Guida introduttiva a PHP

Piccola premessa

Questa piccola guida non pretende di essere sostitutiva ad altre sicuramente migliori e più complete, ma solamente una introduzione per chi si avvicina a PHP.

Spero che quanto leggerete in queste pagine vi sia di aiuto e soprattutto vi renda comprensibili le guide ed i manuali che andrete a leggervi successivamente.

Ma ora passiamo subito al dunque e leggete le prime lezioni.

Lezione 1 - Piccole regole di sintassi

Una pagina che contiene codice scritto in PHP deve essere salvata con estensione php, questo per permettere all'interprete di sapere che deve lavorarci!

Altra necessità di questo linguaggio è l'uso dei tag che indicano l'inizio e la fine della parte scritta in PHP, praticamente prima di iniziare a inserire delle istruzioni PHP bisogna scrivere **<?php** e dopo l'ultima **?>** ; questo permette anche di alternare parti scritte in html e parti in php nella stessa pagina.

Ultima nota, ma non meno importante, ogni riga (a parte quelle dei tag di inizio e fine!) deve terminare con **;** .

A questo punto vediamo come fare l'output a video, la funzione è semplicissima potete usare **echo** *"Stringa da visualizzare"***;** o la **print("***Stringa da visualizzare***");**

Stringa da visualizzare: può essere qualsiasi cosa, anche codice html, considerate che l'output che PHP manderà al browser sarà codice html, quindi se dopo la stringa vogliamo inserire un ritorno a capo possiamo inserire il tag **echo** "*stringa da visualizzare*
";.

La parte più interessante e utile è l'utilizzo delle variabili, sono queste che ci permetteranno di dare vita al sito interagendo con i visitatori. Le variabili in php sono identificate dal simbolo $ prima del nome e possiamo chiamarle come vogliamo: $pippo, $a, ecc..

Lezione 2 - Le variabili

Come anticipato nell'articolo precedente, le variabili si identificano dal simbolo $ prima del nome.
Cosa sono le variabili? Beh, una variabile si può assimilare ad un contenitore nel quale possiamo metterci dei valori o delle stringhe da utilizzare per operazioni successive.
Per assegnare ad una variabile un valore si utilizza = (mentre per effettuare dei confronti si userà ==), ecco un piccolo esempio: $a = 5; (assegna il valore 5 alla variabile $a.
Ecco un piccolo esempio di utilizzo delle variabili, molto semplice, mostrerà i valori delle variabili e la somma:

```
<?php
$a = 5; //assegna il valore 5 ad a
$b = 7; //assegna il valore 7 a b
$c = $a + $b; //assegna la somma di a e b alla variabile c
echo "Il valore di a è $a<BR>";
echo "Il valore di b è $b<BR>";
echo "La somma di a + b è $c";
?>
```

Ovviamente alle variabili possiamo anche assegnare delle stringhe e per concatenarle utilizziamo il . come in questo piccolo esempio che scriverà: Mi chiamo Pippo:

```
<?php
$a = "Mi chiamo";
$b = "Pippo";
echo $a." ".$b."$a<BR>";
?>
```

Come potete notare ho concatenato le due stringhe inserendo anche uno spazio (che è anch'esso una stringa).
Usare le variabili in questo modo però non permette di avere dei valori inseriti da un utente e quindi serve solo per valori/stringhe che sono fissi. Per ricevere i dati inseriti da un utente dobbiamo necessariamente avere una pagina con un form per l'inserimento dei valori e la pagina che utilizza i dati immessi, vediamo un piccolo esempio:

La pagina con il form, pagina1.html

```
<form action="pagina2.php" method="post"> //come metodo di invio dei dati
possiamo anche usare get
Valore 1: <input type="text" name="valore1">
Valore 2: <input type="text" name="valore2">
<input type="submit" name="invia" value="Invia">
```

La pagina che somma i dati, pagina2.php

```
<?php
$a = $_POST["valore1"]; //se come metodo usate get, qui dovrete usare
$_GET["valore1"]
$b = $_POST["valore2"]; //se come metodo usate get, qui dovrete usare
$_GET["valore2"]
$c = $a + $b; //assegna la somma di a e b alla variabile c
echo "Il valore di a è $a<BR>";
echo "Il valore di b è $b<BR>";
echo "La somma di a + b è $c";
?>
```

Ora siete in grado di usare delle variabili semplici per manipolare i dati inseriti dagli utenti.

