HyperText Preprocessor 4

Mauro Malvestio

2002
v1.1

Sommario

1. Introduzione

1.1 – Il php storia ed evoluzione di un linguaggio

Questa documentazione tratta di PHP, il linguaggio di scripting per il Web di tipo open source che ha raggiunto Perl, Asp, e Java nel gruppo dei linguaggi utilizzabili per creare ambienti online dinamici.

PHP è un linguaggio che si è sviluppato ben oltre i limiti impliciti nel proprio nome. In origine era stato concepito come un insieme di macro tese a facilitare ai programmatori il compito di amministrare le home page personali; da qui il nome (Personal Home Page). Da allora, però le capacità di PHP sono state ampliate e lo hanno trasformato da una serie di strumenti di utilità in un linguaggio di programmazione completo, in grado di amministrare ambienti online basati su database, anche di notevoli dimensioni.

La popolarità di PHP, in seguito, è cresciuta di pari passo con l'accrescersi delle sue potenzialità. Secondo NetCraft http://www.netcraft.com, nel novembre del 1999 PHP era presente su più di un milione di Host. Nel febbraio 2000 il numero di questi era salito a un milione e quattrocentomila.

PHP è attualmente conosciuto come PHP: HyperText Preprocessor; si tratta di un linguaggio di scripting lato server generalmente usato in un contesto html. A differenza di una normale pagina HTML, uno script PHP non viene inviato direttamente dal server al client; viene analizzato dal modulo o dal file binario di PHP.
I tag HTML presenti nello script vengono mantenuti tali, mentre il codice PHP viene interpretato ed eseguito. Il codice contenuto in uno script PHP è in grado di interrogare database, creare immagini, leggere e scrivere file, dialogare con server remoti; le possibilità sono realmente infinite. L'output prodotto dal codice PHP viene poi unito con la parte HTML dello script fino a ottenere il risultato definitivo, che viene trasmesso all'utente.

La prima versione del PHP fu creata da Rasmus Lerdorf nel 1994 come serie di macro rilasciate sotto il nome di Personal Home Page Tools e, in seguito, riscritte e ampliate fino a comprendere un pacchetto chiamato Form Interpreter (PHP/FI) .
Dal punto di vista dell'utente, PHP/FI costitutiva già di per se una proposta interessante, e di pari passo iniziò e aumentò anche l'interesse della comunità degli sviluppatori a tal punto che nel 1997 un'intera squadra di programmatori dedicava il proprio lavoro al progetto PHP3
La versione successiva PHP3 infatti scaturì da questi sforzi congiunti. PHP3 era effettivamente una riscrittura di PHP, con un parser completamente nuovo creato da Zeev Suraski e Andi Gutmans, con differenze di sintassi e nuove funzionalità. Questa versione fece di PHP uno dei linguaggi di scripting lato server più appetibili e interessanti
Il supporto offerto per Apache e MySql non fece altro che rafforzare la popolarità di PHP, tutt'oggi Apache è il server Web più utilizzato al mondo e PHP3 può essere compilato come modulo di Apache, e PHP dispone di una libreria di funzioni per MySql.

La crescita della popolarità di PHP è avvenuta in concomitanza con un cambiamento degli atteggiamenti verso la pubblicazione sul Web; basti pensare che verso la metà degli anni novanta era ancora del tutto normale realizzare siti, anche di dimensioni ragguardevoli, mediante centinaia di pagine singole codificate manualmente in HTML. E' pero andata aumentando la capacità di trattamento dei database, che consentono di amministrare più efficacemente i contenuti e di offrire un maggiore livello di personalizzazione dei siti in base alle preferenze dell'utenza.
L'impiego dei database come fonte di dati e di un linguaggio di scripting dedicato al recupero dinamico dei dati diventerà ancora più necessario quando questi dati dovranno essere trasmessi, da un'unica fonte, a una serie di destinatari differenti tra cui telefoni cellulari, PDA, TV digitali e ambienti Internet a banda larga.

1.2 – Le novità di PHP4

PHP4 rappresenta il decisivo salto di qualità, con l'introduzione di nuove svariate librerie contenenti a loro volta centinaia di funzioni che potranno rendere molto più facile la vita dei programmatori.

Le novità sono:

- Una nuova istruzione *foreach,* simile a quella presente in Perl.
- La nuova introduzione del tipo boolen.
- Il nuovo supporto nativo alle sessioni utente.
- L'introduzione dell' operatore (===) che verifica l'equivalenza di tipo e valore.
- E stato ampliato il supporto relativo alla programmazione ad oggetti.
- PHP4 comprende un supporto nativo per Java e XML

1.3 – Il motore Zend

Il nuovo PHP4 ha un parser completamente nuovo, modificato ed ottimizzato a livello di motore. Queste modifiche a livello di motore saranno in grado di apportare al nuovo PHP4 un successo duraturo, la gran parte di codice PHP3 scritto in passato, continuerà ad essere eseguito senza richiedere modifiche, ma la velocità di esecuzione sarà incrementata di 200 volte.

1.4 – Prestazioni & Portabilità

Grazie alla potenza del motore Zend, PHP4 regge molto bene il confronto con ASP nei test di benchmark, raggiungendo in alcune prove specifiche il primato. Il codice PHP compilato, poi, stacca ASP di varie lunghezze.

Php è stato progettato per essere eseguito su svariati sistemi operativi e per colloquiare con molti web-server e database. Diventa così possibile realizzare progetti destinati a un ambiente UNIX e poi passare a Windows NT senza incontrare alcun problema. Si possono condurre le fasi di test di un progetto con Personal Web Server di Microsoft e, in seguito, installare il tutto su un sistema UNIX in cui PHP viene eseguito come modulo di Apache.

2. Fondamenti di PHP

2.1 – Tag di Apertura e chiusura

Il codice PHP embedded in HTML deve essere racchiuso tra i tag <? ?>, che indicano al parser PHP che si intende mandare in esecuzione i comandi inseriti all'interno dei tag <? ?>, in caso contrario il codice verrà scambiato per un normale HTML e visualizzato nel browser.
La seguente tabella indica i tag che offrono garanzia del corretto funzionamento su ogni piattaforma.

