Linguaggi di Programmazione per il Web – Parte 1

PHP – Hypertext Preprocessor

Introduzione al linguaggio

Autore

Prof. **Rio Chierego**
riochierego@libero.it

Siti Utili

http://www.riochierego.it/mobile

http://www.html.it/

http://www.mrwebmaster.it

http://www.php.net/

http://php.html.it/

PHP – Hypertext Preprocessor

PHP è conosciuto come **PHP:Hypertext Preprocessor** ed **è un linguaggio completo di scripting** che permette di generare pagine web dinamicamente.

Sofisticato e flessibile **può girare praticamente su qualsiasi server web e su qualsiasi sistema operativo e consente di interagire con qualsiasi tipo di database.**

Si può utilizzare per i più svariati tipi di progetti, **dalla semplice home page dinamica fino al grande portale o sito di e-commerce**

Nasce alla fine del **1994** ad opera di **Rasmus Lerdorf** attualmente è arrivato alla versione 5.4.12

Funzionamento del PHP: PHP e JS

Confrontiamo **PHP** con un altro linguaggio di scripting molto diffuso sul Web, cioè **JavaScript**, che di solito viene usato come linguaggio **"lato client".**

JavaScript non viene eseguito dal server, ma dal browser dell'utente (il client, appunto). JavaScript ci consente di eseguire operazioni che riguardano il sistema dell'utente. Ci permette anche di avere un'interazione con l'utente. Per svolgere tutti questi compiti, JavaScript deve essere eseguito sul sistema dell'utente: per questo il codice JavaScript viene spedito al browser insieme al codice html.

Quindi l'utente ha la possibilità di visualizzarlo, contrariamente a ciò che accade con PHP.

PHP svolge principalmente la funzione di 'creare' il codice della pagina che viene spedita dal web server all'utente: di conseguenza, così come può creare codice HTML, allo stesso modo può creare codice JavaScript. Questo significa che PHP ci può permettere, ad esempio, di decidere se ad un utente dobbiamo spedire il codice JavaScript che apre una nuova finestra, oppure no. **In pratica, quindi, lavorando sul lato server abbiamo il controllo anche del lato client.**

Come riconoscere un file PHP

Come fa il server a sapere quando una pagina contiene codice PHP che deve essere eseguito prima dell'invio al browser?

Si basa, semplicemente sull'estensione delle pagine richieste.

Se ad esempio abbiamo *pagina1* con estensione **.html** o **.htm** mentre *pagina2* ha l'estensione **.php**: il server sa che nel secondo caso deve eseguire PHP, mentre nel primo può spedire il file così com'è. Le estensioni che di solito permettono di riconoscere file PHP sono: ***.php, .php3, .php4***

E' comunque possibile configurare il server in modo da assegnare al PHP qualsiasi altra estensione: spesso viene usata ***.phtml***

Perché scegliere PHP

- ***è un prodotto open source***, gratuito e continuamente migliorato da chi lo utilizza
- ***è portabile***: 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 girare tranquillamente su NT o perfino su un modesto Windows 98. Questa sua capacità gli dà un grosso vantaggio su la sua maggior rivale, cioè la tecnologia ASP (ora .NET), che è limitata ai sistemi operativi Microsoft.
- ***è veloce***: con PHP 4 la velocità di esecuzione è agli stessi livelli di ASP, ed in alcuni casi anche oltre.
- ***è facile da imparare***: la sua sintassi deriva dal C (quindi familiare a chi è già programmatore). Allo stesso tempo, però, PHP risulta facilmente digeribile anche da chi è a digiuno di informatica, grazie ad alcune sue caratteristiche di flessibilità
- ***ha la possibilità di connettersi a svariati database*** server 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.
Una pagina PHP può essere quindi composta da codice PHP (ovviamente) mescolato a codice HTML, in fase di escuzione l'interprete PHP convertirà le istruzioni php in codice html restituendo una pagina valida html.
Per essere riconosciuto e interpretato Il codice PHP deve essere compreso fra appositi tag di apertura e di chiusura, che sono i seguenti:

<?php ← tag di apertura (nota bene SENZA ALCUNO SPAZIO)
…… *//altre istruzioni PHP*
?> ← tag di chiusura (nota bene SENZA ALCUNO SPAZIO)

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.

