

# HTML & CGI

---

Programmazione in Ambienti  
Distribuiti

marco.torchiano @polito.it

V1.5 - © Marco Torchiano 2006

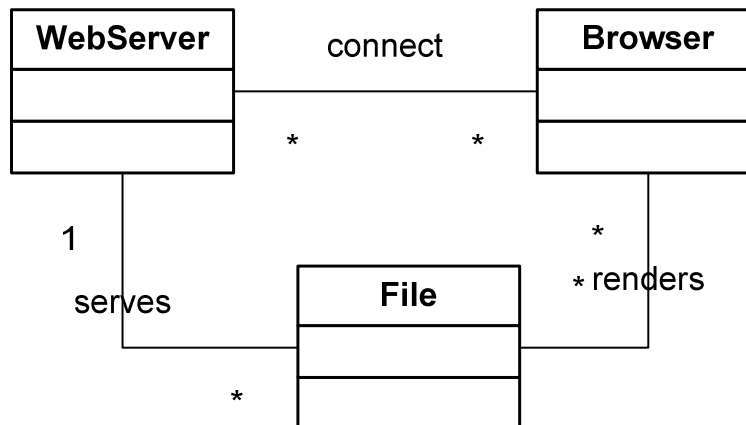
## Contenuti

---

- Siti ed applicazioni web
  - Output del CGI
  - Metodi
    - ◆ GET
    - ◆ POST
  - Esempio di applicazione CGI
  - Libreria CGIC
-

## Sito web

---

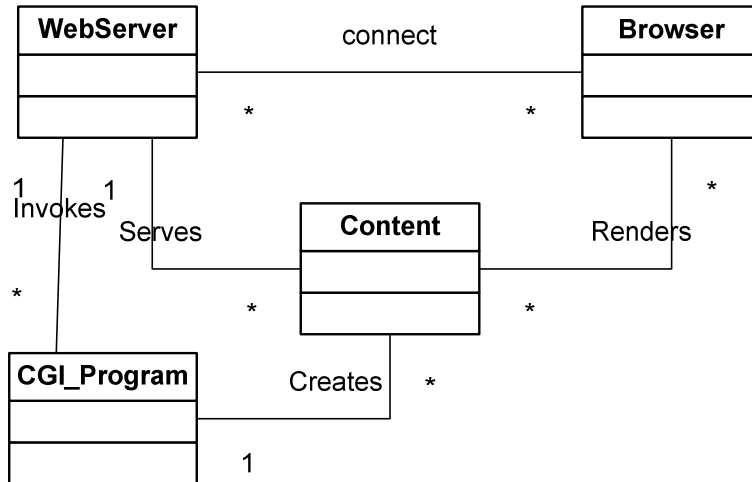


## Installazione web server

---

- Server HTTP
  - ♦ Thttpd (piccolo facile da portare)
- File zip sul sito (per windows)
  - ♦ Su linux usate Apache
- Struttura delle directory (thttpd)
  - ♦ www
    - htdocs : file html
    - cgi-bin : programmi CGI

# Applicazione cgi



# Output del CGI

- Risposta HTTP (RFC 2616)
    - ♦ Status line (fornita dal server HTTP)
    - ♦ Headers
      - Es. Content-type
    - ♦ Riga vuota
      - CRLF
    - ♦ Corpo del messaggio
      - Es. Pagina HTML
- } Prodotti dal CGI

## Esempio

---

```
time_t t;
char* ts;
printf("Content-type: text/html\n\n");
printf("<html><body>\n");
t = time(NULL);
ts = ctime(&t);
printf("Current time: %s\n",ts);
printf("</body></html>\n");
```

---

## Build

---

- Build
    - ♦ Build del programma
    - ♦ `gcc time_cgi.c -o time.exe`
  - Deployment
    - ♦ Copia nella directory dell'http server
      - Es. user home
      - `cp time.exe www/htdocs/cgi-bin/`
  - Accesso al CGI
    - ♦ `http://host/cgi-bin/time.exe`
- 

L'estensione .exe  
serve solo in  
ambiente windows

## Metodo Get

---

- Metodo di default
  - HTML: <form ... method="GET">
  - HTTP: GET url?var1=val1&var2=val2...
  - CGI: variabili d'ambiente
    - Lette con char\* getenv(char\* )
      - Dichiarata in <stdlib.h>
    - REQUEST\_METHOD = "GET"
    - QUERY\_STRING = "var1=val1&var2=val2..."
- 