Stile Tag	Tag iniziale	Tag finale
Tag Standard	<?php	?>
Tag Brevi	<?	?>
Tag ASP	<%	%>
Tag Script	<SCRIPT LANGUAGE="php">	</SCRIPT>

Per attivare il riconoscimento per i tag brevi occorre assicurarsi che, nel file php.ini, l'opzione **short_open_tag** sia impostata a **On**.

```
Short_open_tags = On;
```

Per attivare il riconoscimento per i tag ASP® tecnologia proprietaria Microsoft®, e necessario modificare il file php.ini e abilitare l'opzione **asp_tags**.

```
Asp_tags = On;
```

IL MIO PRIMO SCRIPT PHP

```
<html>
<head>
<title> Il mio primo script php </title>
</head>
<body>
<?php
    print "Ciao Mondo";
?>
</body>
</html>
```

2.2 – Aggiunta di commenti al codice PHP

Il codice in chiaro in fase di creazione può verificarsi complesso da comprendere anche dopo poco tempo dalla stesura, i commenti permettono di aggiungere al codice informazioni utili allo sviluppatore per aumentare chiarezza e comprensione del codice.

Tipo	Stile
// Commento	C++
/* Commento */	C
# Commento	Bourne Shell

2.3 – Variabili

Una variabile in PHP deve essere immaginata come un contenitore dove può essere conservato un valore qualsiasi. Una variabile è formata da un nome il più evocativo possibile, preceduto dal simbolo del dollaro ($).

Nota:
Nel nome di variabile non devono comparire lettere, numeri o caratteri alfanumerici

```
$num1 = 10;
$nome = “Mauro”;
```

Per le variabili in PHP, la distinzione non è cosi ben evidente; infatti esistono le:

- Variabili di Apache
- Variabili di PHP/Ambientali

2.4 – Variabili di Apache

Variabile	Descrizione
GATEWAY_INTERFACE:	la versione delle specifiche CGI utilizzate dal server, ad esempio "CGI/1.1";
SERVER_NAME:	il nome del server;
SERVER_SOFTWARE:	il nome del software utilizzato per il webserver;
SERVER_PROTOCOL:	il nome e la versione del protocollo;
SERVER_PROTOCOL:	il nome e la versione del protocollo tramite il quale è stata richiesta la pagina;
REQUEST_METHOD:	utilizzato nelle form, può essere "GET", "POST", "HEAD", "PUT";
QUERY_STRING:	se presente, la stringa tramite la quale la pagina è stata richiesta;
DOCUMENT_ROOT:	la DocumentRoot del server, come configurata in httpd.conf, ad esempio "/var/www";
HTTP_ACCEPT:	il contenuto dell'header Accept, ad esempio "text/*, image/*, audio/*, application/*";
HTTP_ACCEPT_CHARSET:	il charset accettato, ad esempio "iso-8859-1";
HTTP_ENCODING:	l'encoding della richiesta, se presente; ad esempio, "gzip";
HTTP_ACCEPT_LANGUAGE:	il linguaggio, ad esempio "en";
HTTP_CONNECTION:	il contenuto dell'header Connection, ad esempio "Keep-alive";
HTTP_HOST:	il nome dell'host, ad esempio "localhost";
HTTP_REFERER:	il nome della pagina dalla quale si proviene;
HTTP_USER_AGENT:	il contenuto dell'header User_Agent;
REMOTE_ADDR:	l'indirizzo IP dell'utente connesso alla nostra pagina;
REMOTE_PORT:	come sopra, ma riferito alla porta; tipo: "127.0.0.1 1275";
SCRIPT_FILENAME:	il nome dello script richiesto al server, ad esempio "index.php";
SERVER_ADMIN:	il nome dell'amministratore di sistema;
SERVER_PORT:	la porta sulla quale il server è in ascolto, ad esempio la porta 80;
SERVER_SIGNATURE:	l'eventuale "signature" del server;
PATH_TRANSLATED:	il percorso per lo script richiamato, ad esempio "/mauro/prova.php";
SCRIPT_NAME:	il path, a partire dalla DocumentRoot, dello script; ad esempio, “/prova.php";
REQUEST_URI:	l'URI richiesto per accedere alla pagina;

2.5 – Variabili di PHP

Variabile	Descrizione
argv:	un array contenente i parametri passati allo script;
argc:	come sopra, ma se lo script è>argv: un array contenente i parametri passati allo script;
argc:	come sopra, ma se lo script è eseguito dalla linea di comando;
PHP_SELF:	il nome dello script attualmente in esecuzione;
HTTP_COOKIE_VARS:	un array associativo contenente le variabili passate allo script tramite i cookies
HTTP; HTTP_GET_VARS:	un array associativo contenente le variabili passate allo script tramite una richiesta GET;
HTTP_POST_VARS:	come sopra, ma riferito al metodo POST.

2.6 – Tipi di Dati

Come si sa tipi di dati differenti occupano quantità di memoria differenti, è bene per questo scegliere con cura il tipo di dato più adatto per risolvere il problema specifico.
PHP4 offre vari tipi di dati per compiere le operazioni più svariate, oltre tutto consente anche il confronto per tipo, tramite l'operatore ===, (maggiori informazioni nella sezione Operatori).
Offre anche la possibilità di verificare il tipo dei dati contenuti in una variabile tramite la funzione **gettype()**, la quale accetta come parametri il nome della variabile da esaminare e restituisce una stringa che rappresenta il tipo in questione.

Tipo	Descrizione
Integer	Formato intero, che in genere riflette l'ampiezza degli interi sulla piattaforma utilizzata
Double	Floating-point a doppia precisione.
String	Una collezione di caratteri
Boolean	Valore booleano (1) True, (0) False
Object	Rappresenta una classe
Array	Pùo essere una collezione di un determinato tipo

VERIFICA DEL TIPO DI DATI DI UNA VARIABILE

```
<html>
<head>
<title> Verifica del tipo di dati di una variabile. </title>
</head>
<body>
<?php
$Test = 5;
Print Gettype( $Test ); // integer
Print "<br>";
$Test = "five";
Print Gettype( $Test ); // string
Print("<br>");
$Test = 5.0;
Print Gettype( $Test ); // double
Print("<br>");
$Test = true;
Print Gettype( $Test ); // boolean
Print "<br>";
?>
</body>
</html>
```

2.7 – Operatori aritmetici

La seguente tabella elenca gli operatori aritmetici messi a disposizione da un linguaggio di programmazione standard.

