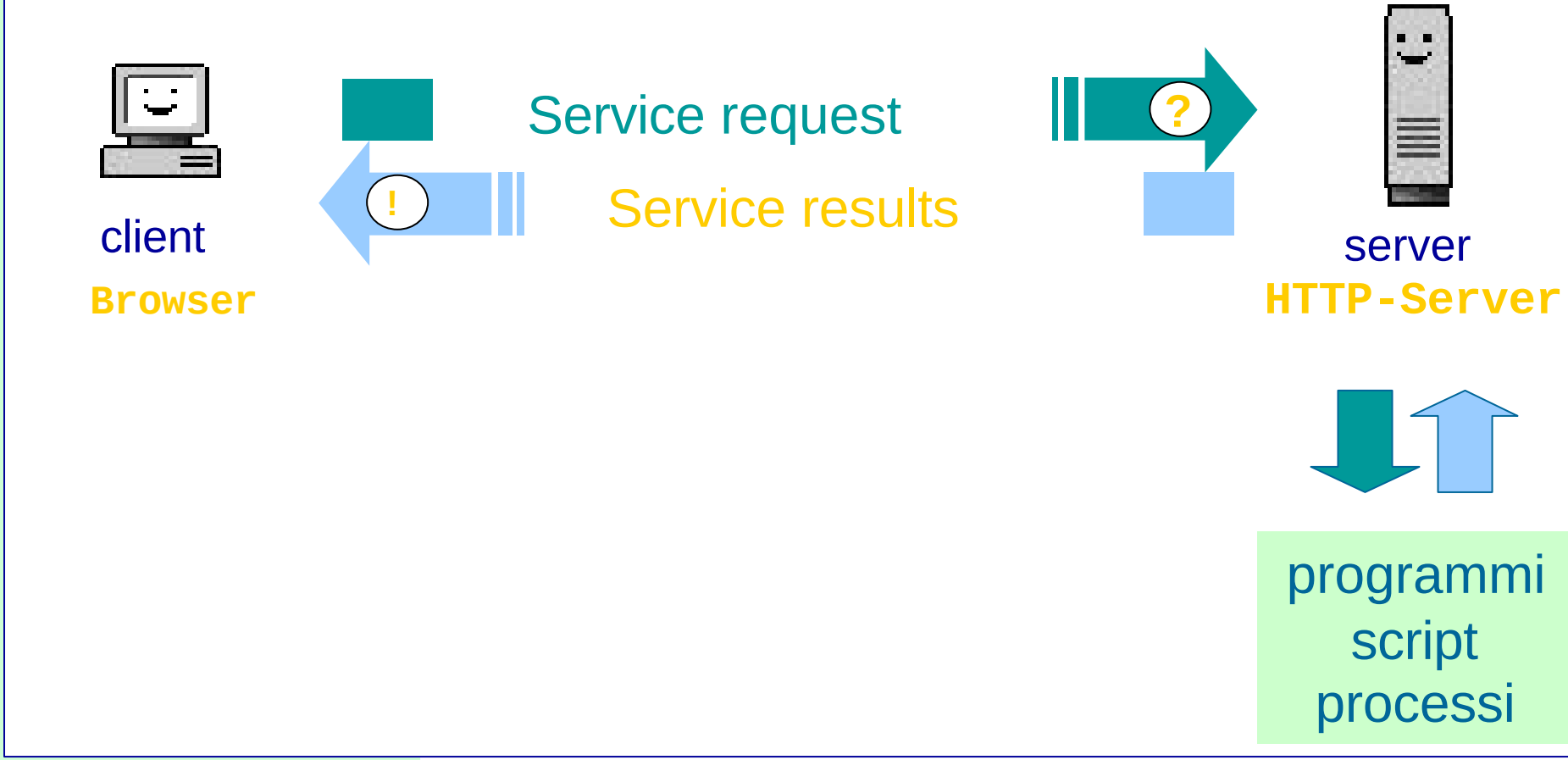
# CGI e PERL

Paola Salomoni
salomoni@csr.unibo.it
martedì 21 novembre 2000



# Programazione Server-Side



# CGI

↗CGI (Common Gateway Interfaces) definiscono come:

- il client effettua il passaggio dei dati di input e la chiamata al programma sul server,
- il server restituisce i dati di output del programma al client.

