

PHP

Prof. Capobianco ing. Nicolò

Introduzione

PHP (Hypertext Preprocessor) è un prodotto Open Source, gratuito ed è praticamente comprensibile dalla totalità dei Server.

La sua più grande qualità, oltre a quella di essere un linguaggio di programmazione completo e moderno, è che i suoi script sono “embedded server side” ossia incorporati nel server dove vengono interpretati ed eseguiti.

Sul browser client arriva solo ciò che il server interpreta.

Il PHP viene, infatti, definito come un “linguaggio script dal lato del server immerso nel codice HTML”.

Il PHP permette di rendere dinamico l'HTML che è solo un linguaggio di descrizione e formattazione del testo.

Il codice può essere messo in un qualunque punto della pagina HTML dentro i tag `<? php ?>` o spesso `<? ?>`

[esempio1.php](#)

Introduzione

Il PHP permette di rendere dinamiche le pagine HTML dal lato server. Il server qui utilizzato è APACHE 2.

Il PHP viene utilizzato assieme a tutti i database più conosciuti. In particolare useremo MySQL.

Inoltre, interagisce con numerosi servizi tramite i protocolli: IMAP, POP3, HTTP, SNMP, ...

Esiste anche la possibilità di usarlo su Windows anche se le prestazioni migliorano se usato su piattaforma Gnu/Linux.

Oltre ai marcatori già visti, è possibile usare anche:

`<script language="php"> </script>` (classico)

`<%%>` (like ASP)

E' necessario vedere il file **php.ini** per poter impostare i marcatori e non solo....

Variabili - 1

- ◆ Le variabili devono essere sempre precedute dal simbolo \$ e sono case sensitive.
- ◆ Non devono essere inizializzate prima del loro utilizzo.
- ◆ Il tipo sarà quello del primo dato assegnato.
- ◆ E' possibile cambiare il tipo di dati con l'istruzione: `settype($variabile, tipo_variabile);` Restituisce un valore booleano (T o F) ed il tipo è scalare e object.
- ◆ E' possibile ottenere il tipo di dato contenuto in una variabile con l'istruzione: `gettype($variabile);` Restituisce una stringa che contiene uno di questi tipi: integer, double, string, array, object, unknown type.
- ◆ E' possibile cancellare una variabile con l'istruzione: `unset($variabile);`
- ◆ Le variabili sono: **scalari** (integer, boolean, double, string), **composti** (array, object), **speciali** (resource, NULL).
- ◆ [esempio2.php](#)

Variabili - 2

- ◆ Il PHP riconosce come parametro di escape la \ come il C++.
- ◆ Alcuni caratteri:
 - ◆ \t tabulazione
 - ◆ \n fine riga
 - ◆ \\ barra obliqua inversa
 - ◆ \" doppio apice
 - ◆ \' apostrofo
 - ◆ \\$ dollaro
- ◆ Le variabili possono essere locali e globali. Come in C++.
- ◆ Se vogliamo forzare ad usare una variabile globale dentro una funzione, è sufficiente inserire dentro la funzione la parola **global** seguita dalla variabile.
- ◆ Come in C++, possiamo riferirci alle variabili (mettendo & prima della variabile tra le proprietà della funzione).

Variabili (array - 1)

- ◆ Sono dei contenitori dove vengono inseriti dati di tipo anche diverso, che possono essere richiamati quando desiderato.
- ◆ Un array può essere creato in due modi diversi, perfettamente equivalenti:
 - ◆ con il costrutto `array()` -> **`array([chiave =>] valore,);`**
 - ◆ valorizzandone gli elementi -> `$a[]=2; $a[]="ee";`
- ◆ Si possono usare sia interi non negativi, sia stringhe per indicizzare gli array; nel qual caso la chiave deve essere fra "chiave"=>....
- ◆ Le chiavi degli array, se non altrimenti specificato, partono da 0. E' possibile non mettere tutti gli indici; ci pensa l'interprete.
- ◆ Per visualizzare il contenuto di un array, comprensivo delle chiavi, si usa `print_r($nome);`
- ◆ Gli array possono avere anche chiavi miste.