Operatore	Nome	Esempio	Risultato
+	Addizione	10+3	13
-	Sottrazione	10-3	7
/	Divisione	10/3	3,333333333
*	Moltiplicazione	10*3	30
%	Modulo	10%3	1

2.8 – Operatore di Concatenamento

Questo operatore è costituito dal singolo "." e considera entrambi gli operandi come delle stringhe, questo operatore aggiunge l'operatore di destra a quello di sinistra.

```
      "Ciao" . " mondo"
diventa:
      "ciao mondo"
```

2.9 – Altri Operatori di Assegnazione

PHP4 fornisce una serie di operatori che consentono di applicare delle assegnazioni combinate, evitando quindi l'uso di operatori singoli:
La seguente tabella riporta gli operatori di assegnazione combinata, con un esempio equivalente.

Operatore	Esempio	Equivalente a
+=	$x += 5	$x = $x + 5
-=	$x -= 5	$x = $x – 5
/=	$x /= 5	$x = $x / 5
*=	$x *= 5	$x = $x * 5
%=	$x %= 5	$x = $x % 5
.=	$x .= "Ciao"	$x = $x. "Ciao"

2.10 – Operatori logici

Gli operatori di confronto effettuano controlli sui propri operandi, gli operatori di confronto supportati dalla versione 4 di PHP.

Operatore	Nome
==	Equivalenza
!=	Diverso
===	Identità
>	Maggiore
>=	Maggiore o uguale
<	Minore
<=	Minore o uguale

Gli operatori logici sono riportati nella tabella seguente.

Operatore	Nome
||	Or Logico
Or	"
Xor	Or Escusivo
&&	And Logico
And	"
|	Not

2.11 – Incremento e decremento di una variabile intera

Durante la programmazione in PHP ci si trova spesso a dover incrementare o decrementare una variabile intera, il PHP4 come la maggior parte dei linguaggi di programmazione mette a disposizione gli operatori di incremento e decremento.

```
$x = $x + 1;
$x += 1;
$x++;
```

2.12 – Le costanti

Se si desidera che un determinato valore non subisca alcuna modifica durante l'esecuzione dello script, è possibile definire una costante.
Per definire una costante è d'obbligo usare la funzione **define()**, di PHP, che accetta come parametri il nome della costante da definire, e il valore che si intende attribuire alla costante.

```
Define ( "NOME_COSTANTE", 50 );
```

Per convenzione il nome delle costante va espresso in maiuscole, per accedere al valore della costante non è obbligatorio far precedere il simbolo di dolloro ($).

DEFINIZIONE DI UNA COSTANTE

```
<html>
<head>
<title> Definizione di una costante</title>
</head>
<body>
<?php
Define ( "UTENTE", "Mauro" );
Print "Benvenuto ".UTENTE;
?>
</body>
</html>
```

3. Controllo del flusso di elaborazione

Per modificare il flusso di elaborazione il PHP offre le seguenti istruzioni:

- Espressione condizionale *If*
- Istruzione condizionale *Switch*
- Operatore ternario *?*
- L'istruzione iterativa *While*
- L'istruzione iterativa *Do..While*
- L'istruzione iterativa *For*
- L'istruzione di interruzione *Break*
- L'istruzione di continuazione *Continue*
- L'istruzione iterativa *ForEach*

3.1 – Espressione condizionale *If*

Sintassi:

```
if ( espressione ) {
    blocco1;
}//THEN
else {
    blocco2;
}//ELSE

if ( espressione ) {
    blocco1;
}//THEN
elseif ( espressione_2 ) {
    blocco2;
}//ELSEIF
else {
    blocco3;
}//ELSE
```

3.2 – Istruzione condizionale *Switch*

Sintassi:

```
switch ( espressione ) {
    case risultato_1:
        blocco1;
    break;
    case risultato_2:
        blocco2;
    break;
    .
    .
    .
    case risultato_n:
        bloccon;
    break;
}//SWITCH
```

3.3 – Operatore ternario *?*

Sintassi:

```
( espressione ) ? restituito_se_espr_e_vera : restituito_se_espr_e_falsa;
```

3.4 – L'istruzione ciclica *While*

Sintassi:

```
while ( espressione ) {
    .
    .
    blocco;
    .
    .
}//WHILE
```

3.5 – L'istruzione iterativa *Do..While*

Sintassi:

```
do {
    .
    .
    blocco;
    .
    .
} while ( espressione ); //DO
```

3.6 – L'istruzione iterativa *For*

Sintassi:

```
for ( variabile = valore; espressione test; incremento variabile ) {
    .
    .
    blocco;
    .
    .
}//FOR
```

3.7 – La parola chiave *Break*

Break esce dal ciclo più interno che la contiene.

3.8 – La parola chiave *Continue*

Continue forza l'esecuzione dell' iterazione successiva.

3.9 – L'istruzione iterativa *ForEach*

ForEach è un nuovo potente costrutto iterativo introdotto dalla versione 4 di PHP, che permette di eseguire cicli su array indicizzati numericamente, e associativi per chiave.

Sintassi: Indicizzato numericamente.

```
Foreach ( $array as $temp ) {
    .
    .
    blocco;
    .
    .
}//FOREACH
```

Sintassi: Associativo per chiave.

```
Foreach ( $array as $chiave => $valore ) {
    .
    .
    blocco;
    .
    .
}//FOREACH
```

4. Le Funzioni

4.1 – Definizione

Per definire una funzione in PHP4 si deve utilizzare la parola chiave **function**:

```
Function esempio ( $arg_1, $arg_2, … , $arg_n  ) {
    .
    .
    codice;
    .
    .
}//ESEMPIO
```

Nella dichiarazione, si deve specificare il nome della funzione, e una lista di parametri attuali.
È bene utilizzare il più possibile le funzioni per parametrizzare, e risolvere problemi mediante la metodologia **top-down**.

