

**Programmazione  
in ambiente web**

**Antonio Lioy**  
< lioy@polito.it >

**Politecnico di Torino**  
**Dip. Automatica e Informatica**

---

---

---

---

---

---

---

**World Wide Web (WWW)**

- anche abbreviato semplicemente “web”
- insieme di:
  - protocolli di comunicazione
  - formati dati
- appoggiato su canali TCP/IP

CSS	PNG	JS
HTML	XHTML	
HTTP	FTP	
TCP		
IP		

---

---

---

---

---

---

---

**Protocolli per il web**

- usabili molti protocolli già esistenti (es. FTP)
  - limitazioni / complicazioni perché non pensati per il web
- definito un nuovo protocollo applicativo:
  - HTTP
- le funzionalità ottenibili sono dettate dal protocollo applicativo (es. con FTP solo GET e PUT di file)

---

---

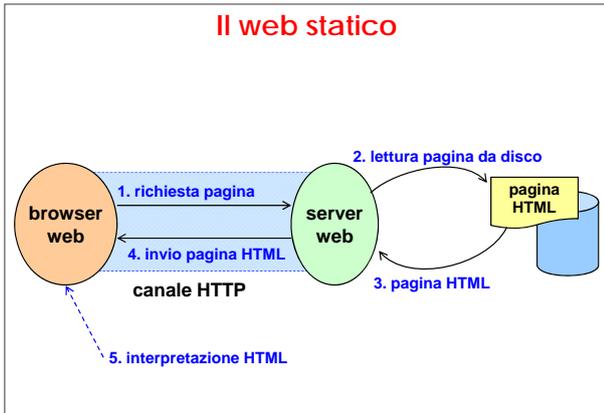
---

---

---

---

---




---

---

---

---

---

---

---

---

- ### Web statico
- pagina web non varia mai il suo contenuto
  - ... finché l'autore non la cambia esplicitamente
  - il contenuto della pagina:
    - non dipende dall'interazione con l'utente
    - non dipende dalle informazioni inviate al server dal client
    - non dipende dall'istante di tempo in cui viene richiesta
  - pagina implementata in HTML / CSS

---

---

---

---

---

---

---

---

- ### Il web statico: pro e contro
- ad ogni pagina web corrisponde un file HTML
  - (+) massima efficienza (basso carico di CPU)
  - (+) possibilità di fare caching delle pagine:
    - in RAM (sul server) o su disco (sul client o proxy)
    - sul server
    - sul client
    - sui proxy
  - (+) pagine indicizzabili dai motori di ricerca
  - (-) staticità dei dati
  - (-) nessuna adattabilità ai client ed alle loro capacità

---

---

---

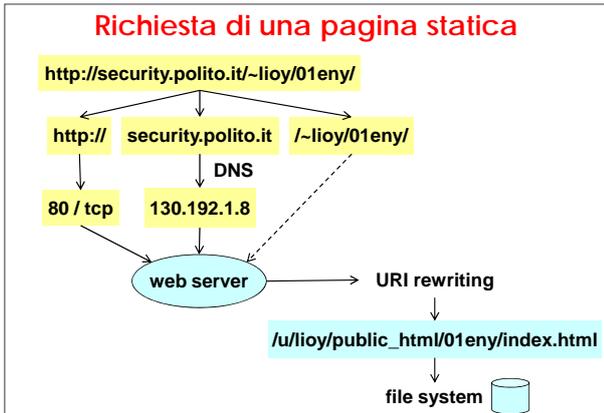
---

---

---

---

---




---

---

---

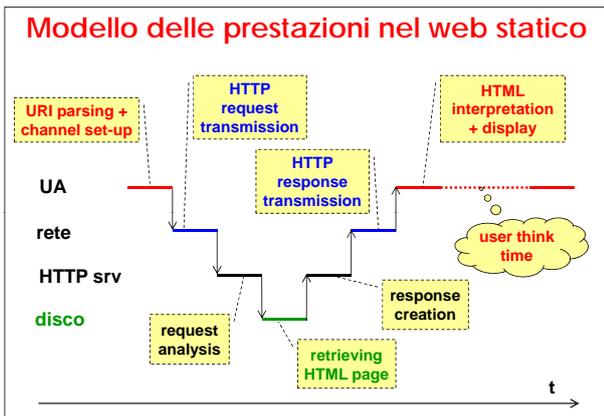
---

---

---

---

---




---

---

---

---

---

---

---

---

- ### Agent, server, proxy e gateway
- User Agent = browser (ma anche spider, robot, ...)
  - Origin Server = fornitore del servizio desiderato
  - possibile avere elementi intermedi tra UA ed OS che fungono sia da client sia da server:
    - gateway
      - interfaccia pubblica per server
      - es. per sicurezza o load balancing
    - proxy (delegato)
      - lavora per conto dei client
      - inoltra la domanda al server o risponde lui stesso tramite una cache
      - anche per autenticazione / autorizzazione