Lezione 3 - Gli operatori

In questo articolo non troverete commenti, ci sarà solo l'elenco degli operatori di base.

Aritmetici

+	Somma
-	Sottrazione
*	Moltiplicazione
/	Divisione
%	Resto di una divisione

Comparazione

==	Uguali
===	Identici (uguali e dello stesso tipo, solo PHP 4)
!= o <>	Diverso
!==	Non identici
<	Minore
>	Maggiore
<=	Minore o uguale
>=	Maggiore o uguale

Incremento/Decremento

++$a	Incrementa $a di 1 e poi restituisce il valore di $a
$a++	Restituisce il valore di $a e poi incrementa di 1
--$a	Diminuisce $a di 1 e poi restituisce il valore di $a
$a--	Restituisce il valore di $a e poi diminuisce di 1

Logici

and o &&	Restituisce vero se entrambi i valori sono veri
or o ||	Restituisce vero se almeno uno dei valori è vero
xor	Restituisce vero solo se uno dei due valori è vero
!	Negazione

Stringhe

.	Per concatenare 2 stringhe

Ecco fatto, questi sono gli operatori di base per lavorare con le variabili, esistono anche delle varianti:

"Variante"	**Uguale a**
$a += $b	$a = $a + $b
$a -= $b	$a = $a - $b
$a .= $b	$a = $a.$b

Lezione 4 - I Cicli

Cosa sono i cicli? O meglio, cosa si intende per ciclo? Diamo una definizione un po' grossolana ma che riesca a far capire semplicemente di cosa si tratta.

Per ciclo intendo il ripetere N volte uno stesso gruppo di istruzioni.
Ci sono due metodi per effettuare dei cicli e si possono suddividere in due casi particolari:
1. il primo caso è quando sappiamo quante volte dovranno essere ripetute le istruzioni;
2. il secondo caso è quando non lo sappiamo.

Tutto semplice fin'ora, no? Vediamo quali sono le istruzioni e la loro sintassi:
Per il primo caso, ovvero quando sappiamo il numero di volte che le istruzioni dovranno essere eseguite, utilizziamo il ciclo For.

```
for(valore_inizio;criterio_fine_ciclo;incremento_valore_inizio) {
      /* istruzioni da eseguire */
}
```

valore_inizio è una variabile a cui viene assegnato il valore iniziale per contare quante volte viene eseguito il ciclo;
criterio_fine_ciclo è il controllo che verifica il N° di volte che è stato eseguito il ciclo;
incremento_valore_inizio è l'aumento (o diminuzione, a seconda dei casi) della variabile che conta il n° di volte che il ciclo viene effettuato.
Vediamo subito un esempio pratico, in modo da rendere ancora più chiaro tutto!

```
<?php
for ($i=0;$i<=10;$i++){
      echo $i."<BR>";
}
?>
<?php
for ($i=10;$i>=0;$i--){
      echo $i."<BR>";
}
?>
```

Allora questi due cicli nell'esempio scrivono semplicemente il valore di i, che nel primo caso parte da 0 e arriva a 10 (nel criterio abbiamo posto che fino a quando i è minore o uguale a 10 ripete le istruzioni), mentre nel secondo parte da 10 e ritorna a 0!
Come vedete sappiamo già quante volte il ciclo deve essere eseguito, nell'esempio abbiamo messo 10, ma si può utilizzare una variabile risultato di un altro calcolo (quando useremo mysql sarà il numero di righe), come criterio di termine del ciclo.
Per il secondo caso, ovvero quando non sappiamo quante volte ripetere le istruzioni, si utilizza il ciclo while (oppure do - while!), ed ecco la sintassi:

```
while (criterio_fine_ciclo) {
      /* istruzioni da eseguire */
}
            O, in alternativa:

do {
      /* istruzioni da eseguire */
}while (criterio_fine_ciclo);
```

