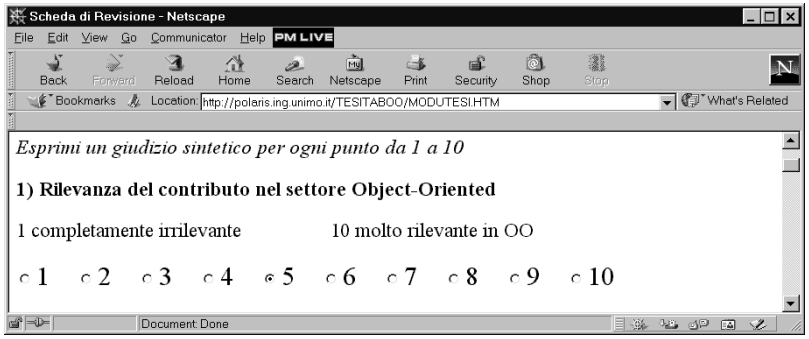
Select Fiedls Tags and Attributes

Tag/attributes	Use
<SELECT>	Fissa un'area in un form per un field di selezione che può intendersi come una lista a cascata.
NAME= "…"	Stabilisce un nome
SIZE= "n"	Fissa la dimensione di larghezza per i campi. Il default crea una lista a cascata. Si può cambiare il default (2 o maggiore) se si vogliono più opzioni ben visibili.
MULTIPLE	Fa si' che un campo selezionato accetti più di una selezione. Si utilizza questo attributo insieme a SIZE= per fissare un numero grande quanto il massimo numero di possibili selezioni.
<OPTION>	Evidenzia i valori inclusi nel campo selezionato. Si avrà un <OPTION> per ogni item che si inserirà . Il closing tag è opzionale.
VALUE= "…"	Fornisce un valore associato a Name
SELECTED	Fa specificare una selezione per default che apparirà quando il form è inizialmente caricato o resettato

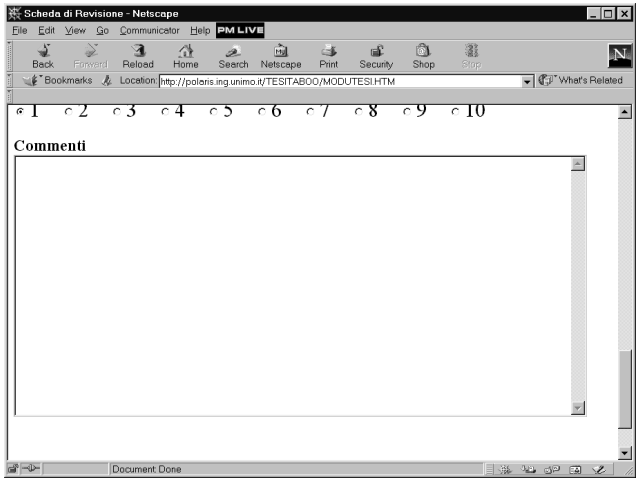
ESEMPI



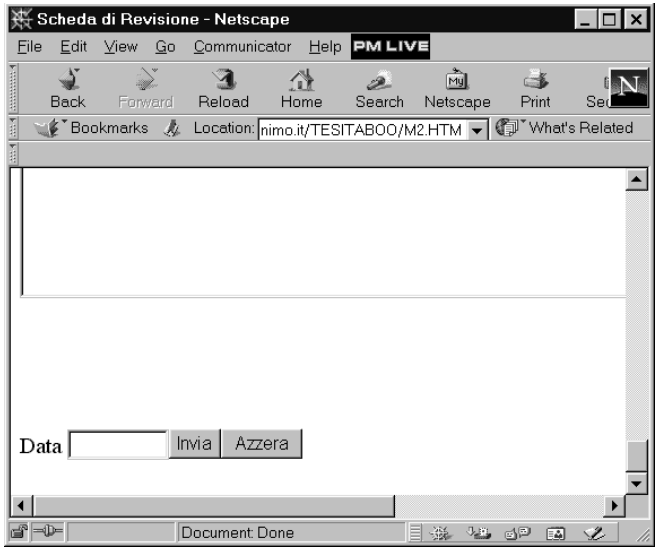

```
<FORM method="POST" action="/cgi-bin/giudizio">

<B><FONT SIZE=+1>Dati Revisore</FONT></B>

<INPUT TYPE=”HIDDEN” NAME=”ANNO” VALUE=”2001”>


<SELECT NAME="Revisore">

<OPTION SELECTED VALUE="Nessun Nome">Nome Revisore

<OPTION  VALUE=" Sonia Bergamaschi"> Sonia Bergamaschi

<OPTION  VALUE=" Letizia Leonardi"> Letizia Leonardi

<OPTION  VALUE=" Franco Zambonelli"> Franco Zambonelli

</SELECT>

....
```


```
<P><I><FONT SIZE=+1> Esprimi un giudizio sintetico per ogni punto da 1 a 10 </FONT></I>

<P><B><FONT SIZE=+1>1) Rilevanza del contributo nel settore Object-Oriented</FONT></B>

<P><FONT SIZE=+1>1 completamente irrilevante
10 molto rilevante in OO</FONT>

<DL>
<DT>
<FONT SIZE=+2>

<INPUT type="RADIO" name="Rilevanza" value="1" checked>1

<INPUT type="RADIO" name="Rilevanza" value="2">2

<INPUT type="RADIO" name="Rilevanza" value="3">3

…

<INPUT type="RADIO" name="Rilevanza" value="10">10
</FONT>

</DT>
</DL>
```




```
<P><P><B><FONT SIZE=+1>Commenti </FONT></B>
<BR>
<textarea name="Commenti Generali" rows="20" cols="80"
maxlength="65000" wrap="HARD">
</textarea><P>
<FONT SIZE=+1> 
</FONT>
```


```
<P>
<p>Data
<input type="TEXT" size="8" maxlength="8" name="Data">

<input type="SUBMIT"  value="Invia">

<input type="RESET" value="Azzera"></p>
</FORM>
</BODY>
</HTML>
```