4.2 – Restituzione di valori

Per restituire un singolo valore, si deve utilizzare la parola chiave **return.** Altrimenti per ottenere dei parametri di input/output si possono passare le variabili per indirizzo, facendo precedere nella chiamata di funzione in simbolo di e-commerciale (&) alle variabili.

UNA FUNZIONE CHE RESTITUISCE UN VALORE

```
<html>
<head>
<title> Una funzione che restituisce un valore </title>
</head>
<body>
<?php
Function SommaNumeri( $PrimoNumero, $SecondoNumero ) {
     $Risulatato = $PrimoNumero + $SecondoNumero;
     Return $Risultato;
}//SOMMANUMERI
Print SommaNumeri(3,5);
?>
</body>
</html>
```

4.3 – Chiamate di funzioni dinamiche

È possibile assegnare, come stringhe, nomi di funzioni a variabili, e quindi trattare queste variabili proprio come se fossero funzioni.

CHIAMATA DINAMICA A FUNZIONE

```
<html><head>
<title> Chiamata dinamica a funzione</title>
</head>
<body>
<?php
function PrintCiao() {
    print "Ciao”<br>";
}//PRINTCIAO
$function_name = "PrintCiao";
$function_name();
?>
</body>
</html>
```

5. Array

5.1 – Definizione di Array con la funzione Array()

La funzione array risulta molto utile quando si desidera assegnare a un array una serie di valori in una sola volta.

```
$Utenti = Array (“Mauro”,”Alice”,”Mirco”,”Fabio” );
```

A questo punto e possibile accedere al terzo elemento tramite indice, $utenti[2].

5.2 – Array Associativi

Gli array indicizzati numericamente sono utili quando si vogliono memorizzare dei valori in base al loro ordine di inserimento o in base a un differente criterio di ordinamento. A volte però sarebbe necessario accedere agli elementi di un array mediante un nome evocativo che si associa all’elemento, qui vengono in aiuto gli array associativi. Gli indici di un array associativo sono formati da stringhe anziché da numeri.

5.3 – Definizione di un array associativi tramite la funzione Array()

Per definire un array associativo tramite la funzione **Array()**, è necessario definire per ogni elemento sia la chiave che il valore.

```
$Utente = Array ( Nome=>”Mauro”,
                  Professione=>”Studente”,
                  Eta=>”19” );
```

A questo punto è possibile accedere a uno qualsiasi degli elementi di $Utente:

```
Print $Utenti[Eta];
```

5.4 – Array Multidimensionali

Come in C e in C++, si incontra anche in PHP l’array di array cioè l’array mutidimensionale; la differenza sostanziale da un comune array resta nell’uso degli indici; il primo indica l’array nell’array, il secondo il campo specifico dell’array.
Il seguente è un esempio di definizione di un array multidimensionale.

DEFINIZIONE DI UN ARRAY MULTIDIMENSIONALE

```
<html>
<head>
<title> Definizione di un array Multidimensionale </title></head>
<body>
<?php
$Utenti = Array (
              Array ( Nome=>"Mauro",
                  Professione=>"Programmatore",
                  Eta=>19 ),
              Array ( Nome=>"Alice",
                  Professione=>"Web-Designer",
                  Eta=>18 ),
          );

print $Utenti[0][Professione];
?>
</body>
</html>
```

6. I moduli

Sul Web, i moduli HTML rappresentano il mezzo principale per passare informazioni dall'utente al server. PHP è stato progettato per acquisire le informazioni inviate mediate i moduli HTML e lavorare su di esse.

6.1 – Uno script per acquisire l'input dell'utente

Nel seguente esempio viene definito un modulo che contiene un campo di testo chiamato utente e un campo chiamato indirizzo.
L'effetto provocato è il passaggio delle variabili Utente e Indirizzo mediate la variabile **QUERY_STRING** del Server Web al file stampa.php

UN SEMPLICE MODULO HTML

```
<html>
<head>
<title> Un semplice modulo HTML</title>
</head>
<body>
<form action="stampa.php" method="GET">
<input type="text" name="Utente">
<br>
<textarea name="Indirizzo" rows="5" cols="40">
</textarea>
<br>
<input type="submit" value="Vai!">
</form>
</body>
</html>
```

COME LEGGERE LE INFORMAZIONI INVIATE DA UN MODULO

```
<html>
<head>
<title> Come leggere le informazioni inviate da un modulo</title>
</head>
<body>
<?php
print "Benvenuto <b>$Utente</b><P>\n\n";
print "Il tuo indirizzo è:<P>\n\n<b>$Indirizzo</b>";
?>
</body>
</html>
```

6.2 – Accesso a tutti i campi di un modulo in un array associativo

È attualmente oggi la tecnica più potente e affidabile per processare moduli, anche di notevoli dimensioni. Si basa sul fatto di considerare tutti gli elementi di un modulo come un grande array associativo. E a seconda del metodo utilizzato sia esso GET o POST si avranno a disposizione le variabili di PHP **$HTTP_GET_VARS** e **$HTTP_POST_VARS**.
Questi sono infatti degli array associativi che contengono coppie di chiave valore.

UTILIZZO DELL'ARRAY $HTTP_GET_VARS PER LEGGERE L'INPUT DI UN MODULO QUALSIASI

```
<html>
<head><title> Utilizzo dell'Array $HTTP_GET_VARS per leggere l'input di un
modulo qualsiasi </title>
</head>
<body>
<?php
Foreach ( $HTTP_GET_VARS as $Chiave=>$Valore ) {
   print "Chiave == $Valore<BR>\n";
}//FOREACH
?>
</body>
</html>
```

6.3 – Come leggere i campi di un modulo qualsiasi

Per dare maggiore flessibilità e potenza ad uno script che può accettare dati da una fonte qualsiasi, si deve decidere se leggere l'array **$HTTP_GET_VARS** o l'array **$HTTP_POST_VARS**. La distinzione della transazione utilizzata può essere fatta utilizzando la variabile ambientale **$REQUEST_METHOD**, che contiene di volta in volta il tipo della transazione.

