19. LA PROGRAMMAZIONE LATO SERVER

Introduciamo uno **pseudocodice lato server** che chiameremo **Pserv** che utilizzeremo come al solito per introdurre le problematiche da affrontare, **indipendentemente** dagli specifici linguaggi ai quali successivamente bisognerà poi ricorrere.

Vediamo allora come effettuare una *interrogazione ad una base di dati* situata su di un *Web server* (che per coerenza sarà uno **PseudoWeb**) e controllata da un qualsiasi *DBMS* ed alla quale si possa accedere tramite *un server SQL* (che per coerenza chiameremo **PseudoSQL**).

Le caratteristiche dello **pseudocodice Pserv** che ipotizzeremo sono:
- è un linguaggio di *scripting lato server* ossia interpretato;
- segue *l'approccio* basato su Web server (come ASP, JSP, PHP);;
- dispone di funzioni per *l'accesso ai più diffusi database relazionali* (come Oracle, Sybase, MySQL, Access, etc.)

Ricapitolando volendo essere indipendenti da specifici linguaggi, database e Web server ipotizziamo si essere nella seguente situazione:

Web server (Esempio Apache, IIS, etc.**)**	**PseudoWeb**
Server SQL (Esempio SQLserver, MySQL, etc.)	**PseudoSQL**
Linguaggio di programmazione lato server (Esempio ASP, JSP, PHP etc.).	**Pserv**

N.B. Nel mondo dell'informatica è **indispensabile** che i *Web server* possano interagire con i *DBMS* e con i *server SQL* di produttori differenti ed i server SQL devono a loro volta interagire con i *linguaggi di programmazione lato server.*

1) Configurazione del web server per l'esecuzione di programmi lato server

Ogni Web server dovrà essere configurato correttamente per permettere l'esecuzione di programmi lato server.

Precisazione: *Ogni Web server fa riferimento ad una* ***directory root virtuale*** *sullo spazio fisico del computer server dove è installato (quando un utente in rete si collegherà con il Web server per vedere la home page avrà la sensazione di essersi collegato con la directory root del server mentre in realtà sarà collegato con una sottodirectory fisica del disco del server che funge da root virtuale. Nella root virtuale si troveranno tutte le pagine HTML che il server vuole rendere disponibili all'esterno.*

Nel caso del nostro **Pserv** (così come per tutti i linguaggi di programmazione lato server) occorrerà configurare correttamente il **Web server** affinché:
- il **Web server** possa *integrarsi* perfettamente con **Pserv** ed in particolare:
    - il Web server deve poter riconoscere l'estensione ***.pserv** dei file da eseguire;
    - il Web server deve sapere dove si trova l'**interprete Pserv** da mandare in esecuzione sui file ***.pserv**;
    - il Web server deve conoscere la directory del server che conterrà i file ***.pserv** che potranno essere eseguiti.

- **Pserv** dovrà interagire con il **server SQL** (ed in particolare avere la possibilità di interagire tramite funzioni primitive con particolari DBMS installati sul server).

2) Esecuzione di programmi lato server

Una volta **installato configurato correttamente il Web server** il nostro **Pserv** saprà:
- dove è posizionato il suo **interprete** che chiameremo *pserv.exe*;
- qual è la **directory virtuale** del Web serve che conterrà i file *.pserv.

Le istruzioni di **Pserv** possono:
- essere inserite all'interno di file HTML tramite opportuni tag (**Pserv embedded**);
- far parte di file di **soli comandi di Pserv.**

In entrambi i modi per scrivere un file di istruzioni oppure un file di comandi occorre:
a) **scrivere** un file con un *editor di testo*;
b) **salvare** il file con *estensione *.pserv*;
c) **inserirlo** nella *directory concordata* con il web server (la directory di *pubblicazione* degli script lato server).

Nel caso di *Pserv embedded* per poter inserire istruzioni di *Pserv* all'interno di un file HTML sono possibili due **tecniche**:

(*) utilizzare una coppia di **marcatori o tag** che delimitino l'inizio e la fine delle istruzioni.
Nel nostro pseudolinguaggio lato server supponiamo di utilizzare i due tag **<?pserv** e **?>** ossia

```
<?pserv
    Istruzioni Pserv
?>
```

Esempio:

```
<?pserv
    Scrivi ("Testo stampato con Pserv")
?>
```

(*) utilizzare il tag **SCRIPT** dell'HTML nel seguente modo

```
< SCRIPT Language = "Pserv" >
    Istruzioni Pserv
< /SCRIPT >
```

