FONDAMENTI DI

PHP

WEB STATICO E WEB INTERATTIVO

In principio il Web era una semplice collezione di pagine HTML statiche collegate tra di loro tramite link ipertestuali. Lo scopo principale era quello di rendere disponibile su Internet documenti correlati come in un enorme ipertesto.

WEB STATICO E WEB INTERATTIVO

La necessità di una maggiore interattività tra l’utente e il server Web, nata soprattutto nel momento in cui grandi aziende hanno visto nella rete delle reti, un nuovo veicolo commerciale, ha indirizzato gli sforzi nello sviluppo di strumenti per rendere il Web sempre più "dinamico".

WEB STATICO E WEB INTERATTIVO

Un classico esempio è dato dai motori di ricerca: l'utente inserisce in una form una parola chiave e invia al server la richiesta restando in attesa di una risposta in tempo reale. Dalla parte del server viene a questo punto attivata una procedura in grado di interrogare un database e di generare una pagina HTML con la lista dei link richiesti.

WEB STATICO E WEB INTERATTIVO

Per fare questo, l'HTML non era sufficiente fu quindi introdotto uno standard che consentisse di far comunicare le richieste HTTP con una applicazione residente sul server, nacque lo standard CGI (Common Gateway Interface).

CLIENT-SIDE SCRIPTING

L’interfaccia CGI tuttavia presenta dei limiti: ad esempio anche per semplici elaborazioni vengono richieste molte risorse al server, e in certi casi ciò può portare a ritardi sulla risposta o ad un eccessivo carico sul server stesso.

CLIENT-SIDE SCRIPTING

Per superare questo genere di problemi, Netscape prima e Microsoft dopo, hanno pensato di permettere ai loro browser, di interpretare particolari linguaggi, detti linguaggi di scripting (JavaScript, Jscript e VBscript), che permettono al client di effettuare alcune semplici elaborazioni, permettendo sicuramente al server di alleggerire il carico e dando una spinta decisiva verso le attuali capacità interattive del Web.

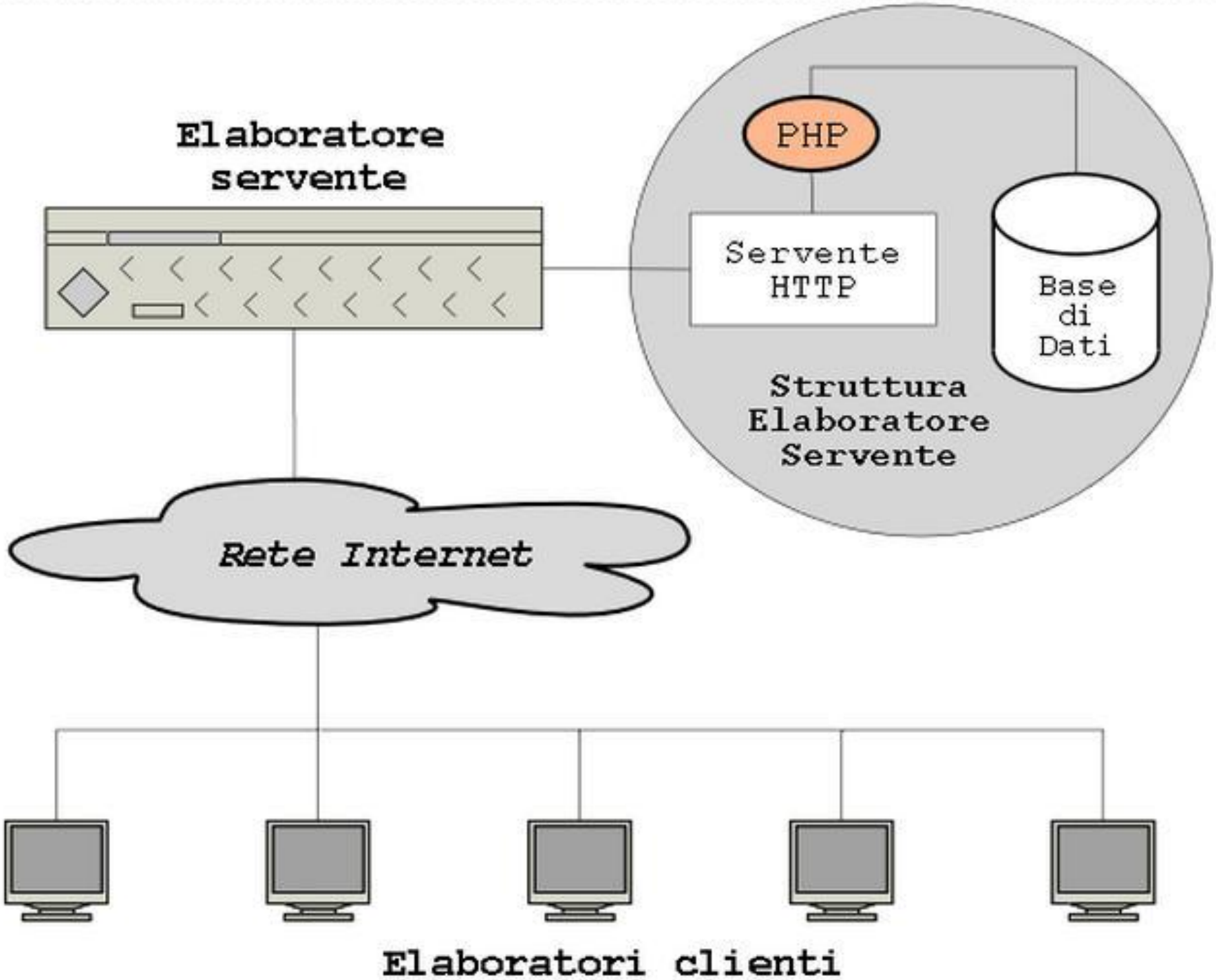
INTRODUZIONE A PHP

Il successo ottenuto dal client-side scripting ha portato in breve tempo allo sviluppo di linguaggi di scripting anche per il lato server, iniziando prima con ambienti specializzati, come ad esempio Borland IntraBuilder dove è Javascript il linguaggio sever-side utilizzato e poi orientandosi verso linguaggi di scripting integrati nel server Web, come ad esempio in Netscape Enterprise Server.