COME ESTRARRE I PARAMETRI DA UNA RICHIESTA QUALSIASI

```
<html>
<head>
<title> Come estrarre i parametri da una richiesta Get o Post </title>
</head>
<body>
<?php
$PARAMS = ( Isset( $HTTP_POST_VARS ) ) ? $HTTP_POST_VARS : $HTTP_GET_VARS;
Foreach ( $PARAM as $Chiave=>$Valore ) {
  if ( Gettype( $Valore ) == "array" ) {
      Print "$Chiave == <br>\n";
      Foreach ( $Valore as $Dim_Valore )
      Print ".........$Dim_Valore<br>";
  }//THEN
  else {
      Print "$Chiave == $Valore<br>\n";
  }//ELSE
}//FOREACH
?>
</body>
</html>
```

6.4 – Unione di codice HTML e PHP in una sola pagina, $PHP_SELF

In determinate circostanze è preferibile riunire il codice di analisi del modulo e il codice HTML nella medesima pagina.
Una simile combinazione può essere utile quando si deve presentare più di una volta lo stesso modulo all'utente.
Per far si che una pagina venga da se richiamata quando avviene l'evento **OnSubmit**, si deve inserire nell'argomento action il valore **$PHP_SELF**.

UN MODULO HTML CHE RICHIAMA SE STESSO – L'INDOVINA NUMERI

```
<?php
$Num_Da_Indovinare = 42;
$Messaggio = "";
if ( ! isset( $Numero ) ) {
   $Messaggio = "Benvenuto nella macchina indovina numeri!";
}//THEN
elseif ( $Numero > $Num_Da_Indovinare ) {
   $Messaggio = "Il numero $Numero è troppo grande. Riprova!";
}//ELSEIF
elseif ( $Numero < $Num_Da_Indovinare ) {
   $Messaggio = "Il numero $Numero è troppo piccolo. Riprova!";
}//elseif
else {
   $Messaggio = "Indovinato!";
}//ELSE
?>
<html><head><title> L'indovina Numeri </title></head><body>
<h1><?php print $Messaggio ?></h1>
<form action="<?php print $PHP_SELF?>" method="POST">
Digita il tuo numero: <input type="text" name="Numero">
</form>
</body>
</html>
```

L'OUTPUT PRODOTTO DALL'ESEMPIO PRECEDENTE

```
<html>
<head>
<title> L'indovina Numeri </title>
</head>
<body>
<h1>Indovinato!</h1>
<form action="/php4/php.exe/indovina_numeri.php" method="POST">
Digita il tuo numero: <input type="text" name="Numero">
</form>
</body>
</html>
```

Si noti con particolare attenzione l'attributo action del tag FORM, il parser PHP4 ha sostituito in modo dinamico **$PHP_SELF** con il nome del file .php interpretato.

6.5 – Effettuare un Redirect

Quando uno script lato-server comunica con il client, deve innanzitutto inviare determinate intestazioni che informano sul documento che seguirà. PHP, si occupa automaticamente di questo compito, ma è anche possibile scegliere di inviare intestazioni personalizzate utilizzando la funzione **Header()**. Per richiamare questa funzione è necessario accertarsi del fatto che al browser non venga inviato alcun output.

Le seguenti sono tipiche intestazioni inviate da PHP4 al browser.

```
HEAD /mauro/esempi/es11.php HTTP/1.0
HTTP/1.1 200 OK
Date: Fri, 06 Apr 2001 16:37:45 GMT
Server: Apache/1.3.9 (Win32) PHP/4.0
Connection: close
Content-Type: text/html
```

Inviando un intestazione “Location” al posto di quella inviata per default da PHP, è possibile reindirizzare il browser a una nuova pagina:

```
Header (“Location: http://www.newebsolutions.f2s.com”);
```

Tramite la funzione **header()**, e senza gestire file .htaccess è possibile gestire l’autorizzazione a delle pagine protette, solamente inviando le corrette intestazioni HTTP al browser.

AUTORIZZAZIONE PER PROTEGGERE UNA PAGINA

```
<?
$Username = "Mauro";
$Pwd = "Segreta";
if (!isset($PHP_AUTH_USER)) {
    Header("WWW-Authenticate: Basic realm=\"Zona Protetta\"");
    Header("HTTP/1.0 401 Unauthorized");
    echo "Impossibile eseguire l'autorizzazione.";
}// THEN
else {
if (($PHP_AUTH_USER == $Username ) && ($PHP_AUTH_PW == $Pwd )) {
    echo "Autorizzazione riuscita per $Username.";
}//THEN
else {
    echo "Autorizzazione fallita per $Username.";
}//Else
}//ELSE
?>
```

Inserendo il precedente codice in file .inc, e includendolo in ogni pagina di un sito, si può far vedere le pagine solo agli utenti che si autenticano con successo.

Il risultato dell’inclusione del file sopracitato nel file index.php di http://www.newebsolutions.f2s.com è il seguente:

7. I Cookie

Un cookie è un piccolo quantitativo di dati memorizzati dal browser in seguito ad una richiesta di un server o di uno script.
Un host può memorizzare un massimo di 20 cookies all'interno del browser dell'utente.
Ogni cookies è composto, oltre che dal nome dell'host di provenienza e dal relativo percorso, da un nome, da un valore, e da una data di scadenza. I cookie non possono avere dimensioni superiori a 4KB.
I cookie vengono generalmente impostati in un'intestazione HTTP, uno script PHP che voglia impostare un cookie dovrà inviare delle intestazioni analoghe alle seguenti:

```
HTTP/1.1 200 OK
Date: Fri, 04 April 2001 21:03:38 GTM
Server: Apache/1.3.9 (Win32) PHP/4.0b.3
Set-Cookie: prova:mio_cookie; expires=Friday,
[ic:ccc]04-Apr-00 22:03:38 GTM; patt=/; domain=newebsolutions.f2s.com
Connectio: close
Content-Type: text/html
```

L'intestazione **Set-Cookie**, contiene una coppia nome/valore, una data relativa al meridiano di Greenwich (GTM), un percorso e un dominio. Il nome e il valore sono URL-encoded. Il campo **expires** indica al browser di "dimenticare" il cookie dopo la data e l'ora specificate.
Il campo **path** definisce il livello del sito Web sotto del quale le pagine sono autorizzate a richiedere l'invio del cookie verso il server. Il campo **domain** determina i domini Internet a cui il cookie può essere inviato, il dominio non può differire da quello che ha richiesto la creazione del cookie.
Se il browser è configurato in modo da permettere l'uso dei cookie, questi rimarranno memorizzati fino alla loro scadenza. E quando l'utente naviga su qualsiasi pagina che corrisponda al dominio di un cookie memorizzato, il browser invia il relativo blocchetto di informazioni al server.

7.1 – Impostazione di un cookie con PHP

In PHP esistono due modi tramite i quali è possibile inviare cookie. Si può utilizzare la funzione **Header()** per generare l'intestazione Set-Cookie, nel seguente modo:

```
Header ("prova=mio_cookie; espires=Friday, 04-Apr-01 21:03:38 GTM; path=/;
domain=newebsolutions.f2s.com
```

oppure tramite la funzione **Setcookie()**, che ha il compito di produrre un'intestazione Set-Cookie.
N.B. La funzione **Setcookie()**, deve essere chiamata prima di ogni output sul browser.

IMPOSTAZIONE E STAMPA DI UN COOKIE

```
<?php
Setcookie("prova","mio_cookie",time()+3600,"/","newebsolutions.f2s.com",0)
?>
<html>
<head>
<title> Impostazione e stampa di un cookie </title>
</head>
</body>
<?php
if ( isset( $prova ) )
    print "<p> Ciao. Qui sei sempre il benvenuto </p>";
else
    print  "<p> Ciao. Questa potrebbe essere la tua prima visita </p>";
?>
</body>
</html>
```

7.2 – Cancellazione di un cookie

In teoria, per cancellare un cookie si dovrebbe richiamare la funzione **Setcookie()** con il solo nome del cookie in questione:

```
SetCookie( “prova” );
```

Questo metodo però non funziona sempre ed è bene non farvi affidamento. Quindi a mio parere è più sicuro impostare il cookie su una data già trascorsa.

```
Setcookie(“prova”,“mio_cookie”,time()-60,”/”,”newebsolutions.f2s.com”,0)
```

Comunque e meglio assicurarsi che **Setcookie()** abbia lo stesso percorso e parametro di sicurezza che è stato passato all’atto della creazione del cookie.

7.3 – Creazione di un cookie di Sessione

Un cookie di sessione è un particolare cookie che dura per un’unica sessione utente; si può realizzare passando a **Setcookie()** una scadenza di 0 secondi, così il browser non ricorderà il cookie dopo la sua chiusura e la sua successiva esecuzione.
Questo è molto utile quando si vuole autenticare un utente tramite cookie, si può quindi permettere l’accesso a informazioni contenute in pagine multiple dopo l’invio di un unico login.

```
SetCookie ( “session_id”, “55453”, 0 );
```

8. Le sessioni

8.1 – Salvataggio dello stato con le funzioni di Sessione

Le funzioni di sessione forniscono all'utente un identificatore unico che può essere utilizzato da un accesso all'altro per gestire informazioni personali. In questo caso il linguaggio PHP svolge già una gran parte del lavoro.
Quando l'utente accede alla prima pagina di questo tipo, se si tratta della prima visita gli viene allocato un nuovo identificatore univoco utilizzato per l'associazione tra client e relativa sessione, altrimenti usa l'accesso precedente.

Le sessioni possono venire gestite in due modi, ambedue supportati da PHP4:

- Tramite *cookie*
- Tramite propagazione dell' *id di sessione*

Lo stato della sessione viene generalmente memorizzato in un file temporaneo, ma a breve saranno disponibili moduli che supportino l'uso dei database più diffusi.

8.2 – Apertura di una sessione con Session_start()

Se il file di configurazione php.ini non è stato modificato, tutte le sessioni si devono aprire esplicitamente tramite la funzione **Session_start()**.
Dopo l'apertura di una sessione si ha subito accesso all'ID di sessione tramite la funzione **Session_id ()**.

AVVIO E RIPRISTINO DI UNA SESSIONE

```
<?php
Session_start();
<html>
<head>
<title> Avvio e ripristino di una sessione </title>
</head>
</body>
<?php
print "<p> Benvenuto, il tuo ID di sessione è ".Session_id()."</p>\n\n";
?>
</body>
</html>
```

8.3 – Le variabili di sessione

Tramite le sessioni è possibile memorizzare un numero qualsiasi di variabili globali nella sessione ed accedervi da qualsiasi pagina abilitata.
Per registrare una variabile nella sessione corrente è necessario la funzione **Session_register()**, essa richiede come parametro una stringa che rappresenta il nome di una o più variabili da memorizzare nella sessione, e restituisce **true** se la registrazione ha avuto successo.

REGISTRAZIONE DI VARIABILI DI SESSIONE

```
<?php
session_start();
?>
<html>
<head>
<title> Registrazione di variabili di sessione </title>
</head>
<body>
<?php
session_register( "prodotto1" );
session_register( "prodotto2" );
$prodotto1 = "HDD IBM 20GB U2WS";
$prodotto2 = "SCANNER 64BIT EPSON";
print session_encode();
print "I prodotti sono stati registrati. Grazie";
?>
</body>
</html>
```

ACCESSO ALLE VARIABILI REGISTRATE

```
<?php
session_start();
?>
<html>
<head>
<title> Accesso alle variabili registrate </title>
</head>
<body>
<?php
print "I prodotti scelti sono:\n\n";
print "<ul><li>$prodotto1\n<li>$prodotto2\n</ul>\n";
?>
</body>
</html>
```

8.4 – Chiusura delle sessioni e rilascio delle variabili

Tramite la funzione **Session_destroy()** è possibile cancellare una sessione eliminando allo stesso tempo tutte le relative variabili di sessione. Da sottolineare che **Session_destroy()** non distrugge immediatamente le variabili di sessione registrate; queste rimangono accessibili allo script in cui è stata invocata **Session_destroy()**; le variabili possono essere eliminate anche prima del termine dello script mediante la funzione **Session_unset()**.