↗La CGI deve funzionare all'interno del protocollo HTTP che è usato come supporto in entrambe le direzioni.



# CGI: fasi

- Fasi:
  - ① (C): raccolta dati dal lato client mediante form
  - ② (C): invocazione del programma server e spedizione dei dati
  - ③ (S): lancio del programma
  - ④ (S): per prima cosa il programma decodifica i dati
  - ⑤ (S): produzione dell'output del programma e reindirizzamento al client
  - ⑥ (C): visualizzazione dell'output (HTML)



# ① Acquisizione dei dati (form)

- Il tag HTML per il data input è `<form></form>`
- Gli attributi tipici del tag `<form>` sono:
  - `action= URL`
  - `method="GET”|“POST”`
- La form consente di raccogliere dati di diversi tipi e con diversi stili di interazione.





# Associazione nome-valore

- I dati raccolti con la form devono essere mantenuti in associazioni

  nome = valore

- Ogni elemento della form deve quindi avere un nome.
- Il programma che riceverà i dati da elaborare, ricostruirà le strutture dati a partire dalle associazioni nome=valore



# Tipologie di input

- Le principali tipologie di input sono:
  - text: `<input type="text" name="nomevar">`
  - hidden:
    `<input type="hidden" name="nomevar" value="val">`
  - password: `<input type="password" name="nomevar">`
  - checkbox:
    `<input type="checkbox" name="varname" value="on">`
  - radiobutton:

```
<input type="radiobutton" name="varname" value="val1">
<input type="radiobutton" name="varname" value="val2">
```





# Tipologie di input (2)

- Altre tipologie:
  - textarea:

```
<textarea name="nomevar" cols="numcol" rows="numrows">
 testo di default </textarea>
```

  - select/option (menù):

```
<select name="nomevar">
    <option value="val1"> valore 1
    <option value="val2"> valore 2
</select>
```





# ② submit

- Per spedire il contenuto della form ad un programma su server si deve cliccare sul bottone di submit.
- Per inserire il bottone di submit si utilizza il tag seguente:

```
<input type="submit" value="Click me">
```

- Al click del bottone di submit è associata l'invocazione del programma server side associato all'attributo **`action`**.





# submit (2)

- I dati sono sono spediti al server in coppie nomevariabile=valore separate da un simbolo &.
- Alcuni caratteri sono sostituiti da simboli (ad esempio + viene sostituito dallo spazio) e codici.

```
nunky.csr.unibo.it/~salomoni/p.sh.cgi?nome=paola&cognome=salomoni
&indirizzo=via+Sacchi
```

(NB: l'esempio e' sul metodo GET)



# ③ environment

- Dal lato server il programma per partire deve essere ospitato in un ambiente (environment).
- All'interno dell'ambiente il server definisce delle speciali variabili, dette appunto variabili d'ambiente, il cui valore viene ricavato dalle informazioni inviate dal browser
- le variabili d'ambiente danno informazioni sul client, sul server e sull'interazione tra client e server



# variabili d’ambiente (server)

- Le variabili d’ambiente più importanti che descrivono il server sono:
  - SERVER_SOFTWARE: il nome e la versione del server http che risponde alla richiesta cgi;
  - SERVER_NAME: il nome del server http che può essere un alias di DNS o un indirizzo IP numerico;
  - SERVER_PORT: il numero di porta a cui è indirizzata la richiesta (porta 80, generalmente);



# variabili d’ambiente (client)

- Le variabili d’ambiente più importanti che descrivono il client sono:
  - REMOTE_HOST: il nome dell'host che invia la richiesta;
  - REMOTE_ADDR: l'indirizzo IP dell'host che invia la richiesta;
  - AUTH_TYPE: il metodo usato dal protocollo di autenticazione dell'utente;
  - REMOTE_USER: il nome dell'utente autentificato.
  - HTTP_ACCEPT: i tipi MIME accettati dal client;
  - HTTP_USER_AGENT: il nome e la versione del browser che ha inviato la richiesta.