La differenza nell'uso del primo o del secondo metodo è semplice ma importante: nel primo ciclo se il criterio non è soddisfatto le istruzioni del blocco non vengono eseguite, mentre nel secondo indipendentemente dal criterio le istruzioni all'interno del ciclo vengono eseguite almeno una volta.
Un altro esempio per chiarire meglio:

```
<?php
$i=true;
$t=0;
while ($i){
      $t++;
      echo $t."<BR>";
      if ($t>4) { $i=false; }
}
?>
```

Così il ciclo viene eseguito 5 volte e viene scritto il valore di t da 1 a 5, dopodiché il ciclo termina (ma non è questo il caso in cui si usa!).
˝
Se all'esterno del ciclo (prima di while) impostiamo $i=false il ciclo non viene eseguito, in quanto il criterio di while non è soddisfatto (infatti while ($i) si può tradurre in: finché $i è vero).

```
<?php
$i=true;
$t=0;
do{
      $t++;
      echo $t."<BR>";
      if ($t>4) { $i=false; }
}while ($i);
?>
```

In questo caso otteniamo lo stesso risultato dell'esempio precedente, ma se impostiamo $i=false; prima del do, vedremo che la prima volta viene scritto il valore di t, poi il ciclo si ferma in quanto il criterio di while non è soddisfatto.
Negli esempi del while ho usato un numero preciso che ho stabilito e come abbiamo detto while si utilizza quando non si sa quando si è alla fine. L'esempio più pratico del concetto è la lettura di un file, dobbiamo continuare a leggere fino a che non siamo alla fine del file ed ovviamente non sappiamo (numericamente) quante volte dobbiamo leggere; ecco l'esempio:

```
<?php
while !(feof($fp)) {
      /*leggi dal file*/
      /*scrivi a video quello che hai letto*/
}
?>
```

Che si traduce in: fino a che non sei alla fine del file leggi dal file e scrivi a video quello che hai letto; ora lo spiego meglio: feof($fp) è una funzione che è vera quando è alla fine del file e falsa quando è in qualsiasi altro punto, per cui con ! davanti diventerà vera (e quindi esegue il ciclo) quando non è alla fine del file.

Lezione 5 - Le "scelte"

A questo punto delle lezioni siamo in grado di scrivere delle pagine in PHP con le istruzioni che vengono eseguite una dietro l'altra, in questa lezione vedremo come "scegliere" le istruzioni da eseguire in base a delle condizioni.
Sì, lo soche non ho usato il termine adatto intitolando il tutto come «Le "scelte"», ma credo che renda meglio l'idea di quello che sto per spiegare.
Vediamo subito il primo metodo:

```
if (condizione) {
      /* istruzioni da eseguire 1 */
}
else {
      /* istruzioni da eseguire 2 */
}
```

(Iniziamo ad usare dei termini più corretti, ok?) Questo salto condizionale si può tradurre in italiano in questa semplice frase (indico tra partentesi le parti del codice quì sopra):
(if) Se è vera la condizione (condizione) fai /* istruzioni da eseguire 1 */
(else) altrimenti fai /* istruzioni da eseguire 2 */
Se è necessario possiamo mettere molti if uno dentro l'altro per operare più distinzioni, mi spiego meglio, è possibile mettere un'altra scelta con if nella parte /* istruzioni da eseguire 1 */ e /* istruzioni da eseguire 2 */, e così via, ottenndo una serie di if annidati.
(condizione) può essere un semplice confronto di una variabile con un valore fisso, un confronto di due variabili, oppure una condizione multipla come ad esempio:

```
if ($a == 1) {
      echo "La variabile a è uguale a 1";
}
else {
      echo "La variabile a non è uguale a 1";
}
```

```
if (($a == 1) and ($b == 7)) {
      echo "La variabile a è uguale a 1 e la variabile b è uguale a 7";
}
else {
      echo "La variabile a non è uguale a 1 o la variabile b non è uguale a 7";
}
```