PHP nasce cavalcando questa tendenza.

STORIA DEL PHP

Il PHP nasce a metà del 1994 dalle mani di Rasmus Lerdorf. Egli sviluppo una prima versione mai ufficializzata che utilizzò sulle sue pagine. Ma il successo non tardò, e già nella prima parte del 1995 usci la prima versione del **P**ersonal **H**ome **P**age (in seguito rinominato **P**hp **H**ypertext **P**reprocessor). Da lì a poco l'engine, anche grazie alla sua filosofia free, si è affermato ed adesso vanta qualcosa come 150.000 siti che implementano questo linguaggio.


Elaboratore servente
PHP
Servente HTTP
Base di Dati
Struttura Elaboratore Servente
Rete Internet
Elaboratori clienti

COS’È PHP

PHP è un linguaggio implementato lato server (server-side HTML-embedded scripting language). Il suo funzionamento è molto semplice ed efficace.
Il Web Server (nel nostro caso Apache) restituisce al client (browser) pagine HTML, rendendo il codice PHP trasparente all'utente finale. Visualizzando il codice delle pagine (dal lato client) si vedrà solo HTML puro e di fatto il sorgente della pagina PHP può essere modificato solo dal Web Master.

PHP è una tecnologia utilizzata dal lato server che permette di creare dei siti dinamici.

Generalmente viene utilizzata associata ad altri prodotti anch'essi Open Source, che servono a completare l'offerta di un sito dinamico. Tipicamente:

- Sistema Operativo Linux
- Web Server Apache
- DATABASE Mysql
- Linguaggio di scripting e relativo interprete server- side PHP

Linux

- Derivato dal UNIX, è una versione Open Source che gira su Personal Computer

Apache

- Programma che permette di effettuare la gestione dei siti Internet

Mysql

- DBMS (Database Management System) basato su un'architettura client-server; gestisce basi di dati in cui è possibile memorizzare le informazioni

  che devono sopravvivere alla singola sessione di utilizzo (per esempio, dati sugli utenti, sui prodotti, ecc.)

php

- Questo strumento è composto dal linguaggio di scripting e dall'interprete di tale linguaggio e permette di generare pagine con contenuto dinamico .

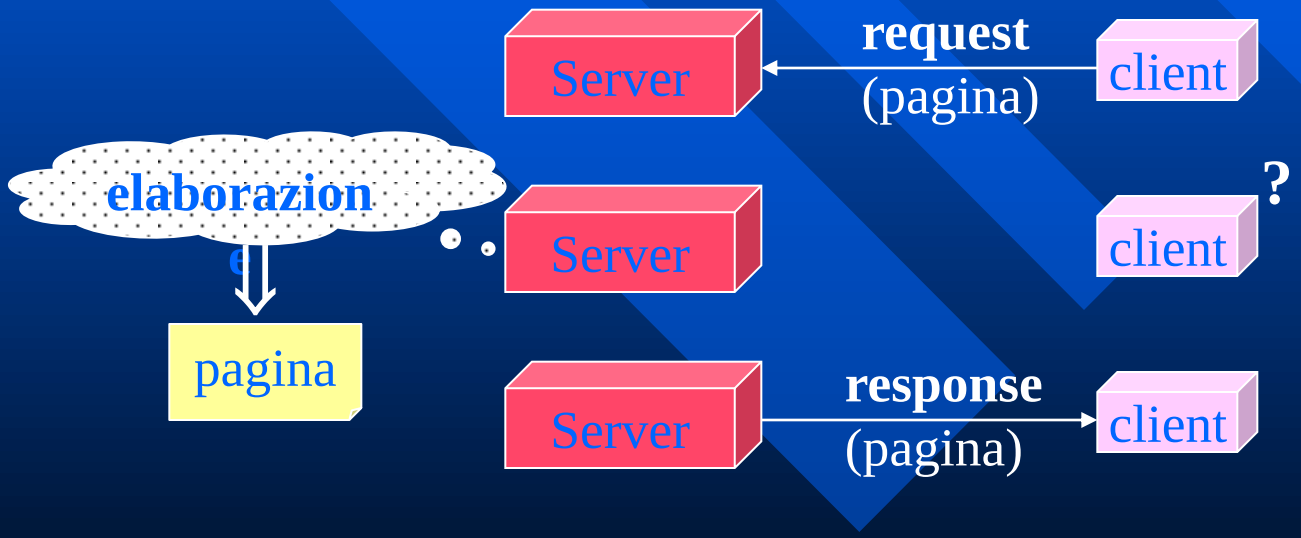
Comunicazione HTTP

- questo linguaggio si basa per l'interazione con il client sui concetti di Richiesta e Risposta:
- il client (browser) invia al server la Richiesta per una pagina PHP
- Il server delega all'interprete PHP l'esecuzione degli script;
- il risultato di tale esecuzione (elaborazione) consiste, in normale codice HTML che viene incluso nella pagina che viene inviata, come Risposta, al client. .

Come funziona il Web

Nel caso più semplice l'URL contiene l'indirizzo di una **pagina HTML** (per es. http://www.di.unito.it/~goy/dida.html): il contenuto è fisso, definito nel momento in cui la pagina HTML viene scritta

Pagine Web "dinamiche" = pagine il cui contenuto viene generato (selezionato, composto) al momento della richiesta



COME SCRIVERE CODICE PHP

Come sappiamo il codice PHP è interpretato dal Web Server, nel nostro caso Apache. Ma come fa il Web Server a capire cosa deve elaborare?
La prima cosa è sicuramente l'estensione del file php, infatti quando al Browser viene chiesto di eseguire una pagina con estensione .php la gira direttamente al Web Server, che ha poi il compito di capire cosa interpretare e cosa lasciare così com'è, e di restituire la pagina.