---

---

---

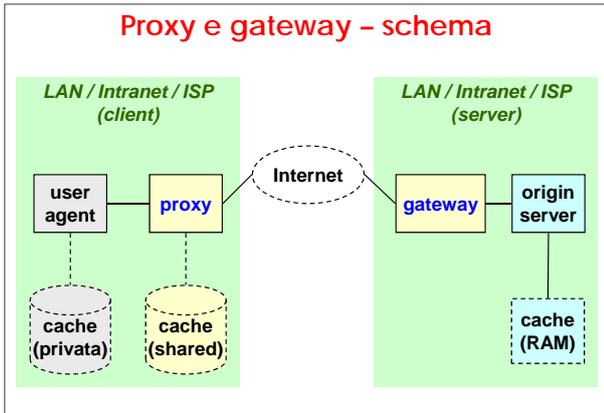
---

---

---

---

---




---

---

---

---

---

---

---

---

- Proxy**
- **cache solo per le pagine statiche**
  - **funzionamento:**
    - trasparente = non altera la richiesta (tranne per le parti obbligatorie)
    - non trasparente = riscrive la richiesta (es. anonymizer)
  - **configurazione su UA:**
    - esplicita (richiede intervento sul client)
    - implicita (richiede intelligenza nella rete)
  - **possibili gerarchie di proxy (es. POLITO, IT, EU)**
  - **spesso usato da ISP per migliorare la velocità di navigazione dei client**

---

---

---

---

---

---

---

---

1. Il protocollo HTTP
  2. Il linguaggio HTML
  3. Le specifiche CSS

---

---

---

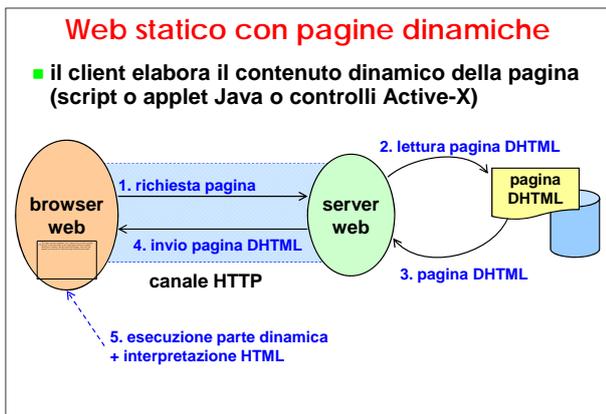
---

---

---

---

---




---

---

---

---

---

---

---

---

- ### Web statico con pagine dinamiche
- pagina varia il suo contenuto in relazione all'interazione con l'utente
    - es. menù a tendina che appare quando il mouse transita su una particolare area
  - si parla genericamente di DHTML:
    - HTML 4.0 o superiore
    - CSS (Cascaded Style Sheet)
    - linguaggio di script lato client
      - ECMA 262 (EcmaScript Edition 3) implementato come JavaScript o JScript
      - VBScript (solo per IE 4.0 o superiore)

---

---

---

---

---

---

---

---

- ### Pagine dinamiche: pro e contro
- presentazione del contenuto variabile
  - (+) efficiente (basso carico di CPU sul server)
  - (-) inefficiente (medio-alto carico di CPU sul client, specie per gli applet)
  - (~) possibilità di fare caching delle pagine
  - (~) pagine indicizzabili dai motori di ricerca (ma solo i dati statici ...)
  - (-) staticità dei dati
  - (-) funzionalità dipende dalla capacità del client (linguaggi di scripting, tipi di applet, CPU, RAM, ...)

---

---

---

---

---

---

---

---

**Pagine dinamiche: applet**

- **due tipi di applet:**
  - applet Java (richiede una JVM sul browser)
  - controllo ActiveX (richiede IE + Wintel)
- **problemi:**
  - compatibilità (quale versione del linguaggio o della JVM?)
  - carico (richiedono esecuzione)
  - sicurezza (esecuzione di un programma vero e proprio):
    - applet Java esegue in una "sandbox"
    - activeX installa una DLL (!)