Variabili (array - 2)

- ◆ Non è necessario specificarne a priori la dimensione. Notevole dinamicità.
- ◆ Possibilità di avere array multidimensionali e array annidati.
- ◆ Per contare gli elementi contenuti in un array basta utilizzare la funzione **count()**;
- ◆ Come per le variabili, si usa la funzione **unset()**; per gli elementi contenuti in un array o tutto l'array.
- ◆ [esempio3.php](#)

Variabili (object)

Prima di parlare di oggetti, si deve vedere una classe come insieme di variabili e di metodi.

Dopo la definizione di classe, si possono istanziare uno o più oggetti.

Vedremo poi....

Costanti

Sono poco usate, non hanno il simbolo \$ e sono case-sensitive.
Per definirle si usa la funzione **define**("Nome", "valore");

Operatori

Il PHP manipola i dati mediante gli operatori:

- aritmetici e di incremento e di decremento;
- di assegnazione;
- logici;
- di confronto;
- di controllo degli errori.

Come in tutti i linguaggi, gli operatori hanno una precedenza di utilizzo che ne regola il lavoro. Per semplicità, si consiglia l'uso appropriato delle parentesi.

[esempio4.php](#)

Controllo degli errori

Supponiamo di voler aprire un file inesistente, l'interprete da errore e si ferma. Non solo, viene anche visualizzata una pagina dove viene indicato chiaramente dove si trova l'errore. Questo può essere pericoloso.

Per evitare la visualizzazione dei messaggi, è sufficiente premettere alla funzione il carattere @.

Il costrutto **die()** termina l'esecuzione del programma, mostrando il messaggio contenuto fra le sue parentesi.

Esempio: `$a= @file('file che non esiste') or die("Errore durante l'apertura del file");`

Strutture di controllo

Non possono mancare in un linguaggio di programmazione, le strutture di controllo che permettono al programmatore di far compiere delle azioni al programma nel caso si verifichino (o no) determinate condizioni.

Le strutture di controllo presenti in PHP sono:

★if, else, elseif, switch. Permettono di eseguire determinate istruzioni al verificarsi di particolari condizioni;

★while, do ... while, for, foreach. Permettono l'esecuzione di operazioni cicliche. Attenzione all'importanza dell'operazione di uscita dal ciclo, per evitare dei cicli infiniti.

★if else elseif switch

★while do ... while for foreach

Funzioni

Sono insiemi di istruzioni raccolte sotto un unico nome. Quando richiamiamo questo nome facciamo eseguire tutte le istruzioni contenute. Per richiamare una funzione bisogna digitarne il nome, in questo modo:

```
nome();
```

Se la funzione dà un risultato che vogliamo associare ad una variabile per poterlo usare, dobbiamo scrivere:

```
$a=nome();
```

da questo momento la variabile `$a` assume il valore risultante della funzione.

Anche il PHP fornisce un numero elevatissimo di funzioni predefinite, ma dà anche la possibilità di costruirne altre.

Per evitare di dover riscrivere più volte le stesse funzioni, con possibili errori e modifiche ripetute, il PHP propone una specie di librerie da premettere in ogni pagina mediante il costrutto `include()` e `require()`.

include() e require()

Un esempio potrebbe essere quello di un menu che deve essere ripetuto in ogni pagina del nostro sito. Viene scritto una volta sola e richiamato quante volte si vuole.

```
<?include("nomedelfile.php"); ?>
```

Semplicemente permette di includere file all'interno della pagina e si attiva ogni volta che lo richiede il programma. Se sussiste un problema dà un "warning" ma la pagina viene comunque aperta.

```
<?require("nomedelfile.php"); ?>
```

La differenza è che il file contenuto in require() viene comunque inserito nel listato al posto della riga require().