# variabili d'ambiente (C/S)

- Le variabili più significative sono quelle che descrivono l'interazione tra client e server:
  - GATEWAY_INTERFACE: la versione delle specifiche CGI;
  - SERVER_PROTOCOL: il nome e la versione del protocollo di comunicazione HTTP;
  - SCRIPT_NAME: il path completo del programma CGI;
  - ….

# variabili d’ambiente (C/S)

- Quelle veramente importanti sono:
  - REQUEST_METHOD: il metodo usato per la richiesta (GET, POST, ed altri);
  - QUERY_STRING: contiene i dati passati dalle richieste GET, ISINDEX, ISMAP;
  - CONTENT_TYPE: per le richieste POST e PUT questa variabile contiene il tipo MIME dei dati passati al cgi;
  - CONTENT_LENGTH: la lunghezza dei dati passati con il metodo POST;



# ④ input

- Sulla base delle variabili del lucido precedente il programma server sa come decodificare i dati in arrivo.
- La decodifica dipende innanzitutto dal metodo usato per il trasferimento (attributo **`method`** del tag **`form`**) che può essere
  - GET o
  - POST





# il programma CGI

- Il programma CGI è un programma che gira dal lato server (quindi deve rispondere alle esigenze architetturali del server)

- Può essere:
  - un programma compilato (C, C++, Fortran, Prolog, ...)
  - uno script ovvero un programma interpretato (tipicamente Perl o Shell)



# il metodo GET

- Quando il programma parte viene predisposto l'ambiente.
- Nella variabile d'ambiente QUERY_STRING vengono inseriti e le coppie nomevariabile=valore separate da &.
- Il programma per ottenere i valori in input deve leggere il valore della QUERY_STRING e decodificare i dati.



# il metodo POST

- Quando il programma parte viene predisposto l'ambiente.
- le coppie nomevariabile=valore separate da &, sono inserite nello standard input del programma
- QUERY_STRING non contiene nulla mentre CONTENT_TYPE e CONTENT_LENGTH contengono rispettivamente il tipo MIME dei dati passati e la loro lunghezza.
- Con tipi MIME diversi da text/plain si possono passare dati strutturati.



# GET vs POST

- La differenza primaria tra GET e POST sta nel passaggio dei valori.
- GET è di default e va bene in tutte le query che non richiedono “data encoding”
- Alcuni casi in cui è meglio POST sono:
  - dati di input “codificati” (es: mail)
  - uso di caratteri non ASCII
  - stringa di input molto lunga



# decodifica dell'input

- In entrambi i casi:
  - GET: preso il valore dalla QUERY_STRING
  - POST: preso il valore dallo stdio

  l'input risulta composto da una lunga sequenza di caratteri in cui:
  - vanno selezionati i valori nelle coppie nomevariabile=valore e assegnati ad opportune variabili interne del programma
  - vanno sostituiti i caratteri protetti (ad esempio + va sostituito dallo spazio)



# ⑤ output

- L'output del programma verrà reinviato al client e perché ciò sia possibile deve essere in un formato compatibile con le CGI
- Le CGI reindirizzano direttamente lo standard output del programma al client
- Tipicamente il formato più adatto per avere un output formattato è l'HTML



# output

- La prima riga da stampare sullo standard output deve indicare il tipo MIME dell'output vero e proprio che seguirà (quindi deve essere conforme a MIME!)

```
echo 'Content-type: text/plain'
echo 'Content-type: text/html'
```

- Questa riga non viene inviata al client ma usata dall'HTTP server per impostare la comunicazione col browser.



# output

- Le successive “stampe” devono essere conformi al tipo MIME specificato:

```
echo ‘ciao!’
echo ‘<HTML><BODY><H1>ciao!</H1></BODY></HTML>’
```

- Queste righe vengono mandate così come sono al client che le visualizza
- Se sono HTML l’output può contenere formattazioni.





# Linguaggi per le CGI

- Vediamo alcuni esempi concreti usando tre linguaggi ‘base’ (UNIX!).
  - **Shell** (interpretato) : è semplicissimo per problemi semplici ma per problemi complessi non è da solo sufficiente. Già per decodificare la stringa di input si appoggia su **sed** e **awk**.
  - **Perl** (interpretato) : semplice e adatto alle CGI perché dotato di primitive sulle stringhe che aiutano a decodificare i dati
  - **C** (compilato): potente, flessibile e efficiente però complesso anche per fare cose semplici.