PHP e HTML: la funzione echo

Per generare il codice da inviare al browser utilizziamo la funzione ***echo ( )***
Vediamo un primo esempio:

```
<HTML>
<HEAD><TITLE>Pagina di prova in PHP</TITLE></HEAD>
<BODY>
<?php
echo "Oggi<BR> \ne' una bellissima giornata";
?>
</BODY>
</HTML>
```

Questo codice PHP produce **un file HTML** il cui contenuto sarà:

```
<HTML>
<HEAD><TITLE>Pagina di prova in PHP</TITLE></HEAD>
<BODY>
Oggi<BR>
e' una bellissima giornata
</BODY>
</HTML>
```

(vedi esempio **PPT-1-esempio-1.php**)

PHP e HTML

Facciamo caso ad un dettaglio: nelle istruzioni in cui stampavamo “Oggi è una bellissima giornata”, abbiamo inserito, dopo il **
,** il simbolo **\n**.

Questo simbolo ha una funzione abbastanza importante nello scripting che serve più che altro per dare **leggibilità al codice HTML** che stiamo producendo.

Infatti PHP, quando trova questa combinazione di caratteri fra virgolette, li trasforma in un carattere di **ritorno a capo**: questo ci permette di controllare l'impaginazione del nostro codice HTML.

Bisogna però stare molto attenti a non confondere il codice HTML con il layout della pagina che l'utente visualizzerà sul browser: infatti, sul browser **è solo il tag
** che forza il testo ad andare a capo.

PHP e HTML

Esempio:

```
<?php
echo "prima riga\n";
echo "seconda riga<br>";
echo "terza riga";
?>
```

Questo codice PHP produrrà il seguente codice HTML:

```
prima riga
seconda riga<BR>terza riga
```

mentre l'utente, sul browser, leggerà:

```
prima rigaseconda riga
terza riga
```

(vedi esempio **PPT-1-esempio-2.php**)

Sintassi generale del linguaggio

Tag per identificare il linguaggio PHP all'interno dell' HTML

<?php … ?>

Terminatore di riga

;

I commenti

// Commento in stile C++

 # Commento in stile Perl

/ 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).

Inoltre il nome delle variabili è sensibile all'uso delle maiuscole e delle minuscole.

Nello script PHP il nome delle variabili è preceduto dal simbolo del dollaro **$**.

PHP ha una caratteristica che lo rende molto più flessibile rispetto ad altri linguaggi di programmazione: non richiede, infatti, che le variabili vengano dichiarate prima del loro uso.

Possiamo quindi permetterci di riferirci ad una variabile direttamente con la sua valorizzazione:

$a = 5;

Le variabili

Vediamo ora qualcosa di leggermente più complesso:

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

Con questo codice abbiamo valorizzato tre variabili: 'a' con il valore 9; 'b' con il valore 4; e 'c‘ che dovrà assumere il valore del prodotto di 'a' e 'b'. Evidentemente, dopo l'esecuzione del codice 'c' varrà 36.

```
echo $z;
```

Questo codice non produrrà alcun output, in quanto la variabile 'z' non esiste. Lo script funziona regolarmente e procede con l'elaborazione delle istruzioni successive però un'istruzione di questo tipo non viene considerata corretta da PHP.

Le variabili dinamiche

In qualche situazione, può presentarsi la necessità di utilizzare delle variabili senza sapere a priori come si chiamano.

Facciamo un esempio: col codice seguente stamperemo a video il contenuto delle variabili 'pluto' e 'paperino':

```
$pluto = 'bau!';
$paperino = 'quack!';
$nome = 'pluto';
echo $$nome;
$nome = 'paperino';
echo $$nome;
```

(vedi esempio **PPT-1-esempio-3.php**)

Il risultato sul browser sarà **bau!** e **quack!.**

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. Le variabili booleane sono le più semplici: il loro valore può essere **TRUE** o **FALSE** (vero o falso) (N.B. scritte anche in minuscolo o con iniziale maiuscola)

Vediamo un rapido esempio:

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

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
```

Virgola mobile. Un numero decimale (detto anche "real"). Anche in questo caso la dimensione massima dipende dalla piattaforma. Si considera un massimo di circa **1.8e308** con una precisione di 14 cifre decimali.

```
$vm1 = 4.153; // ossia 4,153
```

I tipi di variabile

Stringa. Una stringa è un qualsiasi insieme di caratteri, senza limitazione. Le stringhe possono essere espresse in due maniere:

- Delimitate da apici (singoli)
- Delimitate da virgolette (doppie)

Le stringhe delimitate da apici sono la forma migliore:

```
$frase = 'Anna disse: "Ciao!" ma nessuno rispose';
echo $frase;
```

```
$frase2 = "Anna disse:\'Ciao!\' ma nessuno rispose";
echo $frase2;
```

I tipi di variabile

Una stringa può contenere a sua volta un apice o un paio di virgolette, dobbiamo far capire però a PHP che quel carattere fa parte della stringa e non è il suo delimitatore.
Si usa il cosiddetto 'carattere di escape', cioè la barra rovesciata **(backslash: \).**

```
echo 'Torniamo un\'altra volta';      //Torniamo un'altra volta
echo "Torniamo un\'altra volta";      //Torniamo un\'altra volta
echo 'Torniamo un'altra volta';
/*causa un errore, perché l'apostrofo viene scambiato per l'apice di chiusura*/
```

```
echo 'Anna disse "Ciao" e se ne andò';        /*Anna disse "Ciao" e se ne andò*/
echo "Anna disse \"Ciao\" e se ne andò";      /*Anna disse "Ciao" e se ne andò*/
echo 'Anna disse \"Ciao\" e se ne andò';      /*Anna disse \"Ciao\" e se ne andò*/
echo "Anna disse "Ciao" e se ne andò";        // errore
```

Il backslash viene usato anche come 'escape di sé stesso', nei casi in cui vogliamo esplicitamente includerlo nella stringa:

```
echo "questo carattere: \\ e' il backslash";
```

(vedi esempio **PPT-1-esempio-4.php**)

I tipi di variabile

L'ultimo modo di rappresentare le stringhe: la sintassi **heredoc**, poco utilizzata se non in situazioni nelle quali è necessario specificare stringhe molto lunghe.

Consiste nel 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 sè 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;
echo $stringa;
```

(vedi esempio **PPT-1-esempio-5.php**)

I tipi di variabile

Array. Possiamo considerare un array (vettore) come una variabile complessa, che non contiene 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 dopo il nome dell'array:

echo $colori[1]; //stampa 'nero'
echo $colori[4]; //stampa 'rosso‘ (vedi esempio **PPT-1-esempio-6.php**)

Gli array

PHP gestisce **un unico** tipo di array, le cui chiavi possono essere numeriche o associative:

$colori = array ('bianco','nero','giallo','verde','rosso');
In questo modo viene generato **un array con chiavi numeriche (partenza da 0)**

$colori = array (1 => 'bianco','nero','giallo','verde','rosso');
In questo modo viene generato **un array con chiavi numeriche (partenza da 1)**

$colori = array ('primo' => 'bianco', 'secondo' => 'nero', …);
In questo modo viene generato **un array con chiavi associative**

$colori = array (1 => 'bianco', 'secondo' => 'nero', …);
In questo modo viene generato **un array con chiavi miste**

Aggiungere valori ad un array:

$colori[] = 'blu'; //valido solo se l'array ha chiavi numeriche
$colori[7] = 'arancio';
$colori['ultimo'] = 'viola'; (vedi esempio **PPT-1-esempio-7.php**)

La funzione unset()

La funzione ***unset()*** permette di eliminare una variabile:

```
$pippo = 5;                 //assegna a pippo il valore 5
$pippo = 0;                 //azzera il contenuto di pippo
unset ($pippo);             //elimina la variabile $pippo
```

In caso di array occorre fare attenzione perché:

```
$colori = array('bianco','nero','blu','giallo');
unset ($colori[0]);         //elimina solo la posizione 0 cioè 'bianco'
unset ($colori);            //elimina tutto l'array
```