---

---

---

---

---

---

---

---

**Modello delle prestazioni nel web statico con pagine dinamiche**

- **nessuna differenza con web statico per la parte di rete e lato server**
- **maggior carico computazionale e di memoria lato client:**
  - dipende dalla tecnologia dinamica scelta
  - carico crescente per
    - CSS
    - script lato client
    - controlli Active-X
    - applet Java

---

---

---

---

---

---

---

---

**Client-side scripting**

- **HTML è un linguaggio di descrizione di pagine**
- **unica interattività possibile è seguire i link**
- **aggiunta di interattività alla pagina HTML tramite codice da interpretarsi sul client (UA):**
  - NS e SUN inventano il linguaggio LiveWire, poi chiamato JavaScript (ma non è un subset di Java!)
  - MS inventa VBScript (subset di VBA) e poi JScript
  - accordo per fondere JavaScript e JScript in ECMAScript:
    - standard ECMA-262
    - da tutti chiamato JavaScript (versione >= 1.3)

---

---

---

---

---

---

---

---

### Client-side scripting: a cosa serve ?

- **generare dinamicamente parte della pagina HTML**
  - adattamento al browser o preferenze dell'utente
- **alterare dinamicamente la pagina HTML**
  - AJAX (Asynchronous Javascript And XML) per non ricaricare completamente la pagina dal server)
- **eseguire una funzione associata ad un evento scatenato dall'interazione con la pagina:**
  - es. al completamento di un form, validare i dati prima di inviarli al server
    - si risparmia traffico inutile sulla rete
    - si semplifica l'applicazione server-side
    - ma ... attenzione al rischio di ricevere dati errati!

---

---

---

---

---

---

---

---

### Client-side script e pagine HTML

- **trasmissione di uno script client-side verso il browser:**
  - inserire il codice nella pagina HTML usando il tag <script>
  - inserire un'espressione come gestore di eventi (DOM event handler) dentro alcuni tag HTML
  - specificare un'espressione come valore di un attributo HTML ("script macro" - deprecato)

---

---

---

---

---

---

---

---

### Il tag SCRIPT

- **per inserire uno script all'interno di HTML usare il tag SCRIPT**
- **parametri:**
  - type=text/javascript, text/vbscript, ...
  - src=... (se lo script è presente in un file esterno)

```
<script type="text/javascript">
... codice client-side ...
</script>

<script src="controlli.js" type="text/javascript">
</script>
```

---

---

---

---

---

---

---

---

### Il tag NOSCRIPT

- usare il tag NOSCRIPT per inserire HTML visualizzato solo da quegli UA che:
  - non supportano gli script
  - non conoscono il linguaggio usato
  - hanno gli script disabilitati

```
<noscript>
Attensione! questo sito usa JavaScript per
l'aggiornamento dei dati in tempo reale.
Il tuo browser non supporta JavaScript (oppure
ne &egrave; stata disabilitata l'esecuzione) e
quindi tale funzionalit&agrave; non sar&agrave;
disponibile.
</noscript>
```

---

---

---

---

---

---

---

---

### JavaScript: primo esempio

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"><html>
<head>
<title>JS1</title>
</head>
<body>
<script type="text/javascript">
document.writeln("Ciao!")
</script>
</body>
</html>
```

js1.html

---

---

---

---

---

---

---

---

### JavaScript: tavola dei quadrati

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd"><html>
<html>
<head><title>Tavola dei quadrati</title></head>
<body>
<h1>Tavola dei quadrati</h1>
<script type="text/javascript">
var i;
for (i=1; i<=20; i++) {
document.writeln("<p>"+i+"^2 = "+i*i+"</p>");
}
</script>
</body>
</html>
```

quadrati.html

---

---

---

---

---

---

---

---

**DOM event handler**

- è possibile associare comandi JavaScript ad eventi tramite un "gestore di eventi" (o "event handler")
- sintassi:  
<tag . . . eventHandler = "JavaScript\_code">
- ove:
  - "tag" è un generico tag HTML
  - "eventHandler" è il nome dell'event handler (es. onclick, onfocus, onblur, onsubmit, onreset, onchange, onload, onunload)
  - "JavaScript\_code" è una sequenza di comandi JS (spesso è una chiamata di funzione)