## Parametri in Get

---

```
char* method = getenv(REQUEST_METHOD);
char* query = getenv(QUERY_STRING);
printf("Method: %s<br>\n",method);
if(strcmp(method,"GET")==0){
    char** vars = extractVars(query);
    int i=0;
    printf("Query: %s<br>\n",query);
    printf("<ul>\n");
    for(i=0; vars[i]!=NULL; ++i){
        printf("<li>Var %d: %s\n",i,vars[i]);
    }
    printf("</ul>");
}
```

---

## Estrazione parametri

---

```
char** extractVars(char* query){
    int i;
    int nParam = 1;
    for(i=0; query[i]!='\0'; ++i){
        if(query[i]=='&')
            nParam++;
    }
    char** result =
    malloc((nParam+1)*sizeof(char*));
    result[0] = query;
    for(i=1; i<nParam; ++i){
        char* sep = strchr(result[i-1], '&');
        *sep = '\0';
        result[i] = sep+1; }
    return result;}

```

---

## Metodo Post

---

- HTML: <form ... method="POST">
  - HTTP:
    - ♦ POST url
    - ♦ *CRLF*
    - ♦ var1=val1&var2=val2...
  - CGI:
    - ♦ Variabili d'ambiente
      - REQUEST\_METHOD = "POST"
    - ♦ Dati sullo standard input (stdin)
      - stdin: "var1=val1&var2=val2..."
-

## Parametri in Post

---

```
char* method = getenv(REQUEST_METHOD);
char* query = getenv(QUERY_STRING);
printf("Method: %s<br>\n",method);
if(strcmp(method,"POST")==0){
    char buff[1024];
    printf("<table><tr><td>stdin<td>");
    while(fgets(buff,1024,stdin)!=NULL){
        printf("%s",buff);
    }
    printf("</table>\n");
}
```

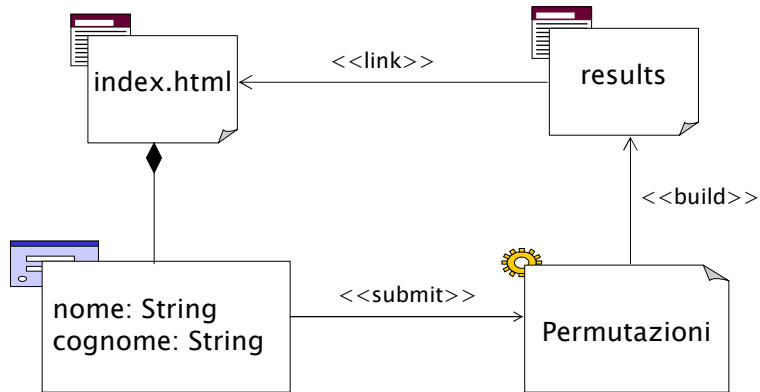
---

## Esempio

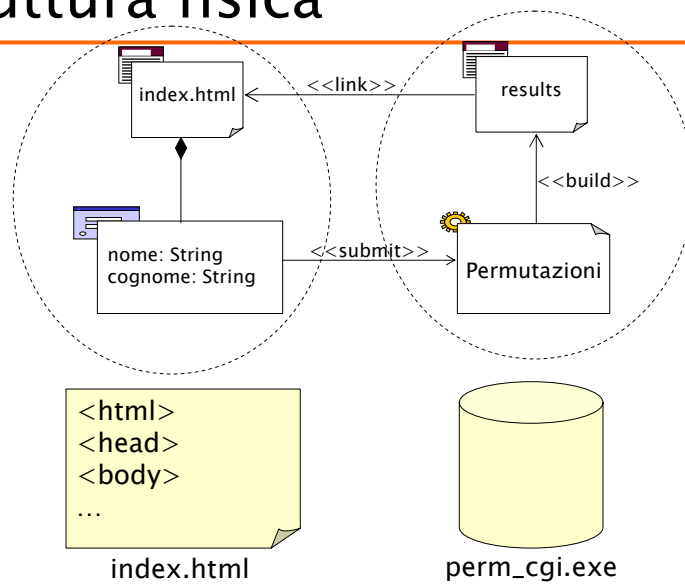
---

- Scrivere un'applicazione CGI in grado di
    - ♦ accettare un nome,
    - ♦ restituire una pagina HTML che contenga una tabella con tutte le permutazioni circolari del nome.
    - ♦ ogni permutazione deve essere numerata a partire da 0
  - La tabella deve presentare una permutazione per riga
-