Gli operatori

Assegnazione:

```
$nome = 'Giorgio';
$nome_app = $nome;
```

Operatori aritmetici:

```
$a = 3 + 7;      //addizione
$b = 5 - 2;      //sottrazione
$c = 9 * 6;      //moltiplicazione
$d = 8 / 2;      //divisione
$e = 7 % 4;      //modulo
```

Concatenare le stringhe:

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

Gli operatori

Spesso è necessario fare operazioni che modificano il valore di una variabile:
$a = $a+10; *//il valore di $a aumenta di 10*

Un risultato del genere si può ottenere anche con gli operatori di assegnazione combinati:
$x += 4; *//incrementa $x di 4 (equivale a $x = $x + 4)*
$x -= 3; *//decrementa $x di 3 (equivale a $x = $x - 3)*
$x .= $a; *//concatena $x con $a (equivale a $x = $x . $a)*
$x /= 5; *//equivale a $x = $x / 5*
$x *= 4; *//equivale a $x = $x * 4*
$x %= 2; *//equivale a $x = $x % 2*

Per gli incrementi di un unità ci sono operatori ancora più sintetici:
$a++; *//postfisso prima usa $a e poi la incrementa di 1: equivale ad $a = $a + 1, o $a += 1*
$a--; *//postfisso prima usa $a e poi la decrementa di 1: equivale ad $a = $a - 1, o $a -= 1*

++$a; *//prefisso prima incrementa di 1 $a e poi la usa*
--$a; *//prefisso prima decrementa di 1 $a e poi la usa*

Gli operatori di confronto

Gli operatori di confronto: ci permettono, effettuando dei confronti fra valori, di prendere delle decisioni. Quando utilizziamo gli operatori di confronto, confrontiamo i due valori posti a sinistra e a destra dell'operatore stesso. Il risultato di questa operazione sarà, ogni volta, vero (TRUE) o falso (FALSE). Questi operatori sono:

== //uguale (n.b. anche di tipi diversi)
!= //diverso
> //maggiore
>= //maggiore o uguale
< //minore
<= //minore o uguale
=== // valore uguale e dello stesso tipo
!== // valore diverso oppure tipo diverso

Gli operatori logici

Gli operatori logici. Con gli operatori logici possiamo combinare più enunciati semplici ottenendo un enunciato composto. Questi valori sono:

OR: FALSO se entrambi gli operatori sono falsi;
si può indicare con '***Or***' oppure '***||***' (ma anche '***OR***' oppure '***or***')

AND: VERO se entrambi gli operatori sono veri;
si può indicare con '***And***' o '***&&***' (ma anche '***AND***' oppure '***and***')

NOT: è la negazione logica e risulta VERO quando l'operatore è FALSO, e viceversa;
si può indicare con '***Not***' oppure '***!***' (ma anche '***NOT***' oppure '***not***')

XOR: viene chiamato anche 'or esclusivo', e valuta se **uno solo** dei due operatori è vero: l'altro deve essere falso;
si può indicare con '***Xor***'(ma anche '***XOR***' oppure '***xor***')

Gli operatori logici

Per quanto riguarda gli operatori 'and' e 'or', le due diverse notazioni differiscono per il livello di precedenza in caso di espressioni complesse.

Infatti, siccome è possibile combinare molti operatori in espressioni anche assai complicate, è necessario sapere con quale ordine PHP valuta i diversi operatori. Queste regole ricalcano le regole algebriche ma sono più complesse perché devono considerare anche altri operatori.

1] Operatori di incremento e decremento (**++ --**)
2] Moltiplicazione, divisione, modulo (***** **/** **%**)
3] Addizione e sottrazione (**+ -**)
4] Operatori di confronto per minore e maggiore (**< <= => >**)
5] Operatori di confronto per uguaglianza e disuguaglianza (**== === !=**)
6] Operatore logico 'and', nella notazione col simbolo (**&&**)
7] Operatore logico 'or', nella notazione col simbolo (**||**)
8] Operatori di assegnazione, compresi quelli 'sintetici' (**= += -= /= *= %= .=**)
9] Operatore logico 'and', nella notazione letterale (***And***)
10] Operatore logico 'xor' (***Xor***)
11] Operatore logico 'or', nella notazione letterale (***Or***)

Le espressioni

In generale, in PHP, qualsiasi cosa utilizzabile come un valore può essere considerata un'espressione. Vediamo alcuni rapidi esempi:

```
15 * 3; //espressione il cui valore è 45
'Giacomo' . ' Verdi';   //espressione il cui valore è 'GiacomoVerdi'
$a + $b;  //espressione il cui valore è dato dalla somma dei valori delle variabili $a e $b
$a = 6;    //il valore di questa espressione è 6
```

Quando usiamo una espressione per assegnare un valore ad una variabile, il valore che tale espressione assume è uguale a quello che si trova a destra dell'operatore di assegnazione (che è anche quello che viene assegnato all'operatore di sinistra).