COME SCRIVERE CODICE PHP

L'engine (motore PHP del Web Server) capisce cosa processare tramite un tag particolare. Questo particolare "codice" può essere ritrovabile in 4 modi differenti *(alcuni debbono essere settati nel file di configurazione php.ini).*

- **<?php** ...codice PHP... **?>**
- **<?** ...codice PHP... **?>**
- **<script language="php">** ...codice PHP... **</script>**
- **<%** ...codice PHP... **%>**

COME SCRIVERE CODICE PHP

<?php ...codice PHP... **?>**
Questo è il metodo di default per inserire codice.
Tutto quello che si trova all'interno dei tag verrà processato dal motore php.
Esempio:

```
<html>
 <head>
  <title>...</title>
 </head>
 <body>
  <!-- codice HTML non interpretato -->
   <?php
     /* codice PHP interpretato */
   ?>
  <!-- codice HTML non interpretato -->
 </body>
</html>
```

COME SCRIVERE CODICE PHP

<?

...codice PHP…

?>

Questa è una sintassi molto simile a quella precedente, ma per poter essere usata c'è bisogno dell'abilitazione nel file **php.ini.**

Nella sezione "**Language Options**" occorre settare ad "**On**" il parametro **"short_open_tag**".

Sezione “Language Options” del file Php.ini

```
;;;;;;;;;;;;;;;;;;;
; Language Options ;
;;;;;;;;;;;;;;;;;;;

; Enable the PHP scripting language engine under Apache.
engine = On

; Allow the <? tag.  Otherwise, only <?php and <script> tags are recognized.
; NOTE: Using short tags should be avoided when developing applications or
; libraries that are meant for redistribution, or deployment on PHP
; servers which are not under your control, because short tags may not
; be supported on the target server. For portable, redistributable code,
; be sure not to use short tags.
short_open_tag = On

; Allow ASP-style <% %> tags.
asp_tags = Off
```

COME SCRIVERE CODICE PHP

<script language="php">

...codice PHP...

</script>

Questa sintassi è simile a quella di Javascript.

E' attiva di default.

COME SCRIVERE CODICE PHP

<%

...codice PHP...

%>

Questa è la sintassi utilizzata dalle ASP e deve essere attivata per poter essere utilizzata. La sua attivazione deve essere effettuata nel file **php.ini.**

Nella sezione "**Language Options**" occorre settare ad "**On**" il parametro "**asp_tags**".

Sezione “Language Options” del file Php.ini

```
;;;;;;;;;;;;;;;;;;;
; Language Options ;
;;;;;;;;;;;;;;;;;;;

; Enable the PHP scripting language engine under Apache.
engine = On

; Allow the <? tag.  Otherwise, only <?php and <script> tags are recognized.
; NOTE: Using short tags should be avoided when developing applications or
; libraries that are meant for redistribution, or deployment on PHP
; servers which are not under your control, because short tags may not
; be supported on the target server. For portable, redistributable code,
; be sure not to use short tags.
short_open_tag = On

; Allow ASP-style <% %> tags.
asp_tags = Off
```

COME SCRIVERE CODICE PHP

Al contrario di altri linguaggi, gli spazi bianchi e gli a-capo (vedi Javascript) non sono significativi per il PHP (eccetto all'interno di stringhe). E' possibile scrivere il codice usando spazi in più, lasciando linee vuote e usando le tabulazioni per avere un codice più chiaro e comprensibile. Quello che conta per separare i blocchi di sintassi è il punto e virgola, che va posto alla fine di ogni istruzione.

VARIABILI

Le variabili in PHP sono **loosely typed** (non tipizzate) non hanno un tipo, ma lo assumono automaticamente in base al valore ad esse assegnato.

La maggior parte dei linguaggi di programmazione (ad eccezione del Perl) non usa un carattere speciale per identificare il nome delle variabili; nel PHP questo carattere è il simbolo del dollaro (‘$’) da inserire all’inizio del nome.

echo() print()

Le istruzioni echo() e print() servono per produrre l'output visualizzato dal browser.

echo("stringa") ha l'effetto di visualizzare la stringa posta tra parentesi tonde (le quali sono opzionali).
Nella versione senza parentesi tonde è possibile utilizzare anche la sintassi:
echo "stringa1","stringa2",….,"stringaN";
Es:
echo "Ciao a tutti ","è una bellissima giornata";

echo() print()

La sintassi di **print()** è molto simile a quella di echo();

print (“stringa”);

A differenza di echo(), **print()** accetta un solo parametro in input e restituisce un valore che può essere 1 se la visualizzazione ha avuto successo, 0 altrimenti.

CONVERSIONE IMPLICITA DI TIPO

```
echo 234 + "145";
```

In questo caso PHP converte la stringa "145" in un numero intero, lo somma all'intero 234 *(converte il risultato in stringa)* e lo visualizza sulla pagina tramite la funzione echo().

n.b.: in Javascript il risultato sarebbe stato 234145

CONVERSIONE ESPLICITA DI TIPO

Per forzare il tipo di una variabile si debbono usare gli operatori di cast oppure la funzione settype().
Il cast si fa in questo modo:

```
$variabile = (string) 234;   // Il PHP assegnerà a $variabile il valore "234"
```

Usando la funzione settype() invece:

```
$variabile = 12;
settype($variabile, "double");   // La variabile adesso contiene un numero decimale di tipo double
```

MAIUSCOLE E minuscole

I nomi delle variabili sono case sensitive, mentre le parole chiave e i nomi delle funzioni NON lo sono!

```
print("Questa frase viene visualizzata");
PRINT(“ed anche questa viene visualizzata");
```

Lo stesso discorso non vale con i nomi delle variabili:

```
$miavariabile = 1;
$MiaVariabile = 2;

echo $miavariabile;
echo $MiaVariabile;
```

COSTANTI

In Php le costanti si creano con la funzione define():

```
<?php

define('VELOCITA_SUONO', 340);

echo 'La velocità del suono è di ' . VELOCITA_SUONO . ' metri al secondo';

?>
```

Per convenzione, in Php, le costanti si scrivono in caratteri maiuscoli.

COSTANTI BOOLEANE

Il PHP rende disponibili anche due costanti booleane: **TRUE** e **FALSE**.

TRUE equivale all'intero 1
FALSE equivale alla stringa vuota ("").

Attenzione! **TRUE** e **FALSE** sono due costanti, e per questo non sono case-sensitive: true, false, True e False funzionano lo stesso.

COMMENTI

Per delimitare i commenti nel PHP esistono tre sistemi

```
/* questo marcatore delimita un commento su più righe
      attenzione che questo tipo di commento non può essere
      annidato */
```

```
// un commento su un'unica riga
$temp = 234;      // può essere inserito anche sulla stessa riga di un'istruzione
```

COMMENTI

un altro modo di scrivere un commento su una sola riga

```
$temp = "Scrivi il tuo nome: ";      # anche questo può essere inserito
                                     # a fianco di un'istruzione
```

Vi chiederete probabilmente il perché di due modi diversi per indicare i commenti "in linea". La risposta è semplice: il PHP è un linguaggio Open Source e per questo si adatta alle esigenze di chi programma: la sintassi '//' è quella abituale per i programmatori C/C++, mentre la sintassi '#' è quella del Perl e degli script di shell.

ESPRESSIONI

Sono consentite sostanzialmente le stesse espressioni lecite in C/C++,Java:

- espressioni numeriche: somma, sottrazione, prodotto, divisione, modulo.
- espressioni condizionali con ? (operatore ternario)
  es.: *c=(a==b?0:1)*
- espressioni stringa: concatenazione con .
- espressioni di assegnamento con =

```
echo(18/4);
echo(18%2);
echo("paolino“.'paperino');
```

OPERATORI PER IL TESTO

Il PHP mette a disposizione, oltre ad una ricca libreria di funzioni per lavorare con le strighe di testo, un operatore per la concatenazione. Questo operatore è il punto '.'
Guardiamo l'esempio:

```
$saluto = "Buongiorno, ";
$cognome = “Pallino";
$nome = “Pinco";
$formula_saluto = $saluto . $nome . " " . $cognome . "<br>";
```

n.b.: in Javascript questa operazione si fa con il +

STRUTTURE DI CONTROLLO

PHP dispone delle usuali strutture di controllo:

- if, switch, for, while, do/while
- In un if, le condizioni booleane esprimibili sono le solite (==, !=, >, <, >=, <=) con gli operatori logici AND (&&), OR (||), NOT (!)
- Anche i cicli for, while, do/while funzionano nel solito modo

ARRAY

In PHP il metodo più veloce per generare un array é utilizzare la funzione **array()** la cui sintassi é la seguente:

```
$List=array("mele", "banane", "arance");
```

In questo esempio il primo elemento avrà come indice 0, il secondo 1 e il terzo 2.
E' possibile assegnare l'indice usando la funzione array() nel seguente modo:

```
$List=array(1=>"mele",2=>"banane",3=>"arance");
```

ARRAY

La funzione **count()** serve a contare il numero di elementi di un array.

```
count($List);
```

riferito all'ultimo esempio, da 3 come risultato.

Esiste un metodo alternativo per aggiungere un valore ad un array:

```
$List[] = "Marancuia";
```

Verrà aggiunto in coda con indice successivo all'ultimo elemento con indice numerico.

FUSIONE DI ARRAY

In PHP 4.0 esiste una funzione **array_merge()** che consente di aggiungere un array alla fine di un altro.

La sintassi della funzione array_merge() é la seguente:

$NuovoArray=**array_merge(**$PrimoArray,$SecondoArray**)**

ARRAY

Il metodo più semplice per accedere a tutti i valori di un array é usare un ciclo, insieme alla funzione **each()**.

$Line=**each(**List**)**

creerà un array di nome $Line, che avrà due indici particolari:

key	→	rappresenta la posizione corrente
value	→	rappresenta il contenuto corrente

n.b.: key e value vanno scritti in minuscolo

ARRAY

All'interno di un ciclo for, **each()** può essere ripetuta tante volte quanti sono gli elementi dell'array.

```
for ($n = 0; $n <count($List); $n++)
{
        $Line = each ($List);
        echo("La posizione è $Line[key] il contenuto è
        $Line[value].<br>");
}
```

n.b.: notare l'uso del punto . per concatenare le stringhe

ARRAY

Quanto visto in precedenza, poteva essere scritto anche:

```
for ($n = 0; $n <count($List); $n++)
{
echo("La posizione è $n il contenuto è $List[$n].<br>");
}
```

ORDINAMENTO DI ARRAY

Per ordinare i valori senza tener conto delle chiavi (indici), bisogna usare la funzione **sort().**

Per ordinare i valori(sempre senza tener conto delle chiavi) in ordine inverso bisogna usare **rsort().**

sort($Array**)**;

rsort($Array**)**;

es.: sort($List);

ARRAY ASSOCIATIVI

Gli indici degli elementi non debbono necessariamente essere sempre numerici: possono essere anche delle stringhe. In questo caso si parla di array associativi.
Ad esempio:

```
$AnimaliAfricani[“nome”] = “Giraffa”;
```

Con questa istruzione abbiamo definito un array di nome *$AnimaliAfricani*, creando un elemento il cui indice è ‘nome’ e il cui valore è ‘Giraffa’.

ARRAY ASSOCIATIVI

Indici numerici e associativi possono coesistere nello stesso array.

```
$TeamDiSviluppo[1] = “Mario”;
$TeamDiSviluppo[2] = “Carlo”;
$TeamDiSviluppo[“analista”] = “Ettore”;
```

In questo caso abbiamo creato un array con 3 elementi, di cui 2 con indici numerici, e uno con indice associativo. Se in seguito aggiungeremo un elemento usando le parentesi quadre vuote, il nuovo elemento prenderà l’indice 3.

ARRAY ASSOCIATIVI

Avremmo potuto creare lo stesso array usando l'istruzione di dichiarazione dell'array, come segue:

```
$TeamDiSviluppo = array(1=>"Mario","Carlo","analista"=>"Ettore");
```

E' da notare che quando abbiamo usato le chiavi associative le abbiamo indicate fra apici singoli (vanno bene anche doppi).

FUNZIONI PER ARRAY ASSOCIATIVI

Negli array associativi esiste il concetto di **puntatore all'elemento corrente**. Tale puntatore è indispensabile per spostarsi da un elemento al successivo senza conoscere l'indice associativo. Alcune funzioni utili per lavorare con gli array associativi sono:

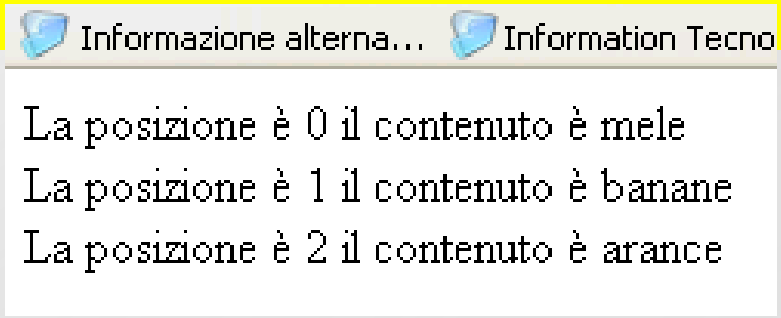
- **current(**NomeArray**)** restituisce il valore dell'elemento corrente
- **next(**NomeArray**)** sposta l'indice all'elemento successivo e ne restituisce il valore.
- **prev(**NomeArray**)** sposta l'indice all'elemento precedente e ne restituisce il valore.
- **key(**NomeArray**)** restituisce la chiave (indice) memorizzata al puntatore corrente

FUNZIONI PER ARRAY ASSOCIATIVI

Esempio:

```
$List=array("banane", "mele", "arance");

for ($n = 0; $n <count($List); $n++)
{
echo ("La posizione è ”.key($List).“ il contenuto è ". current($List));
echo (“<br>”);
next($List);
}
```



CONVERSIONE DA ARRAY A STRINGA

La funzione **implode()** converte un array in una stringa.

- Si potrebbe voler convertire un array in una stringa per poter aggiungere quel valore a un URL e passarlo ad un altra pagina.

- Si potrebbe voler convertire un array in una stringa per memorizzare quelle informazioni in un database.

CONVERSIONE DA ARRAY A STRINGA

Per passare da un array a una stringa si deve semplicamente definire il separatore.

```
$String = implode(“ “, $List);
echo $String;
```

riferito all’esempio precedente, restituirà:

```
mele banane arance
```

n.b.: nell’esempio il separatore è uno spazio.

CONVERSIONE DA STRINGA AD ARRAY

La funzione **explode()** converte una stringa in un array.

- Si potrebbe volere converitre una stringa in un array per convertire un campo di testo delimitato da virgole (per esempio un'area di ricerca di parole chiave in un form) nelle sue componenti separate.

CONVERSIONE DA STRINGA AD ARRAY

Per passare da una stringa ad un array si deve semplicamente informare la funzione **explode()** di quale sia il separatore esistente nella stringa.

$Array = **explode(**separatore, $String**)**;

$Array = **explode(**“ “, $String**)**;
echo $Array[1]; echo $Array[2];…….

separatore indica qualunque carattere che definisce la fine di un elemento all’interno di una stringa e l'inizio di un altro; comunemente si tratta di una virgola o di uno spazio. Nell’esempio il separatore è uno spazio.

DEFINIZIONE DI FUNZIONI

Le funzioni sono introdotte dalla parola chiave *function*

- Possono essere sia procedure (assenza di valore di ritorno), sia funzioni in senso proprio (non esiste la parola chiave **void**)
- I parametri formali sono *senza dichiarazione di tipo*
- Al solito, il corpo va racchiuso in un blocco

```
function sum($a,$b) { return $a+$b;}   // funzione (ha il return)

function printSum($a,$b) { echo($a+$b); }  //procedura
```

CHIAMATA DI FUNZIONE

- Le *funzioni* sono chiamate nel solito modo, fornendo la lista dei parametri attuali

- Esempio:

```
echo ("Somma: “. sum(18,-3). "<br>" );
printSum(19, 34.33);
```

CHIAMATA DI FUNZIONE

- La lista dei parametri attuali *(chiamante)* può non corrispondere in numero ai parametri formali previsti.

- Se i parametri attuali *(chiamante)* sono più di quelli necessari, quelli extra vengono ignorati.

- Se i parametri attuali *(chiamante)* sono meno di quelli necessari, quelli mancanti vengono trattati come variabili non inizializzate.

SCOPE RULES

In PHP il raggio di validità di una variabile dipende dal suo contesto.
Se la variabile viene definita all'interno di una funzione, essa sarà visibile solamente all'interno di quella funzione e viene chiamata **locale**.
Esempio:

```
$var_esempio = "Qualche parola";
function prova()
{
      $var_esempio = 4523;
}
echo $var_esempio;
//La funzione echo visualizzerà la stringa "Qualche parola".
```

GLOBAL SCOPE

Allo stesso modo, le variabili usate nello script principale non vengono viste all'interno delle funzioni.
Per far riferimento a una variabile *globale* all'interno di una funzione occorre prima (all'interno della stessa) dichiararla **global.**

```
$var_esempio = "Del testo a caso";
function scrivi_esempio()
{
      global $var_esempio;      // La funzione "cattura" il valore di $var_esempio
      echo $var_esempio;        // e lo visualizza nella pagina
}
scrivi_esempio();
```

n.b.: scrivere **global $var_esempio = ……..** è sbagliato

USO DI RETURN

Le funzioni hanno spesso bisogno di ritornare dei valori al programma. In PHP per compiere questa operazione si utilizza la parola chiave 'return'.

```
function somma_uno ($x)
{
   return $x + 1;
}
$y = 7;
$z = somma_uno ($y);   // $z adesso vale 8
```

CHIAMATA PER VALORE

Normalmente il PHP passa alla funzione una copia di ogni valore passato come argomento; quando la funzione ritorna il risultato le copie vengono eliminate; questo significa che la funzione non cambia il valore della variabile.
Questo modo di operare si chiama passaggio "by value" ovvero per valore, poiché alla funzione viene passato solo il valore dell'argomento. Per default le funzioni PHP operano in questo modo.

CHIAMATA PER RIFERIMENTO

In alcune circostanze può essere utile operare direttamente sulle variabili invece che su una copia del loro valore.
Passare alla funzione la variabile stessa invece che una copia del suo valore si chiama passaggio "by reference" ovvero per riferimento. In realtà alla funzione non viene passata la variabile, ma un puntatore ad essa.
Per dire alla funzione che deve lavorare "by reference" e non "by value" si usa l'operatore '&'

CHIAMATA PER RIFERIMENTO

```
function raddoppia(&$x)     // tipo procedura
 {
        $x = $x * 2;
        // non c'è il return
}

$x = 5;
echo $x;
raddoppia ($x);
echo $x;                    // stampa "10" sulla pagina
```

PHP & HTML: LE VARIBILI GET E POST

Una delle principali possibilità offerte dai linguaggi di scripting lato server è quella di generare contenuti (dinamicamente) sulla base delle richieste degli utenti.

Questa interattività si realizza soprattutto attraverso le variabili GET e POST che consentono, appunto, agli utenti di passare al server le loro richieste o preferenze attraverso i form (i classici moduli html) o querystring.

Esistono, infatti, due diversi modi per passare dati al server: il metodo POST (generalmente usato nei form) ed il metodo GET (generalmene usato nelle querystring).

METODO GET

Con il **metodo GET** i dati vengono passati direttamente all'interno dell'indirizzo web della pagina, il quale si presenterà accompagnato da un punto di domanda (**?**) seguito dai dati organizzati in coppie nome/valore (qualora vi siano diverse coppie queste saranno legate tra loro dal simbolo **&**):

http://localhost/automobili.php?marca=fiat&modello=panda

METODO GET

http://localhost/automobili.php?marca=fiat&modello=panda

Nell'esempio qui sopra abbiamo ipotizzato di avere una pagina dinamica chiamata “automobili.php" in grado di visualizzare informazioni relativamente a diverse autovetture sulla base di due parametri (marca e modello).
Il contenuto della pagina “automobili.php" cambierà, ovviamente, ogni volta che varieranno i valori dei parametri marca e modello.

FORM HTML (pagina lato client)

```
<FORM ACTION="http://localhost/automobili.php" METHOD="get">

    <INPUT TYPE="text"  NAME="marca"  VALUE=" ">
    <INPUT TYPE="text" NAME="modello" VALUE=" ">
    <INPUT TYPE="submit" VALUE="Invia Dati" NAME="bottone">

</FORM>
```

n.b.: occorre ricordare che la pagina lato client è .html, invece la pagina lato server è .php e deve essere dentro una directory conosciuta dal Web Server

METODO GET (pagina lato Server)

PHP memorizza i dati passati attraverso la querystring all'interno della variabile $_GET. Nello specifico del nostro esempio per recuperare i valori all'interno della pagina “automobili.php” faremo così:

```
<?
//Recupero il valore del parametro “marca"
$marca_auto = $_GET[“marca”];
//Recupero il valore del parametro “modello"
$modello_auto = $_GET[“modello”];
//Ora stampo semplicemente a video il risultato
echo "Hai scelto una " . $marca_auto . " modello " . $modello_auto;
?>
```

Per recuperare i dati passati dalla querystring occorre usare $_GET accompagnato da parentesi quadre al cui interno bisogna scrivere (tra gli apici) il nome del parametro da recuperare

METODO GET (pagina lato Server)

Il metodo GET presenta alcuni indubbi svantaggi tra i quali:

- Non è adatto per i login, poiché sia il nome utente sia la password risulterebbero completamente visibili a video e memorizzabili dal browser del client come pagina visitata;

- Ogni invio con il metodo GET viene registrato nel file di registro del Web server, compresi i valori dei parametri passati nella querystring;

- La lunghezza dell'URL è limitata poiché tale URL viene assegnato a una variabile del server;

METODO POST (pagina lato Server)

Diversamente dal metodo GET, il metodo POST invia i dati in maniera da non essere direttamente visibili per l'utente, attraverso la richiesta HTTP che il browser invia al server. Tornando all'esempio precedente, per recuperare il valore del campo “nome" all'interno della nostra applicazione PHP useremo la variabile $_POST

```
<?
//Recupero il valore del parametro “nome"
$nome_utente = $_POST[“nome”];
//Ora stampo semplicemente a video il risultato
echo "Ciao " . $nome_utente;
?>
```

RICORDA

$_GET & **$_POST** sono in realtà degli array detti **superglobali** in quanto la loro visibilità si estende a tutte le parti del codice compreso l'interno delle funzioni.

GET & **POST** nel form html possono anche essere scritti in minuscolo, ma in php **$_GET** & **$_POST** vanno scritti obbligatoriamente in MAIUSCOLO.

COOKIE

I Cookie sono un metodo rapido per memorizzare, sul computer degli utenti, delle informazioni che si vuole persistano anche nelle successive visite al nostro sito.

I Cookie sono molto utili per memorizzare piccoli dati come ad esempio il nome dell'utente o una serie di preferenze. I cookie non sono adatti per informazioni critiche come password o dati personali in quanto potrebbero crearsi dei problemi di sicurezza.