Esempio:

```
< SCRIPT Language = "Pserv" >
    Scrivi ("Testo stampato con Pserv")
< /SCRIPT >
```

*N.B. Noi useremo la prima tecnica descritta che prevede l'uso dei due tag **<?pserv** e **?>** perché sono facilmente individuabili all'interno di una pagina HTML.*

Il file di istruzioni così creato può essere mandato in esecuzione da un qualsiasi browser digitando un indirizzo che ha la seguente sintassi:

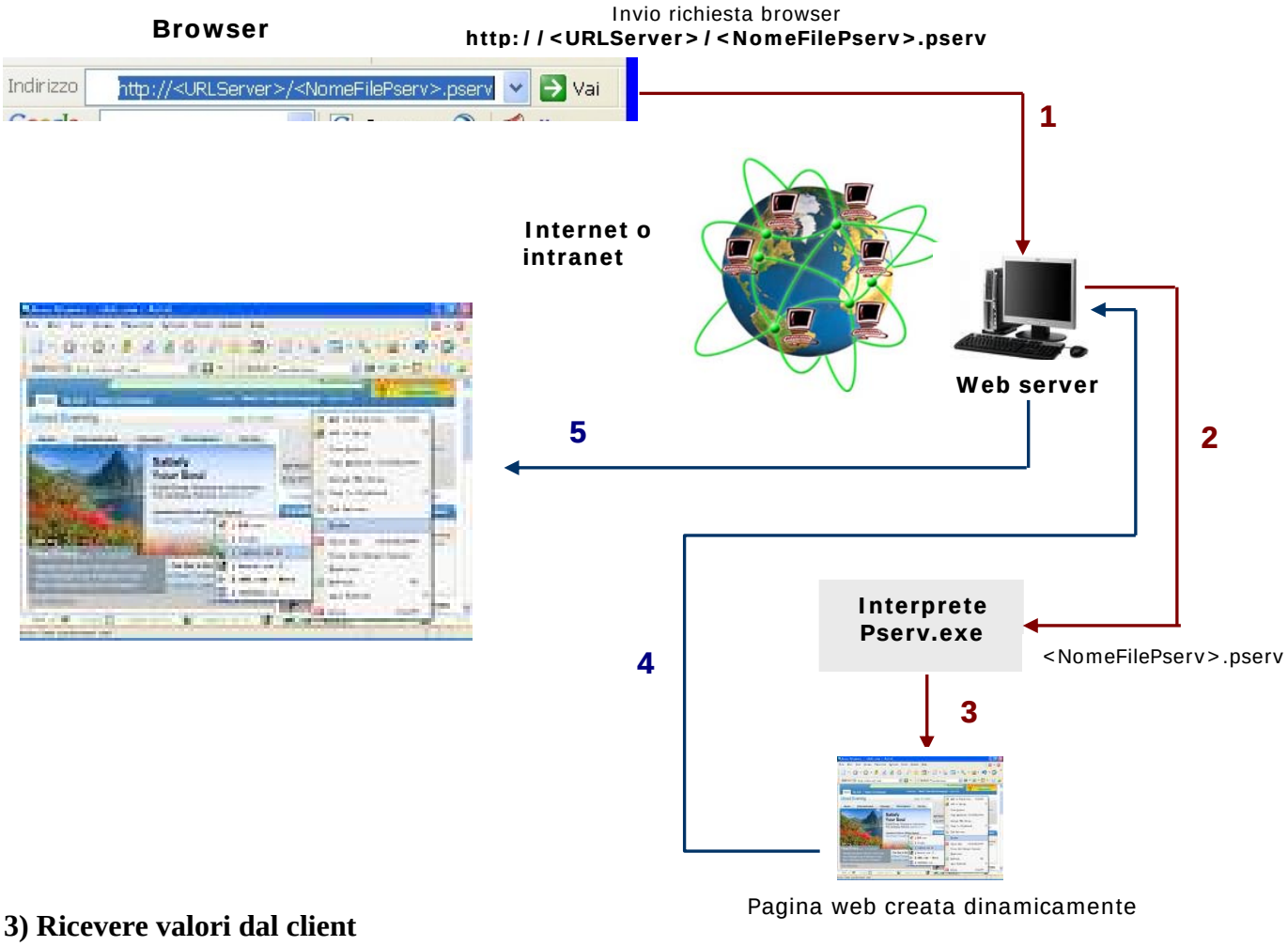
http://<URLServer>/<NomeFilePserv>.pserv

dove:
- <URLServer> è l'URL del server dove sono state memorizzate le pagine *.pserv. E' un uqlsiasi indirizzo IP di un server su di una rete locale o su Internet;
- <NomeFilePserv> è il nome del file contenente le istruzioni Pserv.