---

---

---

---

---

---

---

---

**DOM event handler**

- **onclick** = click sul testo/immagine associata al tag
- **ondblclick**
- **onfocus** = campo attivo per l'input
- **onblur** = campo non più attivo per l'input
- **onsubmit** = invio dei dati di un form tramite SUBMIT
- **onreset** = modifica dei dati di un form tramite RESET
- **onchange** = onblur + cambiamento del contenuto
- **onload** = al caricamento della pagina
- **onunload** = allo "scaricamento" della pagina (per vederne un'altra o chiudere il browser)
- lista completa nella sezione 18.2.3 di HTML 4.01

---

---

---

---

---

---

---

---

**JS: secondo esempio**

```

... <head>
<title>Esempio JS associato ad onclick</title>
<script type="text/javascript">
function makeRed(x){
  obj = document.getElementById(x);
  obj.style.color="red";
}
</script>
</head><body>
<p id="id1" onclick="makeRed('id1') ">
Click on this text to make it red!
</p>
</body> </html>
    
```

js2.html

---

---

---

---

---

---

---

---

### JS: terzo esempio

- quando lo stesso script serve per più pagine, lo si può scrivere in un file esterno e richiamarlo nella pagina HTML
- il file “.js” deve
  - essere un file di testo
  - avere un nome di lunghezza massima 8 caratteri
  - non contenere il tag <script> ma contenere direttamente il codice Javascript

---

---

---

---

---

---

---

---

### JS: terzo esempio (2)

```
...  
<head>  
<script src="js3.js" type="text/javascript">  
</script>  
</head>  
<body>  
<p id="id1" onclick="makeRed('id1')">  
Click on this text to make it red!</p>  
</body>  
...
```

js3.html

```
function makeRed(x) {  
  obj = document.getElementById(x);  
  obj.style.color="red"; }  
}
```

js3.js

---

---

---

---

---

---

---

---

1. Javascript
2. Form HTML
3. Espressioni Regolari Javascript

---

---

---

---

---

---

---

---

### DOM (Document Object Model)

- una "visione ad oggetti" della pagina HTML
- mette a disposizione una mappa degli elementi del web utilizzando una metafora ad oggetti
- DOM è una struttura dati e non un linguaggio
- deve essere usato assieme ad un linguaggio di scripting lato client (JavaScript, VBscript) che manipoli tali strutture dati
- il W3C raccoglie e cerca di rendere standard il modo con cui vari linguaggi di script interagiscono con la struttura dati alla base di HTML
- DOM livello 1:
  - [www.w3.org/TR/1998/REC-DOM-Level-1-19981001](http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001)

---

---

---

---

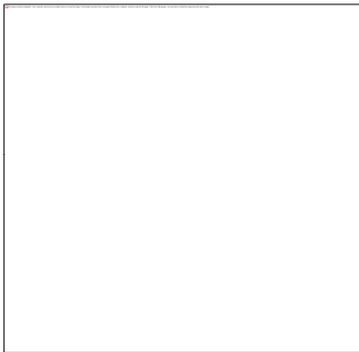
---

---

---

---

### Esempio DOM



---

---

---

---

---

---

---

---

### Esempio DOM: gerarchia oggetti



```
<script ...>
. . .
  name = document.forms[0].elements[0].value
  alert("Ciao " + name)
. . .
</script>
```

---

---

---

---

---

---

---

---

### DOM: dare un nome agli oggetti

- per semplificare l'accesso ad un elemento (invece di usare il riferimento gerarchico) si può assegnargli un "nome" univoco:
  - attributo "name" (disponibile solo per alcuni tag)
  - attributo "id" (disponibile per qualunque tag)
- esempio ("intro" è un riferimento ad una particolare istanza del tag <h1>):

```

<html>
<body>
<h1 id="intro">Introduzione</h1>
. . .
</body>
</html>
```

---

---

---

---

---

---

---

---

### Gerarchia degli oggetti DOM

```

window
|
|--parent, frames, self, top
|
|--location
|
|--history
|
+--document
   |
   |--forms
   |   |
   |   elements (text elements, textarea, checkbox, radio,
   |             password, select, button, submit, reset, ...)
   |--links
   |--images
   |--background
```

---

---

---

---

---

---

---

---

### Problemi di sicurezza degli script client-side

- codice trasmesso dal server al browser
  - potrebbe essere modificato dal client ...
- non registrare dati riservati dentro lo script (es. verifica password, risposta giusta per un test)
  - script leggibile dal client ...
- codice (non fidato!) eseguito sul client:
  - possibili azioni dannose, da semplici (es. cambio impostazioni) a complesse (es. lettura dati introdotti e trasmissione a terzi, attacchi denial-of-service)

---

---

---

---

---

---

---

---

### Un attacco denial-of-service (DoS)

```

<html>
<!-- attacco.html -->
<head>
<title>DoS attack - apre infinite finestre</title>
</head>
<body>
<script type="text/javascript">
  window.open("attacco.html");
</script>
</body>
</html>

```

---

---

---

---

---

---

---

---

### Funzionalità di sicurezza di JS

- set di comandi ristretto
- il codice JavaScript NON può:
  - leggere o modificare file sul sistema client
  - lanciare in esecuzione altri programmi
  - usare nessuna primitiva per collegamenti in rete
    - attenzione! può aprire nuove finestre del browser e far scaricare al browser contenuti dal web
  - eseguire funzioni multithreading

---

---

---

---

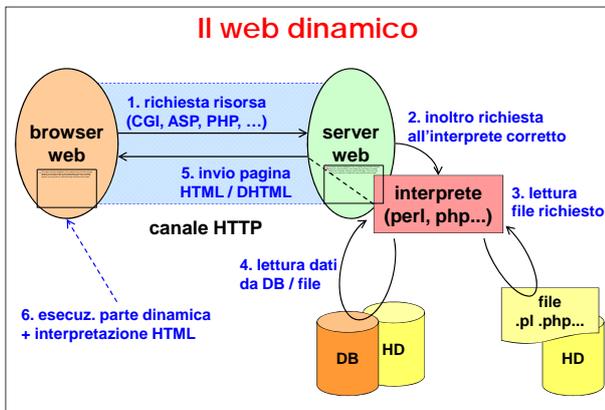
---

---

---

---

### Il web dinamico




---

---

---

---

---

---

---

---

### Web dinamico

- pagina generata dal server in modo dinamico
- varia il suo contenuto informativo in base a:
  - richieste inviate dall'utente
  - contenuto di un database
  - istante di tempo in cui avviene la richiesta
- tecniche per realizzare il web dinamico:
  - CGI
  - linguaggi di script lato server (JavaScript/Jscript, VBScript, PHP, PerlScript, Python, ...)
  - SSI (Server Side Include)
  - servlet, JSP (Java Server Pages)

---

---

---

---

---

---

---

---

### Il web dinamico: pro e contro

- adattamento delle pagine a condizioni variabili:
  - input fornito dal client
  - capacità del client
- (+) massima dinamicità dei dati
- (+) ottima adattabilità ai client ed alle loro capacità
- (-) bassa efficienza (alto carico di CPU)
- (-) impossibile fare caching delle pagine se i parametri di selezione non sono nella URL (es. sono nei cookie)
  - possibili errori se il caching è attivato
- (-) pagine non indicizzabili dai motori di ricerca

---

---

---

---

---

---

---

---

### CGI

- Common Gateway Interface
- <http://hoohoo.ncsa.uiuc.edu/cgi/interface.html>
- RFC-3875
- il web server:
  - attiva l'applicazione CGI
  - le passa eventuali parametri:
    - tramite `stdin` (metodi POST, PUT)
    - tramite una URL modificata (metodo GET)
  - riceve i risultati tramite `stdout`
  - i risultati devono essere in formato web (HTML/CSS/scripting client-side)

---

---

---

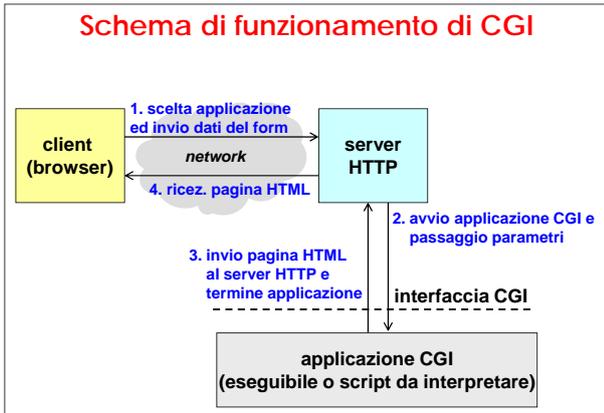
---

---

---

---

---



---

---

---

---

---

---

---

---

### CGI: pro

- metodo generale
- disponibile su tutti i server web (IIS, Apache, ...)
- applicazione scritta in qualunque modo
  - file eseguibile (=più efficienza)
  - script interpretato (=più flessibilità)