Questi esempi sono molto semplici e, come potete notare, ho utilizzato l'operatore logico «and» per indicare che la condizione da verificare è su due variabili e devono essere entrambe vere.
Questo tipo di scelta va bene se i valori delle variabili da verificare sono pochi, nel caso in cui una variabile può assumere molti valori differenti non è ammissibile scrivere nel codice della pagina una lunga serie di if, in questo caso si utilizza la seguente funzione:

```
switch (variabile) {
      case 1:
            /* istruzioni da eseguire */
            break;
      case 2:
            /* istruzioni da eseguire */
            break;
      case 3:
            /* istruzioni da eseguire */
```

```
            break;
      case 4:
            /* istruzioni da eseguire */
            break;
      case 5:
            /* istruzioni da eseguire */
            break;
}
```

Questa funzione confronta il valore della variabile con tutti quelli riportati in ogni case ed esegue le istruzioni corrispondenti, l'istruzione «break;» serve a indicare che le istruzioni del case in cui si è entrati sono terminate e non deve eseguire anche quelle del case successivo.
Vediamo subito un esempio chiarificatore:

```
switch ($a) {
      case 1:
            echo "La variabile è uguale a 1";
            break;
      case 2:
            echo "La variabile è uguale a 2";
            break;
      case 3:
            echo "La variabile è uguale a 3";
            break;
      case 4:
            echo "La variabile è uguale a 4";
            break;
      case 5:
            echo "La variabile è uguale a 5";
            break;
}
```

Questo piccolo codice ci scriverà a quale valore è uguale la nostra variabile, se prima dello switch assegniamo a $a il valore 1 ed eliminiamo tutti i break; verrà visualizzato «La variabile è uguale a 1La variabile è uguale a 2La variabile è uguale a 3La variabile è uguale a 4La variabile è uguale a 5».
Una nota doverosa, prima di chiudere questa lezione, in questi esempi io ho utilizzato dei numeri come condizione da verificare con le variabili, tenete presente che è possibile farlo anche con il testo, come in quest ultimo esempio (che non è il massimo per un login, ma quando utilizzeremo un database sarà differente!):

```
switch ($utente) {
      case 'pippo':
            /* pagina di pippo */
            break;
      case 'pluto':
            /* pagina di pluto */
            break;
      case 'paperino':
            /* pagina di paperino */
            break;

      case 'gastone':
            /* pagina di gastone */
            break;
      case 'ciccio':
            /* pagina di ciccio */
            break;
}
```

Lezione 6 - I Form

Finalmente un po' di interattività nelle nostre pagine, con i form è possibile ricevere dei dati dagli utenti del nostro sito ed elaborarli.

Per utilizzare i campi dei form servono due pagine, la prima che raccoglie i dati con il form appunto e la seconda che li riceve ed elabora; come sapete form hanno 2 metodi per inviare i dati alla pagina che li riceve e sono get e post, personalmente preferisco il metodo post in modo che i dati riamngano invisibili nel passaggio.
Facciamo subito un piccolo esempio, farà la somma di 2 numeri inseriti dall'utente, che riassume tutto:
La prima pagina con i campi del form:

```
<FORM METHOD=POST ACTION="ricevi.php">
Il tuo nome: <INPUT TYPE="text" NAME="nome">
Il 1° numero: <INPUT TYPE="text" NAME="n1">
Il 2° numero: <INPUT TYPE="text" NAME="n2">
<INPUT TYPE="submit" value="Somma">
</FORM>
```

E la seconda pagina che riceve i dati inseriti (ricevi.php):

```
<?php
session_start();
/* qui riceviamo i dati immessi nel form assegnandoli alle variabili
   io di solito mantengo lo stesso nome dei campi del form
   se avessimo usato il metodo get anziché $_POST si sarebbe usato $_GET */
$nome = $_POST["nome"];
$n1 = $_POST["n1"];
$n2 = $_POST["n2"];

/* ed ora facciamo i nostri calcoli e li presentiamo! */

echo "Benvenuto $nome, grazie per aver usato la mia calcolatrice!<br>";
echo "Il primo numero inserito è: $n1<br>";
echo "Il secondo numero inserito è: $n2<br>";
echo "La somma è: ".($n1 + $n2);
?>
```

Mi sembra che non ci sia molto da spiegare è tutto semplice, una volta creata la pagina con il form (con tutti i campi necessari), realizziamo la pagina che li riceve, subito nelle prime righe recuperiamo i dati e poi nelle successive istruzioni li elaboriamo come serve.
Questo esempio non ha una vera e propria utilità dato che una calcolatrice che fa solo la somma non serve, ma se avete capito come funziona il tutto quando imperaremo ad utilizzare mysql verrà utilizzato per permettere agli utenti (o solo a voi) di inserire i record.

Lezione 7 - Passare le variabili

Una breve lezione su un breve argomento: come passare le variabili tra le pagine. Esistono 4 possibilità per passare i valori delle variabili da una pagina all'altra:

1. Con il metodo POST dei form
2. Con il metodo GET
3. Con le sessioni
4. Con i cookies

Il primo metodo, POST, l'abbiamo visto con i form nella lezione precedente, quindi lo saltiamo a piè pari.
Usando il metodo GET le variabili vengono passate attraverso il link e si utilizza $_GET["nomevar"] per recuperare i dati, ecco l'esempio:

Il link che invia i dati:

```
<a href='pagina2.php?var1=5&var2=7'>Prova passaggio valori con get</a>
```

E la pagina che li riceve:

```
<?php
session_start();
$var1 = $_GET["var1"];
$var2 = $_GET["var2"];
echo "Il valore di var1 è: $var1<br>";
echo "Il valore di var2 è: $var2<br>";
?>
```

Se notate l'indirizzo nella barra del browser vedrete che ci sono le due variabili con i rispettivi valori. Generalmente si utilizza questo metodo quando si devono passare tra le pagine dei valori che non sono inseriti dall'utente (prendete ad esempio i forum che ci sono su internet e guardate i collegamenti ai topic).
Il terzo metodo è quello che utilizza le sessioni (faremo una lezione specifica sulle sessioni più avanti), in sostanza salva il valore della variabile e può essere recuperato da ogni pagina (ricordate che deve iniziare con session_start()) e l'inconveniente è che non è possibile farlo in base alle scelte del visitatore. Mi spiego meglio, con il metodo precedente a seconda del link che il visitatore sceglie può essere passato un valore differente (ovvero quello del link), mentre con le sessioni si passano valori statici, un esempio semplice è ricordarsi il nome dell'utente finché resta da noi.
In una pagina chiediamo il nome dell'utente con un form, la pagina che riceve i dati dal form memorizza la variabile con il nome e tutte le pagine successive recuperano il nome dalla sessione:
La prima pagina che chiede il nome dell'utente:

```
<FORM METHOD=POST ACTION="nome.php">
Il tuo nome: <INPUT TYPE="text" NAME="nome">
<INPUT TYPE="submit" value="Dimmi!">
</FORM>
```

La pagina che riceve il nome (nome.php):

```
<?php
session_start();
$nome = $_POST["nome"]; // riceve il nome dal form
$_SESSION["nome"] = $nome; // salva il nome nella sessione
echo "Benvenuto: $nome<br>";
?>
```

In tutte le pagine successive per recuperare il nome del visitatore basterà il seguente codice:

```
<?php
session_start();
$nome = $_SESSION["nome"] = $nome; // recupera il nome dalla sessione
echo "Benvenuto: $nome<br>";
?>
```