COOKIE

Per impostare un cookie si usa **setcookie()**:

```
setcookie("nome_utente", "pippo", time()+3600);  // 1 ora
```

All'interno del comando setcookie() ci sono di solito almeno 3 parametri:

- il primo specifica il nome identificativo del cookie;

- il secondo specifica il valore del cookie;

- il terzo imposta la scadenza del cookie. Se non viene impostata una data di scadenza il coookie non scadrà;

COOKIE

Per esempio vediamo come memorizzare il nome di un nostro utente (richiesto tramite un form) all'interno di un cookie.

```
<?
// recupero il nome dalla querystring
$nome = $_GET['nome'];

// memorizzo il nome in un cookie
// ed imposto la scadenza tra un'ora...
setcookie("nome_utente", $nome, time()+3600);
?>
```

POSSIBILI PROBLEMI CON I COOKIE

La chiamata di setcookie() restituisce TRUE nel caso in cui vada a buon fine, oppure FALSE qualora si verifichino degli errori...ed eccone uno frequente

"Warning: Cannot add header information - headers already sent by (output started at …… on line…..

Il cookie un non può essere inviato dopo che verso il browser sia già stato emesso un qualsiasi output.
Non si può quindi inviare un cookie dopo aver usato un **echo** e neppure dopo aver usato un qualunque tag html, anche se si tratta di tag di apertura della pagina.

POSSIBILI PROBLEMI CON I COOKIE

Si può risolvere parzialmente la cosa attraverso il buffering dell'output (**ob_start**()); le operazioni di output **verranno posposte** e inviate soltanto al termine dell'esecuzione.

```
<?php
ob_start();
echo ("Questo normalmente non si potrebbe fare!");
setcookie ("miocookie", "dati");
?>
```

Viene così risolto il problema degli echo prima del setcookie, ma non quello dei tag html, che dovranno essere inviati in maniera dinamica dentro gli echo o print.

RICHIAMARE UN COOKIE

Ora che abbiamo memorizzato nel cookie il nome dell'utente potremo tranquillamente richiamarlo in tutte le nostre pagine PHP in questo modo:

```
<?
//recupero il valore del cookie...
$nome = $_COOKIE['nome_utente'];

//stampo a video il nome...
echo $nome;
?>
```

CAMBIARE IL VALORE AD UN COOKIE

Se vogliamo cambiare il valore del nostro cookie basterà ripetere semplicemente l'operazione di assegnazione:

```
<?
//imposto come valore "pippo"
setcookie("nome_utente", "pippo");

//ho cambiato idea e imposto come "pluto"
setcookie("nome_utente", "pluto");
?>
```

CANCELLARE UN COOKIE

Se invece vogliamo cancellare il cookie basterà richiamare il cookie senza specificare nessun valore:

```
setcookie("nome_utente");
```

oppure possiamo reimpostare la scadenza ad un momento passato:

```
setcookie("nome_utente", "pippo", time()-9999);
```

In entrambi i casi il cookie verrà cancellato.

SESSIONI

Un altro sistema di memorizzazione di informazioni è l'utilizzo delle sessioni.
A differenza dei cookie le sesioni non scrivono nulla sul computer dell' utente, ma operano (quasi esclusivamente) sul server creando degli specifici files dove vengono salvati alcuni dati importanti relativamente alla sessione di navigazione dell'utente. Una volta che la sessione sarà terminata il file con i dati della sessione verrà eliminato. Una sessione termina nel momento in cui l'utente chiude il browser.

SESSIONI

Le sessioni vengono utilizzate ad esempio nella gestione delle autenticazioni (login) degli utenti che, una volta loggati, verranno identificati come tali da tutte le pagine (.php) del sito.

La prima cosa da fare se vogliamo lavorare con le sessioni è impostare nel file di configurazione del PHP (php.ini) la direttiva session.save_path, indicando la directory nella quale verranno salvate le informazioni sulle sessioni dei nostri utenti.

SESSIONI

La funzione da utilizzare all'interno delle pagine .php per aprire una sessione è **session_start()**.
Questa funzione non prevede parametri.
La funzione **session_start()** deve essere necessariamente utilizzata prima dell'invio di codice html: nella parte precedente del file .php non deve pertanto essere già stato scritto ed inviato del codice html il quale comprometterebbe il buon esito della funzione (come per i cookie).

APERTURA SESSIONE

Poniamo di voler aprire una sessione dove salvare username e password del nostro utente (dati che ci sono stati forniti tramite un form di login):

```
<?
//Recupero username e password dal form
$username = $_POST['user'];
$password = $_POST['pass'];
// Apriamo la sessione
session_start();
// Salviamo i dati
$_SESSION['username'] = $username;
$_SESSION['password'] = $password;
?>
```

A questo punto abbiamo salvato all'interno della nostra sessione (grazie a $_SESSION) due diverse variabili: username e password.

RECUPERARE DATI DA UNA SESSIONE

```
<?
//Apriamo la sessione
session_start();

//Recuperiamo i dati
$username = $_SESSION['username'];
$password = $_SESSION['password'];

//visualizziamo i dati
echo "Ciao " . $username . " la tua password è " . $password;
?>
```

ELIMINARE VARIABILI DI SESSIONE

```
//Per eliminare una specifica variabile di sessione:
unset($_SESSION['username']);

//Per eliminare tutte le variabili di sessione:
$_SESSION = array();
```

RICORDA

Come per i cookie l'istruzione **session_start()** non deve essere preceduta da nessun tag html, tantomeno da istruzioni echo o print.

Il delimitatore "**<?**" del PHP non deve essere preceduto da spazi bianchi.

La variabile $_SESSION("….") rappresenta una variabile superglobale

PHP & MYSQL

PHP è in grado di connettersi a diversi database (MySql, MS Access, PostgreSql, Oracle, Microsoft Sql Server, Sybase,...) tuttavia ci limiteremo a vedere l'interazione con MySql che è senza dubbio la soluzione più comune e diffusa.