---

---

---

---

---

---

---

---

### CGI: contro

- ogni chiamata richiede l'attivazione di un processo:
  - elevato costo di inizializzazione
  - elevata latenza
  - creazione / distruzione di molti processi
- occupazione di memoria proporzionale al numero di processi simultaneamente attivi
- difficoltà di comunicazione tra server web ed applicazione (spazi di memoria differenti)



---

---

---

---

---

---

---

---

**CGI: contro (II)**

- mancano meccanismi per la condivisione di risorse tra programmi CGI
  - ogni accesso ad una risorsa richiede "apertura" e "chiusura" della risorsa
  - non esistono i concetti di sessione né di transazione
- l'interfaccia grafica dell'applicazione web (ossia i tag HTML) è annessa nel codice
- paradigma inadatto ad applicazioni con molti utenti attivi simultaneamente e che richiedono bassi tempi di risposta




---

---

---

---

---

---

---

---

**CGI: possibili miglioramenti**

- uso di variabili di ambiente per comunicare tra server HTTP ed applicazione
- inclusione di uno o più interpreti nel server HTTP:
  - (+) miglior velocità di attivazione
  - (+) miglior comunicazione con l'applicazione
  - (+) minor occupazione di memoria
  - (-) maggior dimensione del server
- pre-attivazione dell'applicazione (in N copie) ed inclusione nel server di un modulo specifico per scegliere una copia libera e comunicare con essa
  - FastCGI

---

---

---

---

---

---

---

---

**Passaggio parametri in input a CGI**

- tre modalità per trasmettere i dati dei form:
  - standard input (quando si usa POST o PUT)
    - il body HTTP viene passato al CGI
  - variabile di ambiente (quando si usa GET):
    - la variabile di ambiente `QUERY_STRING` contiene la parte di URL dopo '?'
  - linea di comando, ossia `argc/argv` (quando si usa `ISINDEX`)
- altre informazioni passate all'applicazione tramite variabili di ambiente (es. `REMOTE_ADDR`, `HTTP_USER_AGENT`)

---

---

---

---

---

---

---

---



### cgic

- libreria ANSI C per programmare CGI
- <http://www.boutell.com/cgic/>
- estrae dati dai form, correggendo errori dei browser
- tratta in modo trasparente GET e POST
- legge i dati dei form o un file uploaded
- funzioni per impostare e leggere i cookie
- tratta correttamente CR e LF nei form di testo
- estrae dati da form (string, int, real, scelte singole e multiple), controllando i limiti dei campi numerici
- carica le var. di ambiente CGI in stringhe non nulle
- compatibile con qualunque server CGI (U\*ix, Win\*)

---

---

---

---

---

---

---

---

### Libwww

- libreria C per scrivere client HTTP+HTML
- usata anche per scrivere robot
- <http://www.w3.org/Library/>

---

---

---

---

---

---

---

---

### Server-side scripting

- tecnologie differenti, ma caratterizzate da avere nel file della pagina codice di scripting mescolato con il template HTML + client-side scripting
- **ASP (Microsoft)**
  - VBscript
  - JScript
  - implementazione anche per Apache (con PerlScript)
- **PHP (open source)**
  - sviluppato per Apache
  - esiste anche per IIS
  - usabile sia come linguaggio generale di scripting sia attraverso CGI

---

---

---

---

---

---

---

---

### Server-side scripting (2)

- **JSP (Sun), tecnologia ibrida**
  - il codice è annegato nel template HTML (come per le altre tecnologia altri script server-side)
  - il codice è costituito da
    - scripting elements (come altri linguaggi server-side)
    - direttive
    - azioni (tag proprietari XML & NS like)
  - le pagine vengono tradotte in servlet dal server Web

---

---

---

---

---

---

---

---

### SSI (Server Side Include)

- **introduce direttive nel codice HTML sotto forma di commento**
  - se SSI *non* è supportato dal server web, le direttive sono ignorate
  - se SSI è supportato, nella pagina HTML restituita al client le direttive sono sostituite con il testo risultato della elaborazione
- **aggiunge nuove variabili di environment**
- **non sostituisce CGI/ASP/..., ma introduce la possibilità di aggiungere dinamismo alle pagine HTML effettuando semplici operazioni**

```
<!--#command tag1=value1 tag2=value2 ... -->
```