Il nome dell'utente rimarrà presente fino a che la sessione non scade o finché resta nel nostro sito.
Il quarto ed ultimo metodo è l'utilizzo dei cookies, questo metodo è un po' più complesso e dipende dal browser del visitatore, se chi passa nel nostro sito ha disabilitato i cookies nel proprio browser non potremo usarlo. Questo metodo è generalmente utilizzato per ricordare alcuni dati dell'utente anche a distanza di tempo, prendete sempre come esempio i forum, una volta registrati se decidete di effettuare il login in automatico ogni volta che tornate, vi viene inviato un cookies (cioè un file di testo contenente il vostro nome utente e la vostra password) ed ogni volta che tornate al forum questo lo legge dal vostro pc e vi registra automaticamente.
I cookies li vedremo anche loro in una lezione più avanti, ma ecco un rapido esempio di come inviare un biscotto:

```
<?php
session_start();
$nome = $_POST["nome"]; // riceve il nome dal form
setcookie("nome",$nome); // invia il cookie
echo "Benvenuto: $nome<br>";
?>
```

E come riprendere il valore nelle pagine successive:

```
<?php
session_start();
$nome = $_COOKIE["nome"]; // riceve il nome dal biscotto!
echo "Benvenuto: $nome<br>";
?>
```

Riassumendo, i metodi più comodi per passare i valori delle variabili sono GET e POST, la sessione per passare valori temporaneamente (come ad esempio il nome utente) ed il cookie per ircordarsi a distanza di tempo i dati, come solitamente avviene in tutti i siti che usano PHP (in particolare i forum).
Un altro esempio è questo mio sito, uso una sola pagina per mostrarvi le lezioni passandole l'indice con il metodo GET (o le schede dei pesci/piante nella sezione dell'acquario), se vi registrate nella sezione di Diablo 2 - Lord of Destruction il metodo POST per recuperare i dati e la sessione per ricordarmi il vostro nome utente.

Lezione 8 - Includere file esterni

Includere file esterni nelle pagine in php ci permette di riutilizzare del codice in più pagine, limitando così il tempo di realizzazione del sito; un esempio pratico è la realizzazione di un menù presente su tutte le pagine, lo si scrive un'unica volta e lo si include in tutte, il risultato è semplice, una riga di codice in ogni pagina anziché l'intero menu e, soprattutto, se si vuole modificare il menù basta cambiare un unico file.
Esistono 2 modi per includere dei file esterni:

1. include("nomefile");

2. require("nomefile");

La differenza tra le due modalità è importante, se si utilizza include e il file da includere non è presente, la nostra pagina mostrerà un messaggio d'attenzione nel punto in cui avrebbe dovuto esserci il file e proseguirà l'esecuzione del codice successivo.
Se invece di include si utilizza require ed il file non è presente, la pagina arresterà la propria esecuzione ed il codice successivo a require non sarà eseguito. Vediamo subito un piccolo esempio:

La pagina con il menù, chiamata ad esempio menu.inc:

```
<TABLE>
<TR>
      <TD>Voce 1</TD>
      <TD>Voce 2</TD>
      <TD>Voce 3</TD>
</TR>
</TABLE>
```

In tutte le pagine che devono visualizzare il menù, in corrispondenza del punto in cui deve essere posizionato:

```
<?php
include("menu.inc");
?>;
```

Come potrete vedere se realizzate più pagine, il menù viene incluso senza doverlo riscrivere in ogni pagina. Se volete che l'eventuale mancanza del file contenente il menù blocchi l'esecuzione della pagina, utilizzate require anziché include.
Nell'esempio sopra riportato, il menù è scritto in html, ma potete utilizzare anche php!
Questo metodo è quello che utilizzo io in questo sito, sia per i menù che per visualizzare gli articoli, a seconda dell'id passato con il metodo get (lezione precedente!) includo una pagina differente, non utilizzo require in quanto aggiorno il menù solo quando è pronta la pagina con l'articolo.
E' tutto molto semplice, vero? Comunque utilizzeremo molto questi due metodi soprattutto per presentare i dati di un database, appena terminate le lezioni su PHP, a presto.