# Percorso degli esempi

- Solo output:
  - provare a scrivere “Ciao mondo”
  - provare a stampare il valore di alcune variabili d’ambiente
- Con input: provare a decodificare semplici stringhe di Input con i metodi
  - GET
  - POST
- Esempi, più significativi, ad hoc



# Perl: breve introduzione

- Perl (Processing Estraction Report Language) è un linguaggio di scripting pensato per fare amministrazione di sistema con scrip bach
- E' molto adatto per gli script CGI perché:
  - è simile al C ma più semplice
  - ha primitive che rendono molto agile operare con le stringhe
  - interagisce bene (ovviamente) con il S.O.



# Perl: breve introduzione

- E’ un linguaggio interpretato, quindi va eseguito insieme al suo interprete.
- Per questo motivo i file perl iniziano tutti con il path (relativo) dell’interprete perl, tipicamente:

```
#!/usr/bin/perl
```

- Tutti i nomi delle variabili sono case-sensitive e iniziano con il simbolo $.



# Perl: array associativi

- Un array associativo è un vettore che consente l'accesso ai suoi elementi utilizzando delle stringhe come indici.
- Il nome deriva dal fatto che se anche gli elementi dell'array sono stringhe, l'array associativo ha lo scopo di "associare" stringhe a stringhe in coppie.
- Sono sostanzialmente sistemi di hash, ovvero la ricerca avviene su strutture non ordinate.



# array associativi

- Gli array associativi iniziano col simbolo % symbol (mentre gli array scalari iniziano con @)

- Sono definite alcune funzioni:
  - **`keys`** ritorna la lista delle  chiavi del vettore associativo (come vettore scalare)
  - **`values`** ritorna la lista dei  valori del vettore associativo (come vettore scalare)
  - **`each`**  ritorna una lista di coppie di elementi chiave, valore



# Perl: default values

- **Valori di default:**
  - **$_** è lo scalare di default: in funzione del contesto, fornisce accesso al valore di default per qual contesto.
  - @_ è l'array di default: opera come $_ ma sull'array default per qual contesto.

```
@colori = ("rosso", "giallo", "blu");
foreach (@colori)
{
        print "colore attuale: $_\n";
 }
```





# Perl: funzioni sulle stringhe

- Alcune funzioni:
  - join concatena più stringhe
  - lenght restituisce la lunghezza di una stringa
  - grep come sui sistemi UNIX
  - split scompone la stringa sulla base di espressioni regolari



# Perl: operatori sulle stringhe

- Operatori sulle stringhe:
  - . concatenazione
  - eq verifica che le stringhe siano uguali
- Operatori logici:
  - && AND
  - || OR
  - ! NOT





# Perl: espressioni regolari

- Una espressione regolare definisce un pattern che verrà cercato
- Ogni espressione regolare è rinchiusa tra /
- Caratteri speciali:
  - un simbolo \ seguito da un carattere non alfanumerico corrisponde esattamente a quel carattere (\\ corrisponde a \ )
  - [...], corrisponde ad ogni carattere che sia tra [ e ]



# Caratteri speciali

- Altri caratteri speciali sono:
  - \t  cerca una tabulazione
  - \s  cerca uno spazio o una tabulazione
  - \S  cerca ogni carattere che non sia una tabulazione
  - \n  cerca una "newline"
  - \w  cerca ogni singola lettera, cifra ma anche _
  - \W  cerca ogni cosa che non sia una lettera una cifra o _
  - \d  cerca ogni singola cifra da 0 a 9
  - \D  cerca ogni carattere che non sia una cifra



# Altri operatori sulle RE

- Altri operatori importanti sulle RE sono:
  - (), raggruppa più elementi in un pattern da cercare una volta
  - *, da un minimo di 0 volte
  - + da un minimo di 1 volta
  - ? 0 oppure 1 volta
  - s -- Opera una sostituzione