---

---

---

---

---

---

---

---

### SSI (2)

- **su IIS le pagine HTML che contengono direttive SSI devono avere estensione “.shtm” o “.shtml”**
- **è possibile configurare opportunamente i server web in modo da elaborare le direttive SSI anche per le pagine con estensioni “.htm” o “.html”**
- **server web**
  - Apache supporta SSI (e XSSSI dalla versione 1.2)
  - IIS supporta solo la direttiva #include di SSI
    - deve essere inserita nella parte di HTML
    - non può essere prodotta da codice ASP
    - in IIS le altre funzionalità di SSI sono ottenibili con oggetti ASP

---

---

---

---

---

---

---

---

### Variabili di environment SSI

- **DOCUMENT\_NAME**: il nome del file corrente
- **DOCUMENT\_URI**: il percorso virtuale a questo documento (come /docs/tutorials/foo.shtml)
- **QUERY\_STRING\_UNESCAPED**: versione della stringa di ricerca inviata dal client, con tutti i caratteri speciali di shell, preceduti da '\'
- **DATE\_LOCAL**: data corrente, fuso orario locale; soggetto al param. `timefmt` nel comando `config`
- **DATE\_GMT**: analogo a **DATE\_LOCAL**, ma relativo al meridiano di Greenwich
- **LAST\_MODIFIED**: l'ultima data di modifica del documento corrente; come altri soggetto a `timefmt`

---

---

---

---

---

---

---

---

### Direttive SSI

- **#config**: permette di impostare alcuni parametri
  - `errmsg`: messaggio da restituire nel caso di errore nel parsing delle direttive SSI
  - `timefmt`: il formato della data e dell'ora; stringa di definizione come quella della funzione di sistema Unix `strftime()`
  - `sizefmt`: il formato della dimensione di file
    - `bytes`: espresso in byte
    - `abbrev`: formato abbreviato (KB o MB)

```
<!--#config errmsg="MSG_ERRORE" -->
<!--#config timefmt="STRINGA_DI_FORMATO" -->
<!--#config sizefmt="bytes" -->
```

---

---

---

---

---

---

---

---

### Direttive SSI (2)

- **#echo**: restituisce il contenuto della variabile (tag: `var`) di environment passata come parametro
  - `var`: nome della variabile
- **#exec**: esegue un comando di shell o uno script CGI il cui nome è passato come parametro e restituisce il relativo output, tag supportati:
  - `cmd`: comando di shell (Unix: `/bin/sh`, Win32: `cmd.exe`) indicato dalla stringa
  - `cgi`: script CGI indicato dalla stringa (path virtuale); nessuna elab. se non la convers. da URL a tag `<A>`

```
<!--#echo var="NOME_VARIABILE_ENV" -->
<!--#exec cmd="PATH_SHELL_SCRIPT" -->
<!--#exec cgi="VIRT_PATH_CGI_SCRIPT" -->
```

---

---

---

---

---

---

---

---

### Direttive SSI (3)

- **#flastmod**: restituisce data e ora dell'ultima modifica di un file (tag: file) il cui nome è passato come parametro

```
<!--#flastmod file="NOME_FILE" -->
```

- **#fsize**: restituisce la dimensione di un file il cui nome è passato come parametro; il formato è modificabile da sizefmt; tag supportati:

- virtual: path virtuale, dalla radice del server (no accesso a CGI script)
- file: path relativo fisico a partire dalla directory corrente (no path assoluti, no uso di './')

```
<!--#fsize virtual="VIRT_PATH_NOME_FILE" -->
<!--#fsize file="REL_PATH_NOME_FILE" -->
```

---

---

---

---

---

---

---

---

### Direttive SSI (4)

- **#include**: inserisce il contenuto di un file nella pagina restituita al client; il nome del file è passato come parametro; tag supportati:

- virtual: path virtuale (no accesso a CGI script)
- file: path relativo fisico a partire dalla directory corrente (no path assoluti, no uso di './')

```
<!--#include virtual="VIRT_PATH_NOM_FILE" -->
<!--#include file="REL_PATH_NOME_FILE" -->
```

- attenzione! il file incluso non può contenere direttive SSI

---

---

---

---

---

---

---

---

### Esempi SSI (1)

- inserisce data ed ora locale in forma standard:

```
<!--#echo var="DATE_LOCAL" -->
```