PHP & MYSQL

MySql è un database veloce e potentissimo in grado di gestire applicazioni con un elvato grado di criticità e, cosa non secondaria, è un software open source, liberamente scaricabile dal sito www.mysql.com.

CONNESSIONE AL SERVER MYSQL

PHP mette a disposizione dello sviluppatore diverse funzioni per interagire con i db MySql.
Per prima cosa vediamo come fa PHP a connetersi al MySql Server.
Allo scopo soccorre la funzione mysql_connect() che si utilizza con la seguente sintassi:

mysql_connect(URLserver, utente, password, numPorta**)**

$myconn = **mysql_connect(**“localhost”,”root”,“password”,”3306”**) or die(“**Errore...”**)**;

SELEZIONARE IL DATABASE

Una volta stabilita la connessione è necessario selezionare uno specifico db sul quale lavorare. A questo scopo PHP ci fornisce la funzione mysql_select_db() da utilizzarsi con la seguente sintasi:

mysql_select_db(database, $connessione**)**

Ad esempio:

```
mysql_select_db(“mio_database”, $myconn) or die(“Errore...”);
```

INTERROGARE IL DATABASE

Vediamo come è possibile recuperare dei dati presenti in un database precedentemente creato.
Per fare questo dobbiamo formulare ed eseguire una query.
Una volta formulata la query useremo la funzione mysql_query() con la seguente sintassi:

mysql_query($query, $connessione**)**

```
$query = "SELECT * FROM tabella";
$result = mysql_query($query, $myconn) or die("Errore...");
```

IL RECORDSET

```
$query = "SELECT * FROM tabella";
$result = mysql_query($query, $myconn) or die("Errore...");
```

La variabile $result contiene una struttura dati che rappresenta la tabella con le *t*-uple risultanti dalla esecuzione della *SELECT* sul database.
Tale struttura prende il nome di **Recordset**.

Nome	**Cognome**	**Telefono**
Pinco	Pallino	091 76545665
Cisco	Ciasco	091 65547890

Vedremo nelle slide successive come prelevare le singole righe da questa struttura.

ESEMPIO COMPLETO

Vediamo ora **un esempio completo**.
Poniamo di voler recuperare dalla tabella "amici" una serie di dati (nome, cognome e telefono) e di volerli stampare a video per ogni occorrenza trovata nel database.

```
<?
//Mi connetto al MySql Server
$myconn = mysql_connect(“localhost”, “root”, “password”,”3306”) or
die("Errore Connessione”);

//Seleziono il database degli amici
mysql_select_db(“database”, $myconn) or die("Errore Selezione");

//Imposto ed eseguo la query
$query = "SELECT nome,cognome,telefono FROM amici";
$result = mysql_query($query, $myconn) or die("Errore Query");
```

ESEMPIO COMPLETO

```
//Conto il numero di occorrenze trovate nel db
$numrows = mysql_num_rows($result);
//Se il database è vuoto lo comunico
if ($numrows == 0)
        {
        echo "Database vuoto!";
        }
//Se invece trovo delle occorrenze...
else
        {
                //Ciclo for che si ripete per il numero di occorrenze trovate
        for($x=0; $x<$numrows; $x++)
                {
                //Recupero il contenuto di ogni record rovato
                $resrow = mysql_fetch_row($result);
                $nome = $resrow[0];
                $cognome = $resrow[1];
                $telefono = $resrow[2];
```

ESEMPIO COMPLETO

```
                    //Stampo il risultato
                    echo "nome: <b>" . $nome . "</b> <br>";
                    echo "cognome: <b>" . $cognome . "</b> <br>";
                    echo "telefono: <b>" . $telefono . "</b> <br>";

                    }
            }
?>
```

PRELEVARE I DATI DAL RECORDSET

- **mysql_num_rows()**

```
$numrows = mysql_num_rows($result);
```

Serve per contare il numero di records trovati all'interno del db sulla base di una data query.

- **mysql_fetch_row()**

```
$resrow = mysql_fetch_row($result);
```

Un metodo per recuperare le singole righe dal **recordset.**

PRELEVARE I DATI DAL RECORDSET

Un altro modo per estrarre le righe dal **recordset** è:

Mysql_fetch_object()

```
while ($data=mysql_fetch_object($result))
{
echo "<br>";
echo “Nome: <b>".$data->nome."</b> <br>";
echo “Cognome: <b>".$data->cognome."</b> <br>";
echo “Telefono: <b>".$data->telefono."</b> <br>";
}
```

In questa maniera ci riferiamo direttamente ai nomi di campo, invece che all’indice dell’array.

INSERT, UPDATE E DELETE

Vediamo come eseguire altre importanti operazioni con i database. Con **Insert** si inseriscono nuovi dati nel db, con **Update** si aggiornano dei dati già presenti, con **Delete** si cancellano dei dati.

Dal punto di vista di PHP queste operazioni non differiscono tra loro, l'unica cosa che cambia è la query che viene eseguita.

```
$query = "INSERT INTO tabella (campo1,Campo2,Campo3)
VALUES('valore1','valore2','valore3')";
```

```
$query = "UPDATE tabella SET campo1='valore1', campo2='valore2',
campo3='valore3' WHERE id = 1";
```

```
$query = "DELETE FROM tabella WHERE id = 1";
```

n.b.: Se nelle query di Update e Delete non viene usata la condizione “where“ verranno aggiornati/eliminati tutti i record del db!

INSERT, UPDATE E DELETE

```
<?
//Mi connetto al MySql Server
$myconn = mysql_connect('localhost',‘root', ‘password‘,’3306’) or
die("Errore Connessione”);

//Seleziono il database degli amici
mysql_select_db('database', $myconn) or die("Errore Selezione");

//Imposto ed eseguo la query
$query = "DELETE FROM tabella WHERE nome = “Pinco”;
$result = mysql_query($query, $myconn) or die("Errore Query");
?>
```

LINK UTILI

- http://it.php.net/manual/it/index.php
- http://freephp.html.it/guide
- http://www.mrwebmaster.it/fareweb/php/guida_php