Lezione 9 - Le Sessioni

Premetto che questa non vuole essere una lezione esaustiva di tutto quello che si può fare con le sessioni, ma semplicemente suggerirvi un modo di utilizzo.

Le sessioni ci servono per un motivo molto semplice: "tracciare" l'utente che gira per il nostro sito.
Il funzionamento è semplice, controllano se il client ha abilitato i cookie altrimenti utilizza il metodo della variabile nascosta aggiunta automaticamente ad ogni pagina.
Per creare una sessione basta semplicemente chiamare la funzione Session_start(); all'inizio di ogni pagina in PHP, prima di qualsiasi altra istruzione, se la sessione è stata aperta precedentemente, una nuova chiamata a questa funzione farà riaprire la vecchia sessione.
E’ possibile memorizzare nella sessione alcune variabili personali dell’utente, per averle a disposizione in tutte le pagine semplicemente utilizzando $_SESSION["nomevariabile"] = valore; per poi recuperarla nelle pagine successive con $variabile = $_SESSION["nomevariabile"];
La sessione, per impostazione predefinita, termina alla chiusura del browser, tuttavia rimane memorizzata nella memoria del server se la sessione è ancora aperta ed è possibile cancellare l’intera sessione con tutte le variabili correlate utilizzando session_destroy();
La sessione ovviamente deve essere aperta per poter essere cancellata.

Lezione 10 - I Cookies

I cookies, o biscotti per chi preferisce, sono dei piccoli file di testo che il sito può generare sul PC dell'utente per salvare alcune informazioni, sempre che l'utente li abbia abilitati nel proprio browser ovviamente.

In questa breve lezioncina vedremo come impostare e controllare i cookies, così chiudiamo la parte di articoli dedicati a PHP.

Per inviare un biscotto sul computer del visitatore si utilizza la funzione setcookie:

```
<?php
setcookie("nomecookie", "valore", durata);
?>
```

Chiariamo subito i parametri di questa funzione:
nomecookie: è il nome del biscotto, quello che poi utilizzeremo anche per recuperare il valore;
valore: è il valore che contiene il biscotto, un numero, una stringa, ecc.;
durata: è il tempo durante il quale il biscotto rimane valido (numero di secondi), se non indiachiamo nulla il biscotto sarà valido per sempre.
Alcuni esempi di esempi chiarificatori:

Il primo imposta un cookie con il nome dell'utente, precedentemente memorizzato nella variabile $utente, che sarà valido per sempre, ogni volta che la stessa persona torna al nostro sito potremo recuperare il suo nome da questo biscottino:

```
<?php
setcookie("utente", $utente);
?>
```

L'esempio precedente lo impostiamo in modo da durare per un'ora:

```
<?php
setcookie("utente", "paperino", time()+3600);
?>
```

time()+3600 indica, come già detto, un'ora di durata (1 ora = 3600 secondi) a partire da quando viene impostato (time()).

Un'altra cosa importante è quella di eliminare un cookie e la cosa è semplice come impostarlo, dovete semplicemente richiamare il cookie con gli stessi valori con cui è stato impostato; per eliminare il biscotto dell'esempio precedente:

```
<?php
setcookie("utente", "paperino", time()-3600);
?>
```

Ora passiamo al controllo e recupero del nostro cookie, subito l'esempio (recuperiamo quello dell'esempio precedente) e poi lo spiego:

```
<?php
if (isset ($_COOKIE["utente"])) {
      $utente = $_COOKIE["utente"];
      echo $utente;
}
?>
```

Come potete vedere è semplicissimo: con l'if controlla se il cookie è impostato sul pc dell'utente, in caso positivo recupera il valore, lo salva nella variabile $utente e poi lo mostra.
Ecco fatto, ora siete in grado di gestire per bene un intero sito, con variabili, sessioni, cookies, form e quasi tutto quello che serve per interagire con gli utenti.

Le nozioni base di PHP dovreste ormai conoscerle, per cui le prossime saranno su come usare PHP con MySQL.