# ES: ciao mondo in SH

```
#!/bin/sh
# ciao.sh.cgi

echo Content-type: text/html
echo

# html header
echo "<HTML><HEAD><TITLE>ciao!</TITLE></HEAD>"

# html body
echo "<BODY><H1>CIAO MONDO!</H1></BODY></HTML>"
```



# ES: variabili in SH

```
#!/bin/sh
# varabili.sh.cgi

echo Content-type: text/plain
echo

# contenuto di alcune variabili d'ambiente
echo REQUEST_METHOD = $REQUEST_METHOD
echo SCRIPT_NAME = $SCRIPT_NAME
echo QUERY_STRING = $QUERY_STRING
echo HTTP_USER_AGENT = $HTTP_USER_AGENT
echo CONTENT_TYPE = $CONTENT_TYPE
echo CONTENT_LENGTH = $CONTENT_LENGTH
```



# ES: ciao mondo in Perl

```
#!/usr/bin/perl
# ciao.perl.cgi

print “Content-type: text/html\n\n”;

# html header
print "<HTML><HEAD><TITLE>ciao!</TITLE></HEAD>\n"

# html body
print "<BODY><H1>CIAO MONDO!</H1></BODY></HTML>\n"
```



# ES: variabili in Perl

```
#!/usr/bin/perl
# varabili.perl.cgi

print “Content-type: text/plain\n\n”;

# contenuto di alcune variabili d’ambiente
print “REQUEST_METHOD = $ENV{REQUEST_METHOD}\n”;
print “SCRIPT_NAME = $ENV{SCRIPT_NAME}\n”;
print “QUERY_STRING = $ENV{QUERY_STRING}\n”;
print “HTTP_USER_AGENT = $ENV{HTTP_USER_AGENT}\n”;
print “CONTENT_TYPE = $ENV{CONTENT_TYPE}\n”;
print “CONTENT_LENGTH = $ENV{CONTENT_LENGTH}\n”;
```



# ES: ciao mondo in C

```
/* ciao.c */
#include <stdio.h>

main(int argc, char ** argv){
printf("Content-type: text/html\n\n");

/* html header */
printf("<HTML><HEAD><TITLE>ciao!</TITLE></HEAD>\n");

/* html body */
printf("<BODY><H1>CIAO MONDO!</H></BODY></HTML>\n");
}
```



# ES: variabili in C

```
/* variabili.c*/
#include <stdio.h>
#include <stdlib.h>

main(int argc, char ** argv){
printf("Content-type: text/plain\n\n");

/* contenuto di alcune variabili d'ambiente */
printf("REQUEST_METHOD = %s\n", getenv("REQUEST_METHOD"));
printf("SCRIPT_NAME = %s\n", getenv("SCRIPT_NAME"));
printf("QUERY_STRING = %s\n", getenv("QUERY_STRING"));
printf("HTTP_USER_AGENT = %s\n", getenv("HTTP_USER_AGENT"));
printf("CONTENT_TYPE = %s\n", getenv("CONTENT_TYPE"));
printf("CONTENT_LENGTH = %s\n", getenv("CONTENT_LENGTH"));
}
```