N.B. Su molti Web server l'URL **localhost** indica un indirizzo particolare ossia quello dell'host che ha effettuauo la chiamata (per default il suo IP è 127.0.0.1)
Quindi per testare la nostra applicazione in locale possiamo digitare in un browser il seguente indirizzo

http://localhost/prova.pserv

Schematicamente abbiamo:



3) Ricevere valori dal client

E' importantissimo poter passare valori dal file HTML scaricato sul client al file contenente i comandi Pserv presente sul server.

Supponiamo di avere il seguente file di nome **visualizza.pserv** che ha come obiettivo quello di mostrare una stringa qualsiasi inserita dall'utente (notiamo che *$Messaggio* non ha valore definito)

```
<?pserv
    Scrivi ($messaggio)   /* stampa una qualsiasi stringa prelevata dal client */
?>
```

Creiamo il file **visualizza.html** per l'inserimento del testo da visualiizzare

```
……
< BODY>
        <FORM METHOD = "post" ACTION = http://<URLServer>/visualizza.pserv>
        testo:<INPUT TYPE = "text"   NAME = "messaggio" >
        <INPUT TYPE = "submit"   VALUE = "Ok" >
        </FORM>
</BODY>
…..
```

Osservando il codice **HTML** scritto notiamo che:

- l'attributo **ACTION** del form specifica il nome del file Pserv da chiamare sul server.
Una volta premuto il bottone *submit* i dati contenuti nei campi del form verranno inviati attraverso il protocollo HTTP al Web server al quale giungerà la richiesta di esecuzione della pagina *visualizza.pserv* seguita dal valore della variabile *messaggio*
Questo valore sarà associato alla variabile $messaggio per poter essere visualizzato.

N.B. *uindi lo scambio dati tra* ***HTML*** *e* ***Pserv*** *avviene utilizzando nei file* ****.pserv*** *variabili con lo stesso nome dei campi dei form HTML in cui è presente l'invocazione dei file *.pserv da eseguire.*

- l'attributo **METHOD** del form specifica la modalità con la quale il browser trasferirà le informazioni sul server.

N.B.

http://<URLServer>/visualizza.pserv&messaggio

4) Interazione con un server SQL tramite un programma lato server

Per poter interagire tramite un browser con una base di dati memorizzata su un Web server occorre che nel file con estensione *.pserv venga:

a) **stabilita una connessione con il server SQL** attraverso la pseudoistruzione
SQLConnetti (<URLserver>, <Utente>, <Password>)
dove:
- <*URLserver*> è l'URL del server SQL (può essere un qualsiasi indirizzo IP di un server in una rete locale o di Internet);
- <*Utente*> *e* <*Password*> sono il nome dell'utente e la password a lui assegnata per essere riconosciuti dal serverSQL (vengono anche chiamati **parametri di connessione**).

Questa pseudoistruzione restituisce un valore di tipo intero chiamato **identificativo o ID di connessione** che individua in caso di successo univocamente la connessione aperta.

b) **selezionato un database** attraverso la pseudoistruzione
SelezionaDB (<IDConnessione>, <NomeDB>)
dove:
- <*IDConnessione*> è l'identificativo della *connessione* stabilita con il server SQL a seguito di una precedente istruzione *SQLConnetti*;
- <*NomeDB*> é il nome del database da utilizzare.

c) **impostata l'istruzione SQL** memorizzandola all'interno di una variabile.

d) **eseguita l'istruzione SQL** non appena si è stabilita la connessione con il database di interesse e si è preparata l'istruzione SQL desiderata, si deve eseguire utilizzando la pseudosistruzione
Esegui (<IDConnessione>, <Istruzione>)
dove:
- <*IDConnessione*> è l'identificativo della *connessione* stabilita con il server SQL a seguito di una precedente istruzione *SQLConnetti*;
- <*Istruzione*> é una stringa o il nome di una variabile stringa che contiene il comando SQL.

e) **chiusa una connessione con il server SQL** attraverso la pseudoistruzione
Chiudi (<IDConnessione>)
dove:
- *<IDConnessione>* è l'identificativo della *connessione* stabilita con il server SQL a seguito di una precedente istruzione *SQLConnetti*.

Schematicamente abbiamo: