Php: un'introduzione

Cos'è PHP

A metà degli anni Novanta il Web era ancora formato in gran parte da pagine statiche, cioè da documenti HTML il cui contenuto non poteva cambiare fino a quando qualcuno non interveniva manualmente a modificarlo.

Con l'evoluzione di Internet, però, si cominciò a sentire l'esigenza di rendere dinamici i contenuti, cioè di far sì che la stessa pagina fosse in grado di proporre contenuti diversi, personalizzati in base alle preferenze degli utenti, oppure estratti da una base di dati (database) in continua evoluzione.

PHP nasce nel 1994, ad opera di Rasmus Lerdorf, come una serie di macro la cui funzione era quella di facilitare ai programmatori l'amministrazione delle home page personali: da qui trae origine il suo nome, che allora significava appunto Personal Home Page.
Oggi PHP è conosciuto come PHP: Hypertext Preprocessor, ed è un linguaggio completo di scripting, sofisticato e flessibile, che può girare praticamente su qualsiasi server Web, su qualsiasi sistema operativo

Perché scegliere PHP?

I motivi sono più di uno, e non è facile stabilire quale sia quello principale. Tuttavia, è sicuramente molto importante il fatto che si tratta di un prodotto open source e gratuito, quindi a disposizione di tutti. E' importante notare che il vantaggio garantito dall'Open Source non è limitato al non dover sborsare denaro ma si estende oltre: infatti tutti i progetti open source molto seguiti generano delle vaste comunità di utenti alle quali si può fare riferimento per trovare facilmente una soluzione ai propri problemi.

La portabilità è un altro fattore determinante: PHP può girare su moltissime piattaforme, sia per quanto riguarda i sistemi operativi che i server web. Il suo "habitat naturale" è un server Apache su una macchina Linux, ma può girare tranquillamente anche su IIS o perfino su un modesto Windows 98 con Personal Web Server.

Il motore Zend, introdotto con PHP 4, ha fornito un formidabile aumento della velocità di esecuzione, portando PHP sugli stessi livelli di ASP, ed in alcuni casi anche oltre.

PHP è anche facile da imparare: la sua sintassi deriva dal C, e quindi sarà familiare a chi è già programmatore ed arriva da questo genere di linguaggi.

PHP è in grado di interagire con una vasta quantità di database server, permettendoci con pochi comandi di leggere e scrivere i dati su di essi, e di realizzare così il Web dinamico.

PHP e HTML

PHP è un linguaggio la cui funzione fondamentale è quella di produrre codice HTML, che è quello dal quale sono formate le pagine web. Siccome però PHP è un linguaggio di programmazione, abbiamo la possibilità di analizzare diverse situazioni (l'input degli utenti, i dati contenuti in un database) e di decidere, di conseguenza, di produrre codice HTML condizionato ai risultati dell'elaborazione. Questo è, in parole povere, il Web dinamico.

Il codice php deve essere compreso fra appositi tag di apertura e di chiusura, che sono i seguenti:
<?php //tag di apertura
?> //tag di chiusura

Tutto ciò che è contenuto fra questi tag deve corrispondere alle regole sintattiche del PHP, ed è codice che sarà eseguito dall'interprete e non sarà inviato al browser. Per generare il codice da inviare al browser abbiamo due costrutti del linguaggio, che possiamo considerare equivalenti, che sono: print() e echo().

PHP e HTML: un primo esempio

```
<HTML>
<HEAD>
<TITLE>Pagina di prova in PHP</TITLE>
</HEAD>
<BODY>
<?php
print("Buongiorno a tutti!<br>\n");
echo("E' una bellissima giornata");
?>
</BODY>
</HTML>
```

```
<HTML>
<HEAD>
<TITLE>Pagina di prova in PHP</TITLE>
</HEAD>
<BODY>
Buongiorno a tutti!<br>
E' una bellissima giornata
</BODY>
</HTML>
```


PHP e HTML: una precisazione

```
<?php
print("prima riga\n");
print("seconda riga<br>");
print("terza riga");
?>
```

Questo codice php produrrà il seguente codice html:

```
prima riga
seconda riga<br>terza riga
```

mentre l'utente, sul browser, leggerà:

prima riga seconda riga
terza riga

I commenti

Abbiamo tre diverse possibilità per posizionare i commenti all'interno del nostro codice: la prima è l'uso dei commenti in stile C++, caratterizzati da due barre:

```
// Commento in stile C++
```

La seconda sono i commenti in stile Perl, contraddistinti dall'uso del cancelletto:

```
# Commento in stile Perl
```

Il terzo tipo di commento (in stile C) inizia con una barra seguita da un asterisco, e termina con un asterisco seguito da una barra:

```
print ('Buonasera<br>'); /* Questo è un commento in stile C,
che può occupare più righe ma deve essere chiuso
con l'apposito simbolo */
```


Le variabili

In PHP possiamo scegliere il nome delle variabili usando lettere, numeri ed il trattino di sottolineatura, o underscore (_). Il primo carattere del nome deve essere però una lettera o un underscore (non un numero). Dobbiamo inoltre ricordarci che il nome delle variabili è sensibile all'uso delle maiuscole e delle minuscole: di conseguenza, se scriviamo due volte un nome di variabile usando le maiuscole in maniera differente, per PHP si tratterà di due variabili distinte! Nello script PHP il nome delle variabili è preceduto dal simbolo del dollaro ($).

PHP non richiede che le variabili vengano dichiarate prima del loro uso. Possiamo quindi permetterci di riferirci ad una variabile direttamente con la sua valorizzazione:

```
$a = 5;
```

Vediamo ora qualcosa di leggermente più complesso eseguito con le variabili:

```
$a = 9; $b = 4; $c = $a * $b;
```

Con questo codice abbiamo valorizzato tre variabili: 'a', alla quale stavolta abbiamo dato il valore 9; 'b', a cui abbiamo assegnato il valore 4; e infine 'c', alla quale abbiamo detto che dovrà assumere il valore del prodotto di 'a' e 'b'. Evidentemente, dopo l'esecuzione del codice 'c' varrà 36.

Le variabili: una precisazione

Negli esempi precedenti abbiamo visto l'inizializzazione delle variabili. In realtà, possiamo riferirci ad una variabile anche senza che sia stata inizializzata. Ad esempio, supponendo che nel nostro script non sia stata valorizzata nessuna variabile 'z', potremmo avere un'istruzione di questo genere:

```
print $z;
```

Questo codice non produrrà alcun output, in quanto la variabile 'z' non esiste. C'è però da notare che, nonostante lo script funzioni regolarmente e proceda con l'elaborazione delle istruzioni successive, un'istruzione di questo tipo non viene considerata corretta da PHP, che produrrà una segnalazione di errore di tipo 'notice' per farci notare che abbiamo usato una variabile non inizializzata. Gli errori di tipo 'notice' sono gli errori di livello più basso, cioè meno gravi, che normalmente non vengono mostrati da PHP, ma che possiamo abilitare attraverso il file di configurazione php.ini.

Un errore di questo genere in effetti non compromette il buon funzionamento dello script, che infatti viene eseguito regolarmente; però potrebbe essere ugualmente indice di un qualche errore commesso da chi ha scritto il codice.

Le variabili dinamiche

In qualche situazione, infatti, può presentarsi la necessità di utilizzare delle variabili senza sapere a priori come si chiamano. In questi casi, il nome di queste variabili sarà contenuto in ulteriori variabili. Facciamo un esempio: col codice seguente stamperemo a video il contenuto delle variabili 'pippo', 'pluto' e 'paperino':

```
$pippo = 'gawrsh!'; $pluto = 'bau!';
$paperino = 'quack!'; $nome = 'pippo';
print ($$nome.'<br>');
$nome = 'pluto';
print ($$nome.'<br>');
$nome = 'paperino';
print ($$nome.'<br>');
```

**Il risultato sul browser sarà 'gawrsh!', 'bau!' e 'quack!', ciascuno sulla propria riga (infatti ogni istruzione print crea il tag HTML
 che indica al browser di andare a capo; vedremo più avanti che il punto serve a concatenare i valori che vengono stampati). Il doppio segno del dollaro ci permette infatti di usare la variabile 'nome' come contenitore del nome della variabile di cui vogliamo stampare il valore. In pratica, è come se avessimo detto a PHP: "stampa il valore della variabile che si chiama come il valore della variabile 'nome'".**

I tipi di variabile: valore booleano

Valore booleano. Le variabili booleane sono le più semplici: il loro valore può essere TRUE o FALSE (vero o falso). Vediamo un rapido esempio:

```
$vero = TRUE;
$falso = FALSE;
```


I tipi di variabile: intero

Intero. Un numero intero, positivo o negativo, il cui valore massimo (assoluto) può variare in base al sistema operativo su cui gira PHP, ma che generalmente si può considerare, per ricordarlo facilmente, di circa 2 miliardi (2 elevato alla 31esima potenza).

```
$int1 = 129;
$int2 = -715;
$int3 = 5 * 8; //$int3 vale 40
```

I tipi di variabile: virgola mobile

Virgola mobile. Un numero decimale (a volte citato come "double" o "real"). Attenzione: per indicare i decimali non si usa la virgola, ma il punto! Anche in questo caso la dimensione massima dipende dalla piattaforma. Normalmente comunque si considera un massimo di circa 1.8e308 con una precisione di 14 cifre decimali. Si possono utilizzare le seguenti sintassi:

```
$vm1 = 4.153; // 4,153
$vm2 = 3.2e5; // 3,2 * 10^5, cioè 320.000
$vm3 = 4E-8; // 4 * 10^-8, cioè 4/100.000.000 = 0,00000004
```


I tipi di variabile: stringa (1)

Le stringhe possono essere espresse in tre maniere:

- **Delimitate da apici (singoli)**
- **Delimitate da virgolette (doppie)**
- **Con la sintassi heredoc**

Le stringhe delimitate da apici sono la forma più semplice, consigliata quando all'interno della stringa non vi sono variabili di cui vogliamo ricavare il valore:

```
$frase = 'Anna disse: "Ciao a tutti!" ma nessuno rispose';
print $frase; /*stampa la frase: Anna disse: "Ciao a tutti!" ma nessuno rispose*/
```

Le virgolette ci consentono di usare le stringhe in una maniera più sofisticata, in quanto, se all'interno della stringa delimitata da virgolette PHP riconosce un nome di variabile, lo sostituisce con il valore della variabile stessa (si dice in questo caso che la variabile viene risolta).

```
$nome = 'Anna';
print "$nome è simpatica... a pochi"; /*stampa: Anna è simpatica... a pochi*/
print '$nome è simpatica... a pochi'; /*stampa: $nome è simpatica... a pochi*/
```

Come vedete, l'uso delle virgolette permette di risolvere le variabili, mentre con gli apici i nomi di variabile rimangono... invariati.

I tipi di variabile: stringa (2)

```
print 'Torniamo un\'altra volta'; //stampa: Torniamo un'altra volta
print "Torniamo un'altra volta"; //stampa: Torniamo un'altra volta
print "Torniamo un\'altra volta"; //stampa: Torniamo un\'altra volta
print 'Torniamo un'altra volta'; /*causa un errore, perché l'apostrofo viene scambiato per l'apice di chiusura*/
print 'Anna disse "Ciao" e se ne andò'; /*stampa: Anna disse "Ciao" e se ne andò*/
print "Anna disse \"Ciao\" e se ne andò"; /*stampa: Anna disse "Ciao" e se ne andò*/
print 'Anna disse \"Ciao\" e se ne andò'; /*stampa: Anna disse \"Ciao\" e se ne andò*/
print "Anna disse "Ciao" e se ne andò"; //errore

print ("Questo: \"\\\" è un backslash"); /*stampa: Questo: "\" è un backslash*/
print ('Questo: \'\\\' è un backslash'); /*stampa: Questo: '\' è un backslash*/
print ("Questo: '\' è un backslash"); /*stampa: Questo: '\' è un backslash*/
print ("Questo: '\\' è un backslash"); /*stampa: Questo: '\' è un backslash
```

Analizziamo gli ultimi quattro esempi: il primo backslash fa l'escape del primo paio di virgolette; il secondo backslash fa l'escape del terzo, che quindi viene incluso nella stringa; il quarto fa l'escape del secondo paio di virgolette. Il secondo esempio equivale al primo, con l'uso degli apici al posto delle virgolette.
Negli ultimi due casi non è necessario fare l'escape del backslash, in quanto il backslash che vogliamo stampare non può essere scambiato per un carattere di escape (infatti vicino ad esso ci sono degli apici, che in una stringa delimitata da virgolette non hanno bisogno di escape). Di conseguenza, fare o non fare l'escape del backslash in questa situazione è la stessa cosa, e difatti i due esempi forniscono lo stesso risultato.

I tipi di variabile: stringa (3)

Passiamo ad esaminare l'ultimo modo di rappresentare le stringhe: la sintassi heredoc. Questa ci consente di delimitare una stringa con i caratteri <<< seguiti da un identificatore (in genere si usa 'EOD', ma è solo una convenzione: è possibile utilizzare qualsiasi stringa composta di caratteri alfanumerici e underscore, di cui il primo carattere deve essere non numerico: la stessa regola dei nomi di variabile). Tutto ciò che segue questo delimitatore viene considerato parte della stringa, fino a quando non viene ripetuto l'identificatore seguito da un punto e virgola. Attenzione: l'identificatore di chiusura deve occupare una riga a sé stante, deve iniziare a colonna 1 e non deve contenere nessun altro carattere (nemmeno spazi vuoti) dopo il punto e virgola.

```
$nome = "Paolo";
$stringa = <<<EOD
Il mio nome è $nome
EOD;
print $stringa;
```

Questo codice stamperà 'Il mio nome è Paolo'. Infatti la sintassi heredoc risolve i nomi di variabile così come le virgolette. Rispetto a queste ultime, con questa sintassi abbiamo il vantaggio di poter includere delle virgolette nella stringa senza farne l'escape.

I tipi di variabile: array

Possiamo considerare un array (vettore) come una variabile complessa, che contiene cioè non un solo valore, ma una serie di valori, ciascuno dei quali caratterizzato da una chiave, o indice. Facciamo un primo esempio, definendo un array composto di cinque valori:

```
$colori = array('bianco', 'nero', 'giallo', 'verde', 'rosso');
```

A questo punto ciascuno dei nostri cinque colori è caratterizzato da un indice numerico, che PHP assegna automaticamente a partire da 0. L'indice viene indicato fra parentesi quadre dietro al nome dell'array:

```
print $colori[1]; //stampa 'nero'
print $colori[4]; //stampa 'rosso'
```


Eliminare una variabile

In alcune situazioni ci può capitare di avere la necessità di eliminare una variabile: in questo caso PHP ci mette a disposizione l'istruzione unset():

```
unset($variabile);
```

Dopo l'istruzione unset, sarà come se la variabile non fosse mai esistita.

Gli operatori

```
$nome = 'Giorgio';
$a = 5;
$b = $a;
$a = 3 + 7; //addizione
$b = 5 - 2; //sottrazione
$c = 9 * 6; //moltiplicazione
$d = 8 / 2; //divisione
$e = 7 % 4; /*modulo (il modulo è il resto della divisione, quindi in questo caso 3)*/

$nome = 'pippo';
$stringa1 = 'ciao ' . $nome; //$stringa1 vale 'ciao pippo'

$a = $a + 10; //il valore di $a aumenta di 10

$x += 4; //incrementa $x di 4 (equivale a $x = $x + 4)
$x -= 3; //decrementa $x di 3 (equivale a $x = $x - 3)
$x /= 5; //equivale a $x = $x / 5
$x *= 4; //equivale a $x = $x * 4
$x %= 2; //equivale a $x = $x % 2

$a++; /*incrementa di 1: equivale ad $a = $a + 1, o $a += 1*/
$a--; //decrementa di 1: equivale ad $a = $a - 1, o $a -= 1
```


Gli operatori di confronto

Questi operatori sono:

- **== : uguale**
- **!= : diverso**
- **=== : identico (cioè uguale e dello stesso tipo: ad esempio per due variabili di tipo intero)**
- **> : maggiore**
- **>= : maggiore o uguale**
- **< : minore**
- **<= : minore o uguale**

Gli operatori logici

Con gli operatori logici possiamo combinare più valori booleani, oppure negarne uno (nel caso di NOT). Questi valori sono:

- **or: valuta se almeno uno dei due operatori è vero; si può indicare con 'Or' oppure '||'**
- **and: valuta se entrambi gli operatori sono veri; si indica con 'And' o '&&'**
- **xor: viene chiamato anche 'or esclusivo', e valuta se uno solo dei due operatori è vero: l'altro deve essere falso; si indica con 'Xor'**
- **not: vale come negazione e si usa con un solo operatore: in pratica è vero quando l'operatore è falso, e viceversa; si indica con '!'**

Cos'è falso in PHP?

PHP considera come falsi:

- **il valore numerico 0, nonché una stringa che contiene '0'**
- **una stringa vuota**
- **un array con zero elementi**
- **un valore NULL, cioè una variabile che non è stata definita o che è stata eliminata con unset(), oppure a cui è stato assegnato il valore NULL stesso**

Qualsiasi altro valore, per PHP è un valore vero. Quindi qualsiasi numero, intero o decimale purché diverso da 0, qualsiasi stringa non vuota, se usati come espressione condizionale saranno considerati veri. Quindi:

```
if (5) {
  print "ciao Luca!";
}
```

stamperà sempre il saluto.

If

```
if (CONDIZIONE1) {
        ISTRUZIONE1
} elseif (CONDIZIONE2) {
        ISTRUZIONE2
} else {
        ISTRUZIONE3
}
```

Ad esempio:

```
if ($nome == 'Luca') {
  print "bentornato Luca!";
} elseif ($cognome == 'Verdi') {
  print "Buongiorno, signor Verdi";
} else {
  print "ciao $nome!";
}
```


Switch

```
switch ($nome) {
  case 'Luca':
  case 'Giorgio':
  case 'Franco':
    print "Ciao, vecchio amico!";
    break;
  case 'Mario':
    print "Ciao, Mario!";
    break;
  case 'Paolo':
    print "Finalmente, Paolo!";
    break;
  default:
    print "Benvenuto, chiunque tu sia";
}
```

Abbiamo previsto un unico messaggio per il caso in cui la variabile $nome valga 'Luca', 'Giorgio' o 'Franco'.

L'operatore ternario

```
$tipologia = ($altezza >= 180) ? 'alto' : 'normale';
```

equivale a scrivere

```
if ($altezza >= 180)
  $tipologia = 'alto' ;
else
  $tipologia = 'normale';
```


I cicli

```
for ($mul = 1; $mul <= 10; $mul++) {
  $ris = 5 * $mul;
  print("5 * $mul = $ris<br>");
}

$mul = 1;
while ($mul <= 10) {
  $ris = 5 * $mul;
  print("5 * $mul = $ris<br>");
  $mul++;
}

$mul = 1;
do {
  $ris = 5 * $mul;
  print("5 * $mul = $ris<br>");
  $mul++;
} while ($mul <= 10)
```

Per il trattamento degli array:

```
foreach ($arr as $chiave => $valore) {
  ....istruzioni da ripetere....
}
```


LOONEY TUNES
That's all Folks!
A WARNER BROS. CARTOON