## Struttura logica



## Struttura fisica



# Index.html

```
<html>
<head><title>Permutazioni
  Circolari</title></head>
<body>
  <form action="/cgi-bin/perm.exe"
    method="GET">
    Nome: <input type="text" name="nome"
      value="" size=20> <br>
    <input type="submit" value="Calcola">
  </form>
</body>
</html>
```

Index.html

form

Nome:

Calcola

# CGI

```
printf("Content-type: text/html\n\n");
printf("<html><body>\n");
printf("<a href=\"index.html\">back</a>\n");
char* query = getenv(QUERY_STRING);
char* nome = strchr(query, '=') + 1;
int l = strlen(nome);
printf("<table border=1>\n");
for(i=0; i<l; ++i){
  printf("\n<tr><td>%d<td>", i);
  for(j=0; j<l; ++j){
    printf("%c", nome[(i+j) % l]);
  }
}
printf("</table></body></html>\n");
```

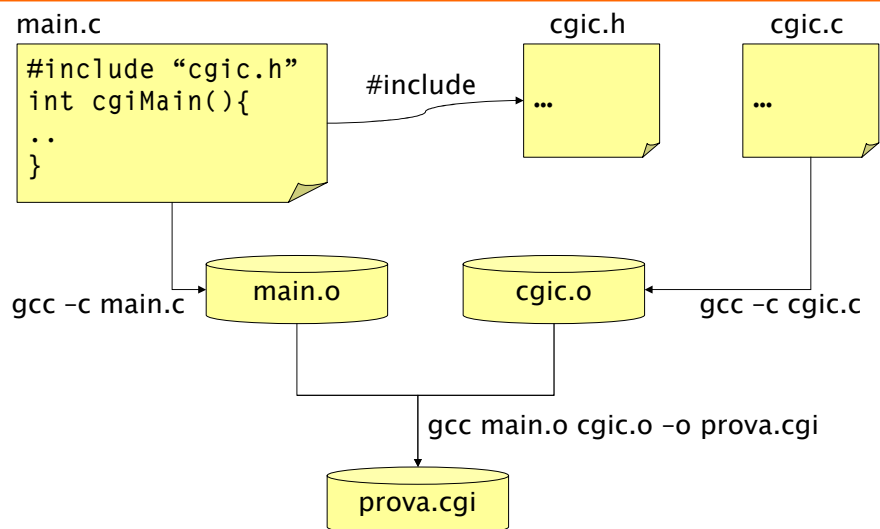
## Libreria CGIC

---

- Permette in maniera semplificata di
    - ♦ Accedere ai parametri
    - ♦ Usare i cookie
    - ♦ Generare la risposta
  - Il programma CGI deve essere definito nel metodo: `int cgiMain()`
    - ♦ Non `main()` come al solito
  - Occorre *linkare* il file con `cgiMain()` al file `cgic.o`
  - CGIC è scaricabile dal sito del corso
- 

## Make

---



## Esempi

---

- Definizione content type
    - ♦ `cgiHeaderContentType("text/html");`
  - Lettura elenco parametri
    - ♦ `char** entries;`
    - ♦ `cgiFormEntries(&entries);`
  - Lettura parametro
    - ♦ `char value[1024];`
    - ♦ `cgiFormString("name",value,1024);`
- 

## Esempio

---

```
cgiHeaderContentType("text/html");
fprintf(cgiOut, "<HTML><BODY>\n");
fprintf(cgiOut, "<a href=\"index.html\">bk</a>");
cgiFormString("nome",buffer,1024);
int l = strlen(buffer);
fprintf(cgiOut, "<table border=1>\n");
for(i=0; i<l; ++i){
    fprintf(cgiOut, "<tr><td>%d<td>",i);
    for(j=0; j<l; ++j){
        int index = (i+j) % l;
        fprintf(cgiOut, "%c",buffer[index]);
    }
}
fprintf(cgiOut, "</table></BODY></HTML>\n");
```

---