Questo significa che noi possiamo scrivere:

```
echo 'Paolo';                  //stampa 'Paolo'
echo ($nome = 'Paolo');        //stampa sempre 'Paolo' (ed inizializza $nome)
echo $nome = 'Paolo';          //stampa sempre 'Paolo' (ed inizializza $nome)
```

Le espressioni

Vediamo qualche altro esempio:

```
7 > 4;          //valore dell'espressione: true (vero)
$a = 7 > 4;     /*valore dell'espressione: lo stesso di prima; la variabile $a assume quindi il valore true*/
$b = 5 * 4;     //valore dell'espressione 20 che viene assegnato a $b
```

Esiste una differenza nella valutazione dell'espressione fra i diversi modi di utilizzare gli operatori di incremento e di decremento postfissi e prefissi

Esempio:.

```
$a = 5;
$b = 5;
echo ++$a;      //$a diventa 6, e viene stampato '6'
echo $b++;      //anche $b diventa 6, ma viene stampato '5'
echo ++$b;      //a questo punto $b è diventato 7, e viene stampato '7'
```

Conversioni tra tipi

Il PHP è un **linguaggio a bassa tipizzazione** questo significa che è possibile far passare semplicemente una variabile da un tipo ad un altro.

```
$a = 5;          //intero
$b = "6";        //stringa
$c = "f";        //stringa
$d = $a . $b;    //$d sarà uguale alla stringa "56";
$e = $a + $b;    //$e sarà uguale al numero 11;
$f = $a + $c;    //$f sarà uguale al numero 5;
```

(vedi esempio **PPT-1-esempio-9.php**)

Da intero a stringa => carattere/i corrispondente al numero
Da stringa a intero => se esiste numero corrispondente a stringa altrimenti 0
Da intero a booleano => FALSE se è 0 (o minore) TRUE se è maggione di 0
Da booleano a intero => FALSE = 0 TRUE = 1
Da stringa a booleano => FALSE se stringa vuota, TRUE in tutti gli altri casi
Da booleano a stringa=> FALSE = “”, TRUE = “1”;

Costanti

```
<?php
define ("PIGRECO",3.14159);
echo PIGRECO;    //stampa 3.14159
?>
```

N.B. Si noti lo standard di utilizzare nomi in maiuscolo per le costanti al fine di distinguere le costanti dalle variabili

Training 1

Cosa verrà stampato a video?

```
<?php
$a = 5;
$b = 73;
$c = $b % $a;
echo $c . "<BR>";        /* ← visualizzazione $c */

$c++;
$risultato = "il risultato e' ";
$app = $c * $a++;
$risultato .=  $app + $a;

echo $app . "<BR>";   /* ← visualizzazione $app */
echo $risultato;          /* ← visualizzazione finale */
?>
```

(vedi **PPT-1-esempio-10.php**)

Training 2

Cosa verrà stampato a video ?

```
<?php
$a = "3";
$b = "";
$c = "-1";
$d = ($a + $c) * $a++;
$d += $a;
$e = $d + ($a And $b And $c Or $d);
echo $e
?>
```

(vedi **PPT-1-esempio-11.php**)

Esercizi

- 1) Dato un numero in ingresso (*Es: $num = 35*) stampare il suo quadrato

- 2) Dato un numero in ingresso stampare se è il numero è pari o dispari

- 3) Dati due numeri un ingresso (Es :$num1 = 3456; $num2 = 67;) stampare il resto della loro divisione e la media aritmetica

- 4) Dati due numeri un ingresso (Es :$num1 = 44; $num2 = 56;) stampare il minimo ed il massimo