- inserisce data ed ora locale nel formato specificato:

```
<!--#config timefmt="%A %B %d, %Y" -->
<!--#echo var="DATE_LOCAL" -->
```

---

---

---

---

---

---

---

---

### Esempi SSI (2)

- esegue un'ipotetico script che restituisce il numero di accessi alla pagina ed inserisce il suo output nella pagina:

```
<!--#exec virtual="/cgi-bin/counter" -->
```

- esegue un comando di sistema (attenzione: il testo <DIR> contenuto nell'output del comando dir può provocare una errata formattazione da parte del browser)

```
<!--#exec cmd="ls" --> <!-- per Linux -->
<!--#exec cmd="dir" --> <!-- per Windows -->
```

---

---

---

---

---

---

---

---

### Esempi SSI (3)

- inserisce un piè di pagina comune ad altre pagine

```
<!--#include file="footer.txt" -->
```

- inserisce la data dell'ultima modifica alla pagina corrente; soluzione 1 (se si cambia il nome alla pagina, occorre modificare la direttiva)

```
<!--#config timefmt="%A %B %d, %Y" -->
<!--#flastmod file="tesine.html" -->
```

- inserisce la data dell'ultima modifica alla pagina corrente; soluzione 2 (la direttiva può essere inclusa così com'è in tutte le pagine)

```
<!--#config timefmt="%D" -->
<!--#echo var="LAST_MODIFIED" -->
```

---

---

---

---

---

---

---

---

### Esempi SSI (4)

- impostare un messaggio di errore differente da quello standard in caso di problemi nel parsing delle direttive SSI

```
<!--#config errmsg="[New error message!]" -->
```

- messaggio di errore standard; il codice della direttiva è sostituito dal seguente testo

```
[an error occurred while processing this directive]
```

- messaggio di errore impostato con la direttiva

```
[New error message!]
```

---

---

---

---

---

---

---

---

### Esempio funzionamento SSI

```
<HTML><HEAD><TITLE>
<!--#include virtual="title.inc" -->
</TITLE></HEAD><BODY>
...
<FONT face=sans-serif size=-2>
<BR>Maintained by: <!--#include virtual="author.inc" -->
<BR>Last modified: <!--#echo var="LAST_MODIFIED" -->
</FONT>
```

pagina prima dell'elaborazione  
(come memorizzata sul server)

```
<HTML><HEAD><TITLE>
Esempio di SSI
</TITLE></HEAD><BODY>
...
<FONT face=sans-serif size=-2>
<BR>Maintained by: <B>Antonio Liroy</B>
<BR>Last modified: Thursday, 21-Feb-2002 18:53:28 MET
</FONT>
```

pagina dopo l'elaborazione  
(come inviata al browser)

---

---

---

---

---

---

---

---

---

---

### Esempio funzionamento SSI

- contenuto del file title.inc
  - Esempio di SSI
- contenuto del file author.inc
  - <B>Antonio Liroy</B>
- **NOTA:** i file inclusi con la direttiva include o il risultato di esecuzione di script (direttiva exec)
  - possono contenere testo e HTML
  - devono rispettare la codifica dei caratteri prevista da HTML): es. quantità => quantiti&agrave;
- una volta inclusi devono rispettare i requisiti HTML/CSS (posizione dei TAG, ecc.)

---

---

---

---

---

---

---

---

---

---

### Server-side o client-side?

- **server-side:**
  - (pro) maggiore sicurezza
  - (con) sovraccarico del server
- **client-side:**
  - (pro) elaborazione sul client
  - (con) capacità del client (funzionalità e prestazioni)
  - (con) minore sicurezza (manipolabile dall'utente)
- **in generale:**
  - meglio server-side per sicurezza e funzionalità
  - meglio client-side per migliorare le prestazioni
  - spesso si usano entrambi simultaneamente

---

---

---

---

---

---

---

---

---

---

### Server-side e client-side

- talvolta non sono equivalenti
- esempio (contenuto di prova.asp):

```
<%  
var d=new Date();  
var h=d.getHours();  
var m=d.getMinutes();  
Response.write(h + ":" + m);  
%>  
<script type="text/javascript">  
var d=new Date();  
var h=d.getHours();  
var m=d.getMinutes();  
document.write(h + ":" + m);  
</script>
```

---

---

---

---

---

---

---

---