Se sussiste un errore genera un "fatal error" ed interrompe l'apertura della pagina.

include() e require()

In programmi complessi potrebbe accadere che il richiamo di più file porti la macchina host a leggere più include o require uguali tra loro.

Per evitare problemi, si usano i costrutti `require_once()` e `include_once()`, che si ricordano se un file è già stato aperto ed evitano il nuovo inserimento.

Nei cicli condizionali è sempre necessario mettere le parentesi graffe anche se abbiamo una sola istruzione.

[esempio7.php](#)

[prova.php](#)

Costanti matematiche predefinite

Costante	Valore	Descrizione
M_PI	3.14159265358979323846	Pi
M_E	2.7182818284590452354	e
M_LOG2E	1.4426950408889634074	$\log_2 e$
M_LOG10E	0.43429448190325182765	$\log_{10} e$
M_LN2	0.69314718055994530942	$\log_e 2$
M_LN10	2.30258509299404568402	$\log_e 10$
M_PI_2	1.57079632679489661923	$\pi/2$
M_PI_4	0.78539816339744830962	$\pi/4$
M_1_PI	0.31830988618379067154	$1/\pi$
M_2_PI	0.63661977236758134308	$2/\pi$
M_SQRTPI	1.77245385090551602729	$\sqrt{\pi}$ [4.0.2]
M_2_SQRTPI	1.12837916709551257390	$2/\sqrt{\pi}$
M_SQRT2	1.41421356237309504880	$\sqrt{2}$
M_SQRT3	1.73205080756887729352	$\sqrt{3}$ [4.0.2]
M_SQRT1_2	0.70710678118654752440	$1/\sqrt{2}$
M_LNPI	1.14472988584940017414	$\log_e(\pi)$ [4.0.2]
M_EULER	0.57721566490153286061	Costante di Eulero [4.0.2]

Funzioni predefinite – Matematica1

Funzioni matematiche: Queste funzioni operano esclusivamente nel range dei tipi di dato integer e float del computer.

$\text{abs}(x)$ – Valore assoluto di un numero

$\text{cos}(x)$ – Coseno

$\text{cosh}(x)$ – Coseno iperbolico

$\text{acos}(x)$ – Arco coseno

$\text{acosh}(x)$ – Inverso del coseno iperbolico

$\text{sin}(x)$ – Seno

$\text{sinh}(x)$ – Seno iperbolico

$\text{asin}(x)$ – Arco seno

$\text{asinh}(x)$ – Inverso del seno iperbolico

$\text{tan}(x)$ – Tangente

$\text{tanh}(x)$ – Tangente iperbolica

$\text{atan}(x)$ – Arco tangente

$\text{atan2}(y,x)$ – Arco tangente di due variabili È simile al calcolo dell'arco tangente di y / x , eccetto per il fatto che viene tenuto in considerazione il segno di entrambi gli argomenti per determinare il quadrante del risultato. La funzione restituisce il risultato in radianti, compreso fra $-\pi$ e π (inclusi).

$\text{atanh}(x)$ – Inversa tangente iperbolica

Funzioni predefinite – Matematica2

`base_convert` (stringa, basein, baseout) – Converte numero fra basi arbitrarie

`decbin` (x) – da decimale a binario

`bindec`(x) – da binario a decimale

`dechex`(x) – da decimale ad esadecimale

`hexdec` (x) – da esadecimale a decimale

`decoct`(x) – da decimale a ottale

`octdec`(x) – da ottale ad esadecimale

`deg2rad`(x) – da gradi a radianti

`rad2deg`(x) – da radianti a gradi

`max`(x1,x2,...) - trova il massimo

`min`(x1,x2,...) - trova il minimo

`sqrt`(x) – radice quadrata

`pow`(x,y) – x elevato a y

`exp`(x) – calcola l'esponente di e

`log`(x[,y]) – Senza la base y restituisce il ln, altrimenti il logaritmo in base y di x

`log10`(x) – logaritmo in base 10

`rand`([x,y]) – senza parametri restituisce un numero casuale fra 0 e `rand_max`, altrimenti un numero tra x e y

`ceil`(x) e `floor`(x) arrotondano all'intero superiore ed inferiore rispettivamente

`round`(x) – arrotonda un numero non intero

Funzioni predefinite – Collegamenti

Matematica a precisione arbitraria. [matematica2.pdf](#)

Data e tempo. Si possono usare queste funzioni per la formattazione dell'output delle date. [data.pdf](#)

Per manipolare ed interagire con gli array. [array.pdf](#)

Per manipolare le stringhe in vari modi. [stringhe.pdf](#)

Per accedere al Server MySQL. [MySQL.pdf](#)

Per la gestione delle variabili. [variabili.pdf](#)

Varie. [varie.pdf](#)

Funzioni dei tipi di caratteri. [caratteri.pdf](#)

Funzioni sul FileSystem. [filesystem.pdf](#)

Gestione degli errori. [errori.pdf](#)

Immagini. [immagini.pdf](#)

Funzioni predefinite – Collegamenti

Gestione della sessione. [sessione.pdf](#)

Url. [url.pdf](#)

ftp. [ftp.pdf](#)

http. [http.pdf](#)

mail. [mail.pdf](#)

rar. [rar.pdf](#)

xml. [xml.pdf](#)

Interfaccia alla libreria mcrypt. [mcript.pdf](#)

Funzioni apache. [apache.pdf](#)

Informazioni sulle classe e istanze degli oggetti. [classi.pdf](#)

Aiuto alle funzioni. [aiutofunzioni.pdf](#)

Conversioni tra differenti tipi di calendario. [calendario.pdf](#)

Socket. [socket.pdf](#)

Funzioni definite dall'utente

Le funzioni create dall'utente raccolgono delle istruzioni che vengono racchiuse tra parentesi graffe ed hanno un nome diverso da ogni altro nome di funzione.

La funzione deve essere dichiarata così:

```
function nomefunzione($a1,$a2,...) {  
    ...  
    ...  
}
```

I parametri non sono obbligatori.

La funzione può essere richiamata, dopo la dichiarazione, tutte le volte che si vuole. Per esempio: **`$a=nomefunzione($x,$y,...);`** o

`nomefunzione(.....);`

Funzioni definite dall'utente

Nel primo caso restituisce un valore, utilizzando il comando `return` dentro la dichiarazione della funzione.

Non importa il nome delle variabili ma solo l'ordine.

Una funzione può anche restituire più valori. Ciò è possibile restituendo un array, i cui valori possono essere successivamente scompattati.

Si può utilizzare la funzione predefinita `list()` che associa il valore della variabile con il valore dell'array che ha la stessa posizione.

Funzioni definite dall'utente

Le variabili interne alla funzione sono le variabili locali e non possono essere usate al di fuori della funzione.

Invece, le variabili esterne alla funzione sono chiamate globali e non possono essere usate dentro la funzione.

Così, due variabili una interna e l'altra esterna, possono avere lo stesso nome, senza problemi. Cioè, le due variabili sono completamente distinte fra loro.

Se vogliamo usare una variabile globale dentro la funzione dobbiamo mettere il comando `global` seguito dalla variabile globale.

Funzioni definite dall'utente

L'utilizzo di `global` non è consigliato, per evidenti motivi di modifica della variabile esterna che comporta una modifica anche all'interno della funzione.

E' possibile, come nel C/C++, riferirci alle variabili, inserendo il simbolo `&` prima del parametro passato alla funzione.

In definitiva il modo migliore di realizzare una funzione è quello di passare alla funzione tutti i dati necessari come parametri della chiamata.

Il PHP permette le funzioni ricorsive, cioè una funzione che richiama dal suo interno se stessa. E' importante in funzioni di questo tipo definire una condizione di arresto, per non finire in un ciclo infinito. [Esempi di funzioni](#)

Passaggio di variabili tramite HTTP

Fino ad ora abbiamo trattato pagine di singole pagine non dinamiche ed interattive.

Il protocollo HTML, però, permette con i suoi tag di passare dei dati tra le pagine. Il PHP poi elabora questi dati.

Il tag che si usa è il `<form>` che riceve dagli attributi `action` e `method` le direttive relative al file che deve ricevere i dati ed al modo in cui essi devono essergli passati.

L'attributo `method` ammette due valori: `get` e `post`.

Passaggio di variabili tramite HTTP - GET

Il metodo **get** prevede che i nomi ed i valori da associare vengono forniti allo script tramite la barra dell'indirizzo del browser.

Bisogna scrivere un primo file html con il form ed il metodo get che passa i dati al secondo file di tipo php che li riceve e li elabora.

Per capire come funziona il metodo get basta osservare il contenuto della barra degli indizi nel browser quando viene visualizzato il secondo script.

All'indirizzo della pagina viene aggiunto un carattere ? e tutte le coppie variabili=valori e separate tra loro dal carattere &.

file origine

file destinazione

Passaggio di variabili tramite HTTP - POST

Nel metodo **post** i dati non vengono visualizzati in alcun modo. Essi sono spediti tramite il protocollo HTTP e non sono visibile sul browser.

Bisogna scrivere un primo file html con il form ed il metodo post che passa i dati al secondo file di tipo php che li riceve e li elabora.

file origine

file destinatario

Quale metodo scegliere?

Se si vuole spedire molti dati, conviene usare il metodo post. Nel file php.ini è possibile definire sia la massima dimensione del file, sia la massima dimensione dei dati accettati tramite il post (inserendo un valore nella variabile `post_max_size`).

Nel caso del get, i dati da spedire devono essere pochi e sono visibili. Esso è comodo quando si usa spesso il tasto avanti e dietro del browser.

Gestione file system

PHP è un linguaggio di programmazione autonomo e derivante dal C ed è nato per soddisfare le esigenze dei programmatori internet. Quindi, il binomio PHP – MySQL o qualsiasi altro database è venuto successivamente.

Tra l'altro i gestori fanno pagare l'uso dei database.

PHP permette di lavorare con i file di testo (.txt): aprirli, leggerli, modificarli, crearli, salvarli.

Ricordiamo che un errore sui file di testo produce un warning su video. Per evitare si inserisca il simbolo @ prima della funzione e così l'errore non viene mostrato.

Non ci sono limiti minimi o massimi per le dimensioni del file.

A differenza degli array non è possibile puntare direttamente ad una riga ben precisa, perchè l'accesso è “sequenziale”.

Le funzioni che gestiscono i file si trovano in [filesystem.pdf](#).

Gestione file system

Noi vediamo le funzioni più utilizzate che permettono l'apertura, la lettura, la scrittura, il salvataggio, la chiusura.

Prima di tutto il file deve essere aperto mediante la funzione `fopen("file.txt", tipo di apertura)`.

Il tipo di apertura può essere:

`r` -> solo lettura, puntatore inizio file;

`r+` -> lettura e scrittura, puntatore inizio file;

`w` -> solo scrittura, puntatore inizio file, elimina il contenuto del file e se non esiste lo crea;

`w+` -> lettura e scrittura, puntatore inizio file, elimina il contenuto del file e se non esiste lo crea;

`a` -> solo scrittura, puntatore fine file, se non esiste lo crea;

`a+` -> lettura e scrittura, puntatore fine file, se non esiste lo crea.

Gestione file system

Il PHP permette anche di aprire file remoti tramite i protocolli più diffusi come HTTP e FTP; è ovvio che il file deve essere accessibile almeno in lettura.