8.5 – Passaggio dell'ID di sessione nella stringa di interrogazione

Negli esempi precedenti, mi sono affidato ai cookie per gestire il salvataggio dell'ID di sessione da una richiesta alla successiva, questo non è il modo più sicuro per salvare lo stato perché non si hanno elementi a priori per stabilire se il browser di destinazione accetti o meno i cookie.
È possibile tuttavia passare l'ID di sessione all'interno di una stringa di interrogazione. PHP rende disponibile per questo una coppia nome/valore all'interno della costante ambientale SID.
La coppia nome/valore SID può essere inserita nei collegamenti HTML/PHP riferiti alle pagine della stessa sessione:

```
<a href="pagina.html?<? print SID; ?>Nuova Pagina</a>
```

L'ID di sessione viene in questo modo riconosciuto automaticamente nella pagina di destinazione quando viene richiamata **Session_start()**.
Se PHP4 è stato compilato sotto piattaforma Unix/Linux con l'opzione **–enable-trans-sid**, il SID viene aggiunto automaticamente a ogni collegamento presente sulle pagine. Sotto piattaforma NT invece l'opzione è già abilitata.

8.6 – Codifica e decodifica delle variabili di sessione

Può essere utile in fase di debugging degli ambienti a sessioni, accedere direttamente alle stringhe codificate con **Session_encode()**.

Session_encode() : Codifica una sessione.
Session_decode() : Decodifica una sessione, e libera le rispettive variabili.

Infatti dopo aver estratto una stringa codificata, è possibile decodificarla e ripristinare i valori con **Session_decode()**.

DECODIFICA DI UNA STRIGA DI SESSIONE

```
Session_start(); // Attivo il supporto Sessioni
Session_unset(); // elimino tutte le variabili di sessione
Foreach ( $prodotto as $p ) {
    Print "$p<br>\n";
}//Foreach
```

8.7 – Verifica della registrazione di una variabile di sessione

Per verificare la presenza di una variabile registrata all'interno di una sezione, basta invocare la funzione **Session_is_registered()**, questa funzione accetta come parametro una variabile di sessione e restituisce **true** se la variabile è registrata.

```
If ( session_is_registered( "prodotti" ) )
    Print "La variabile prodotti e registrata nella sessione";
```

9. Integrazione con MySql

9.1 – Collegamento al server MySql

Per poter usufruire del supporto di PHP a un database relazionale come MySql, si deve per prima cosa effettuare un collegamento al server MySql, tramite la funzione **Mysql_connect()**, che non richiede nessun argomento ma è possibile elencare un nome di host, un nome utente e una password, per accedere a database remoti. Se la connessione va a buon fine **Mysql_connect()**, restituisce un identificativo di connessione, con il quale continuare l'attività sul server MySql.

```
$Link = Mysql_connect ("db.newebsolutions.fs2.com:porta", "user", "pwd" );
if (! $Link )
    die ("Impossibile connettersi a Mysql" . mysql_error());
```

9.2 – Selezione di un database

Una volta stabilita una connessione con MySql è necessario scegliere un database su cui s'intende lavorare. Per selezionare un database si può utilizzare la funzione **Mysql_select_db()**, che richiede il nome del database sul quale s'intende lavorare e l'identificativo di connessione restituito da **Mysql_connect()**.
Se questo è omesso verrà inserito per default l'identificativo dell'ultimo collegamento al server MySql.
Mysql_connect_db(), restituisce **true** se il database esiste ed è possibile accedervi.

```
$Database = "Account";
Mysql_select_db( $Database, $Link )
    Or die ( "Impossibile aprire $Database );
```

9.3 – Aggiunta di dati ad una tabella

All' interno del database Scuola, è stata creata una tabella Account; in questa tabella possiamo inserire i valori di tre attributi IdUtente, Nome, Password; per far questo dobbiamo chiamare la funzione **Mysql_query()**, che richiede una stringa contenente una query SQL e, facoltativamente, un identificativo di connessione. **Mysql_query()** restituisce un valore positivo se la query ha successo; se la query contiene un errore di sintassi, o se non si possiedono i permessi di accesso al database in questione, la runzione ritorna **false**.

AGGIUNTA DI UNA RIGA A UNA TABELLA

```
<html>
<head>
<title> Aggiunta di una riga a una tabella </title></head>
<body>
<?php
$Link =  Mysql_connect( "db.newebsolutions.fs2.com:porta", "user", "pwd" );
if ( ! $Link )
    Die( "Impossibile connetersi a MySql" );
$Database = "Scuola";
$Tabella = "Account";
Mysql_select_db( $Database, $Link )
    Or Die ( "Impossibile aprire $Database );
$Query = "INSERT INTO $Tabella ( IdUtente, Nome, Psw ) values( '', 'Mauro'
'segreta' )";
Mysql_query( $Query, $Link )
    Or Die ( "Impossibile aggiungere record alla tabella $Tabella:"
    .Mysql_error() );
Mysql_close( $Link );
?></body>
</html>
```

9.4 – Acquisizione del valore di un campo a incremento automatico

Nell'esempio precedente si sono aggiunti dei record ad una tabella, senza preoccuparsi del campo IdUtente che gode della proprietà AUTO_INCREMENT definita dallo standard Sql/MySql.
Se in seguito dovesse rendersi necessario conoscere il valore di questo campo per un determinato record, sarà possibile ricavarlo tramite una query di selezione, ma se si deve ottenerlo in maniera diretta subito dopo aver effettuato un SQL INSERT, si deve utilizzare la funzione **Mysql_insert_id()** di PHP.
Questa funzione restituisce il valore di un campo che è chiave primaria a incremento automatico (contatore) dopo l'esecuzione di una query SQL INSERT. La funzione **Mysql_insert_id()** accetta come argomento, facoltativo, un identificativo del collegamento.

9.5 – Determinazione del numero di righe risultanti da una query

Tramite la funzione **Mysql_num_row()**, viene restituito il numero di righe interessate in un istruzione SQL INSERT, questa funzione richiede un identificativo di risultato, e ritorna il numero di righe. Potrebbe essere implementata anche in SQL puro, con due SELECT annidati, dove il primo SELECT esegue il COUNT delle tuple del secondo che esegue una selezione semplice.