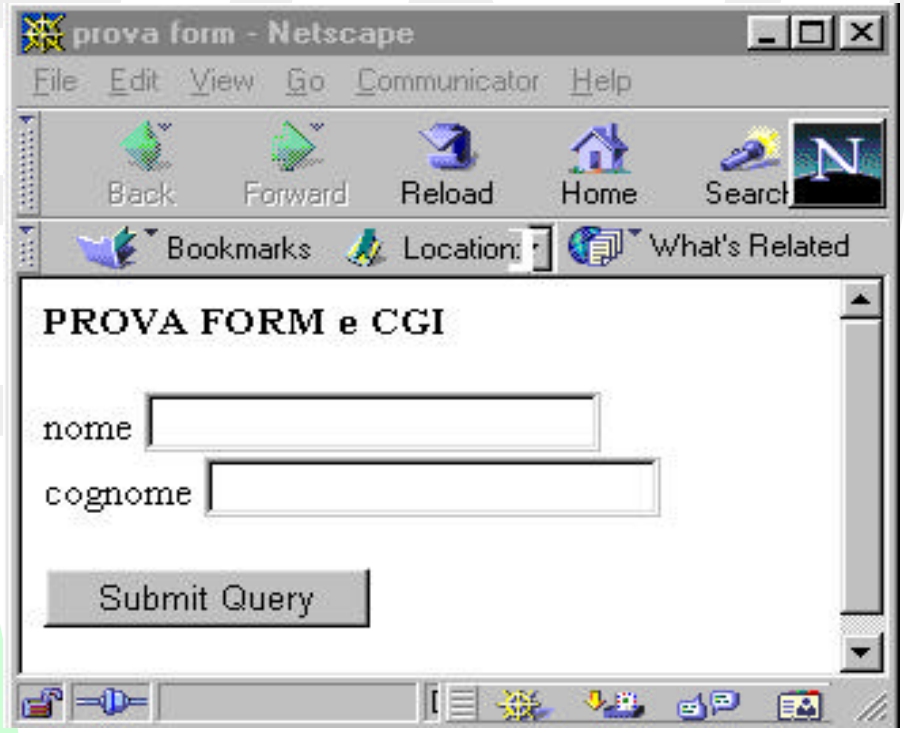
# ES: form nome e cognome

```
<html>
  <head>
    <title> prova form </title>
  </head>
  <body>
   <p>
    <b>PROVA FORM e CGI</b><br>
    <form action="prova.perl.cgi" method="get" >
      nome <input type="text" name="nome" size="20"> <br>
      cognome <input type="text" name="cognome" size="20">
      <br><br>
      <input type=submit>
    </form>
   </p>
  </body>
</html>
```





# ES: form nome e cognome



# ES: nome-cognome in Perl 1

```
#!/usr/bin/perl
# prova.perl.cgi

print "Content-type: text/plain\n\n";

print "prova form nome e cognome\n\n";

# salva il valore della QUERY_STRING in una variabile
$qs = $ENV{'QUERY_STRING'};

# la divide in un array nomevar=valore sulla base degli &
@qs = split(/&/,$qs);

…
```



# ES: nome-cognome in Perl 2

```
foreach $i (0 .. $#qs){
  # converte i + in spazi
  $qs[$i] =~ s/\+/ /g;
  # converte i caratteri esadecimali
  $qs  =~ s/%(..)/pack("c",hex($1))/ge;
  # divide ogni variabile in nomevar e valore
  ($name, $value) = split(/=/,$qs[$i],2);
  # crea un array con indice il nome e valore il valore
  $qs{$name} = $value;
  }

print "\variabili:\n\n";

foreach $name (sort keys(%qs))
  { print "$name=", $qs{$name}, "\n" }
```



# ES: nome-cognome in Perl 3

```
#!/usr/bin/perl
# NOTA: equivalente all'esempio precedente ma POST
# provaPOST.perl.cgi

print "Content-type: text/plain\n\n";

print " prova form POST nome e cognome n\n";

$ct = $ENV{"CONTENT_TYPE"};
$cl = $ENV{"CONTENT_LENGTH"};

# legge dallo standard input la stringa di var=val
read(STDIN, $qs, $cl);

# split it up into an array by the '&' character
@qs = split(/&/,$qs);

… (uguale all'esempio precedente)
```



# ES: il guestbook

- Obiettivi:
  - Inserire un commento nel guestbook firmato dall'autore
  - mandare una mail al proprietario del guestbook
  - tenere traccia degli accessi al guestbook
- Script che risponde a queste richieste con possibilità di personalizzazione.



# Riferimenti


- William E. Weinman, *The CGI Book*, New Riders
- L. Wall, T. Christiansen, J. Orwant, *Programming Perl*, Ed. O'Reilly
- NCSA, *The Common Gateway Interface*, http://hoohoo.ncsa.uiuc.edu/cgi/
- eXtropia, *Instant Web Scripts with CGI/PERL* , http://www.extropia.com/books/instant_web_scripts/index.html
- HTML.IT, *Guida al Perl*, http://www.html.it/perl/




# Riferimenti (1)

- ***E. Wilde, Wilde's WWW, Springer Verlag***
- *Il sito http://www.w3.org/*
- *Il sito http://www.webdav.org/*