In lettura, bisognerà controllare se il file esiste ed effettuare il controllo dell'errore:

- `file_exists()` permette di vedere l'esistenza del file;
- `@fopen()` apre il file con il controllo dell'errore.

Inoltre, bisogna sapere la fine del file con la funzione `feof()`; per la lettura delle singole righe si usa la funzione `fgets()` che restituisce una stringa o un valore falso in caso di errore. Per la lettura di un carattere si usa la funzione `fgetc()`.

Per evitare scritture non desiderate, è necessario chiudere il file quando si è finito il lavoro: `fclose()`.

Nel caso in cui il file da leggere è piccolo si usa la funzione `file()` che restituisce un array con gli elementi uguali ad ogni riga del file di testo. Molto utile al posto dei database. Non necessita di aprire e chiudere il file. Il contenuto, così, può essere solo letto.

Gestione file system

La funzione `fread()` estrae un numero definito di byte dal file di testo. Se volessimo leggere tutto il file si utilizza la funzione `filesize()` che restituisce la grandezza totale del file. Se in più ci sono dei rientri a capo che vogliamo visualizzare dobbiamo inserire la funzione `nl2br()` che aggiunge un `
` quando vi è un ritorno a capo. [lettura.php](#)

Se vogliamo scrivere dobbiamo aprire il file in uno dei modi di scrittura o di accodamento. Per scrivere si può usare `fwrite()` (o `fputs()`) che mette una stringa nel file appena aperto. L'unica differenza è il punto in cui viene effettuato l'inserimento. [home page](#)

Gestione file system

Può capitare che i permessi di accesso dei file sia tale da impedire l'apertura del file stesso. Bisogna modificare i permessi sui file mediante delle funzioni che devono essere inserite prima della funzione che chiude il file.

```
chmod("file", numero), con numero a 4 numeri (0777,  
0600,0644,0755,0750)
```

```
ghgrp("file", "nomegruppo");
```

```
chown("file", "nomeproprietario");
```

Esempio:

```
@$a=fopen("file","w");
```

```
@chmod("file",0777);
```

```
@fclose($a);
```

Sessioni

La sessione è il modo più semplice per riconoscere un utente da quando si collega al server a quando interrompe il collegamento.

Per essere più chiari supponiamo di fornire una password per un'area protetta e poi di navigare. Per ogni pagina dovremmo ridigitare la password. Ma non è così. Perché la nostra sessione è sempre attiva ed il server si ricorda che abbiamo inserito la password e siamo abilitati.

Il funzionamento della sessione è molto semplice. Il server apre una sessione e le assegna un nome univoco. Analizza l'utente, raccoglie i dati che vogliamo e li salva in un file che porta lo stesso nome della sessione. Nello stesso tempo deposita un cookie nella macchina dell'utente.

Ogni volta che ne ha bisogno, dal cookie estrae i dati per risalire alla sessione ed al file salvato. In pratica abbiamo una memoria virtuale salvata sul server.

Quando interrompiamo il collegamento il server cancella tutto.

Sessioni

Come abbiamo visto, la sessione salva un cookie nel browser di chi naviga. Per questo dobbiamo porre la funzione che inizia la sessione prima di ogni altro dato in output dal server. In pratica, neanche il codice HTML deve precedere questa funzione.

Ma se il navigatore ha i cookie disabilitati? Non si apre la sessione.

Il PHP ci fornisce una soluzione che però è poco sicura. Possiamo passare i dati tramite url. La sicurezza è compromessa perchè chiunque può prendere nota del codice di sessione (possibilità comunque remota).

Per procedere con questa soluzione dobbiamo inserire un codice in ogni collegamento a pagine che necessitano della sessione:

```
<a href="newpage.php?<? echo SID?>">Clicca qui</a>
```

L'istruzione `<? echo SID?>` può, per motivi di sicurezza, essere inserita solo in query che puntano a pagine del nostro sito.

SID è una variabile che viene definita solo se il browser utente non supporta i cookie.