UTILIZZO DI MYSQL_NUM_ROWS() PER DETERMINARE IL NUMERO DI RIGHE

```
<html>
<head>
<title> Utilizzo di Mysql_num_rows () </title>
</head>
<body>
<?php
$Link =  Mysql_connect("db.newebsolutions.fs2.com:porta","user","pwd");
if ( ! $Link )
    Die( "Impossibile connetersi a MySql" );
$Database = "Scuola";
$Tabella = "Account";
Mysql_select_db( $Database, $Link )
    Or Die ( "Impossibile aprire $Database );
$Risultato = Mysql_query( "SELECT * FROM $Tabella" );
$Num_righe = Mysql_num_rows( $Risultato );
Print "Ci sono $Num_righe in $Tabella<P>";
Mysql_close( $Link );
?>
</body>
</html>
```

9.6 – Accesso a un insieme di informazioni

Dopo aver eseguito un'istruzione SQL SELECT e possibile effettuare un ciclo per accedere a turno a ognuna delle righe trovate. PHP gestisce un puntatore interno che tiene conto della posizione all'interno di un insieme di informazioni, via via che ogni riga viene esaminata, il puntatore passa alla riga successiva.
È possibile ottenere un array contenente i campi presenti in ognuna delle righe trovate, mediante le funzione **Mysql_fetch_row()** e **Mysql_fetch_array()**.

Queste funzioni richiedono un identificativo di risultato e restituiscono un array associativo per chiave, dove la chiave è il nome dell'attributo (campo), altrimenti restituisce **false.**

ELENCO DI TUTTE LE RIGHE E TUTTI I CAMPI DI UNA TABELLA

```
<html>
<head>
<title> Elenco di tutte le righe e tutti i campi di una tabella </title>
</head>
<body>
<?php
$Link =  Mysql_connect( “db.newebsolutions.fs2.com:porta“,”user”,“pwd” );
if ( ! $Link )
    Die( "Impossibile connetersi a MySql" );
$Database = “Scuola”;
$Tabella = “Account”;
Mysql_select_db( $Database, $Link )
    Or Die ( “Impossibile aprire $Database );
$Risultato = Mysql_query( "SELECT * FROM $Tabella" );
$Num_righe = Mysql_num_rows( $Risultato );
Print "Ci sono $Num_righe in $Tabella<P>";
Print "<table border=1>\n";
While ( $Riga = Mysql_fetch_row( $Risultato ) ) {
    Print "<tr>\n";
    Foreach ( $Riga as $Campo )
        Print "\t<td>$Campo</td>\n";
    Print "</tr>\n";
}//WHILE
print "</table>\n";
mysql_close( $Link );
?>
</body>
</html>
```

9.7 – Modifica dei dati

È possibile modificare la base di dati utilizzando la funzione **Mysql_query()**, insieme a un’istruzione SQL UPDATE. Per sapere se i dati contenuti in una tabella sono stati modificati, è necessario utilizzare **Mysql_effected_rows()**, che accetta un identificativo di connessione, questa funzione può essere utilizzata con qualsiasi query SQL in grado di modificare la base di dati.

UTILIZZO DI MYSQL_QUERY() PER MODIFICARE LE RIGHE DI UN DATABASE

```
<html>
<head>
<title> Utilizzo di MySql query </title>
</head>
<body>
<?php
$Link =  Mysql_connect(“db.newebsolutions.fs2.com:porta“,”user”,“pwd”);
if ( ! $Link )
    Die( "Impossibile connetersi a MySql" );
$Database = “Scuola”;
$Tabella = “Account”;
Mysql_select_db( $Database, $Link )
    Or Die ( “Impossibile aprire $Database );
$Risultato = Mysql_query( "SELECT * FROM $Tabella" );
if ( Isset( $Utente ) && Isset( $Id ) )  {
    $Query = "UPDATE $Tabella SET Utente = '$Utente' where IdUtente=$Id";
    $Risultato = Mysql_query( $Query );
    If ( ! $Risultato )
        Die ("Impossibile aggiornare: ".mysql_error());
    Print "<h1>Tabella aggiornata".Mysql_affected_rows()."record
aggiornati</h1><p>";
}//THEN
```

```
?>
<form action="<? print $PHP_SELF ?>" method="POST">
<select name="IdUtente">
<?
$Risultato = Mysql_query( "SELECT Utente, IdUtente  FROM $Tabella" );
While( $Righe = Mysql_fetch_object( $Risultato ) ) {
    Print "<OPTION VALUE=\"$Riga->Utente\"";
    If ( Isset($Id) && $Id == $Riga->IdUtente )
        print " SELECTED";
        print "> $Riga->Utente\n";
    }//THEN
Mysql_close( $link );
?>
</select>
<input type="text" name="Utente">
</form>
</body>
</html>
```

9.8 – Elenco di database, e tabelle

Tramite la funzione **Mysql_list_dbs()**, e possibile ottenere un elenco di tutti i database disponibili in un server MySql, la funzione sopracitata accetta come argomenti un identificativo di collegamento, facoltativo, e restituisce un identificativo di risultato che può essere utilizzato per visualizzare i database disponibili. Allo scopo, è possibile richiamare la funzione **Mysql_tablename()** che per ognuno dei database trovati, accetta due argomenti: un puntatore al risultato e l'indice di un database; restituisce il nome del database referenziato in base all'indice. Lo stesso discorso con opportune condizioni vale anche per la funzione **Mysql_list_table()**, che restituisce un vettore indicizzato contenente i nome delle tabelle di un determinato database.

ELENCO DEI DATABASE DISPONIBILI IN UNA CONNESSIONE

```
<html>
<head>
<title> Elenco dei database disponibili in una connessione </title>
</head>
<body>
<?php
$Link =  Mysql_connect("db.newebsolutions.fs2.com:porta","user","pwd");
if ( ! $Link )
    Die( "Impossibile connetersi a MySql" );
$Db_Lista = Mysql_list_dbs( $Link );
$Num_Righe = Mysql_num_rows( $Db_Lista );
For( $x = 0; $x < $Num_Righe; $x++ )
    print Mysql_tablename( $Db_Lista, $x )."<br>";
Mysql_close( $Link );
?>
</body>
</html>
```