

Apache Web Server

Il primo sito italiano completamente dedicato al Web Server Apache
Piattaforma Win32

Lezioni sulla programmazione PHP a cura di Davide Anastasia

N° Lezione	Titolo	Data
Lezione 1	Introduzione	28/05/2000
Lezione 2	Come s'installa il PHP3?	28/05/2000
Lezione 3	Come inserire il codice PHP?	03/06/2000
Lezione 4	Include e Require	13/06/2000
Lezione 5	Operatori	20/06/2000
Lezione 6	Variabili	27/06/2000
Lezione 7	Installare PHP4	03/07/2000
Lezione 8	Ciclo if	12/07/2000
Lezione 9	Ciclo if 2	15/07/2000
Lezione 10	Ciclo if 3	01/09/2000
Lezione 11	While // While - Do	01/09/2000
Lezione 12	FOR	01/09/2000
Lezione 13	Installare MySQL	01/09/2000

Lezione sulla programmazione PHP a cura di Davide Anastasia

Lezione 1 – Introduzione

Cos'è il PHP

Il PHP è un linguaggio implementato lato server (server-side HTML-embedded scripting language). Il suo funzionamento è molto semplice ed efficace. L'engine del PHP dà come risposta delle richieste al Web Server (nel nostro caso Apache) pagine HTML completamente formattate, rendendo il codice PHP perfettamente trasparente all'utente finale. Visualizzando il codice delle pagine (l'estensione delle pagine PHP è **.php3**, tranne diversi settaggi del Web Server) si vedrà solo HTML puro e, di fatto, il sorgente della pagina PHP può essere modificato solo dal Web Master.



Com'è nato PHP?

Il PHP nasce a metà del 1994 dalle mani di Rasmus Lerdorf. Egli sviluppò una prima versione mai ufficializzata che utilizzò sulle sue pagine. Ma il successo non tardò, e già nella prima parte del 1995 uscì la prima versione del **Personal Home Page**. Da lì a poco l'engine, anche grazie alla sua filosofia free, si è affermato ed adesso vanta qualcosa come 150.000 siti che implementano questo linguaggio. Cifre di tutto rispetto!

La versione ufficiale

Da pochi giorni sul sito <http://www.php.net/> è stata ufficializzata l'ultima versione del PHP: la **4.0.0**. Noi in ogni modo ci soffermeremo sulla versione 3 perché offre maggiore stabilità. Inoltre un'altra azienda produttrice di software, **Zend** (<http://www.zend.com/>), ha realizzato un progetto molto interessante che si propone di migliorare la velocità dell'engine di una percentuale tra il 40% e il 100%. Anche questo software è free. In ogni modo il progetto è supportato solo dalla versione 4 dell'engine PHP e quindi noi ne parleremo più in là.



Lezioni sulla programmazione PHP a cura di Davide Anastasia

Lezione 2 - Come s'installa il PHP 3?

Installare l'engine PHP 3 sul Web Server Apache è molto semplice. Fidatevi di me! Ecco spiegato cosa fare in pochi passi:

1. La prima cosa da fare è scaricare l'engine dal sito ufficiale del PHP (<http://www.php.net/>). Una volta scaricato bisogna scompattare il pacchetto **.zip** in una cartella a piacere (vi consiglio **c:\php3** ... il tutto sarà più facile successivamente!);
2. Nel pacchetto che avete scompattato c'è un file che si chiama **php3.ini-dist**. Dovete prendere questo file e copiarlo nella cartella **c:\windows** per *Windows 95/98* / **c:\winnt** o **c:\winnt40** per *Windows NT Servers* e rinominarlo **php3.ini**. Ricordate che questo è un passaggio importante! Non sbagliate!;
3. Il file **php3.ini** va modificato in alcune righe... ora vi dico quali:
 - o La riga con il campo "**extension_dir**" va modificata con la patch della cartella in cui avete scompattato il pacchetto **.zip**; se avete seguito le mie istruzioni dovete settare questo valore con **c:\php3**;
 - o La riga con il campo "**doc_root**" va modificata con la patch della root dei documenti del server. Insomma, con la directory più bassa del Web Server Apache (di solito **c:\programmi\apache group\apache\htdocs** ... ma se avete eseguito l'installazione di Apache in altre patch controllate bene! Se sbagliate qui il risultato è assicurato: non parte niente!);
 - o Ci sono varie righe con il campo "**extension=php3_*.dll**". Ognuno di loro carica un modulo del PHP... se volete andare sul sicuro decommentate tutti i campi! L'avvio dell'engine sarà di qualche decimo di secondo più lento, ma poi non avrete problemi!
4. A questo punto siamo a metà dell'opera. Ora dovete salvare tutto il lavoro fatto sul file **php3.ini**;
5. Modificato il file **php3.ini**, adesso è l'ora di modificare il file **httpd.conf** di Apache. Quindi aprite questo file e aggiungete queste righe:

ScriptAlias	/php3/	"c:/path-to-php-
dir/"		
AddType	application/x-httpd-php3	.php3
AddType	application/x-httpd-php3	.phtml
Action	application/x-httpd-php3	"/php3/php.exe"

Ricordate che nel primo rigo va cambiato il valore "c:/path-to-php-dir/" con la patch della directory dove avete scompattato il pacchetto .zip dell'engine (se avete seguito le mie istruzioni dovete settare questo valore con **c:\php3**);

6. Ora tutto dovrebbe funzionare! Quindi fate ripartire Apache;
7. Per vedere se tutto è ok dobbiamo creare un file di prova. Il file, che chiameremo **index.php3**, deve contenere solo una riga:

```
<?php phpinfo(); ?>
```

Salvate il file nella root del Web Server e richiamatelo dal browser... se tutto funziona uscirà una pagina chilometrica che v'indicherà tutti i settaggi del PHP... prendete tutto per buono! Impareremo dopo cosa significano!

Come vedere tutta l'installazione è abbastanza semplice. Il tutto è fattibile con pochi passi ed in pochi minuti, ma è importante non sbagliare niente! Nel caso il Web Server faccia i capricci, ripartite da capo e ricontrollate tutti i passaggi... In ogni modo fate sempre riferimento al manuale del PHP che potrete trovare al sito <http://www.php.net/> ...

Lezioni sulla programmazione PHP a cura di Davide Anastasia

Lezione 3 - Come inserire il codice PHP?

Come sappiamo il codice PHP è interpretato dall'engine che si presenta come un modulo del Web Server, nel nostro caso Apache. Ma come fa l'engine a capire cosa deve elaborare?

La prima cosa è sicuramente l'impostazione dell'estensione del file php (di default **.php3**). Infatti quando al Web Server viene chiesta una pagina con estensione **.php3** egli gira la pagina direttamente all'engine, che ha poi il compito di capire cosa interpretare e di cosa lasciare così com'è e di restituirci la pagina. L'engine capisce cosa processare tramite un tag particolare. Questo particolare "codice" è ritrovabile in 4 modi differenti. Eccoli:

1. **<?php ...codice php... ?>**

Questo è il metodo di default per inserire un'istruzione php. Tutto quello che si trova all'interno dei tag verrà processato dall'engine!

Esempio:

```
<html>
<head>
<title>...</title>
</head>

<body>
<!-- codice HTML non interpretato -->

<?php
/* codice PHP interpretato */
?>

<!-- codice HTML non interpretato -->
</body>
</html>
```

2. **<? ...codice PHP... ?>**

Questa è una sintassi molto simile a quella precedente, ma per essere usata c'è bisogno dell'abilitazione nel file **php3.ini** (lo ricordate?!?).

Ora vi dico cosa fare per modificare questa possibilità: nella sezione "**Language Options**" dovete modificare il parametro "**short_open_tag**" ed inserire "**On**".

3. **<script language="php">**

...codice PHP...

</script>

Questa sintassi è simile a quella di Javascript e può essere molto utile nell'utilizzo di editor HTML visuali che non conoscono le estensioni PHP. E' attivo di default.

4. **<% ...codice HTML... %>**

Questa è la sintassi utilizzata dalle ASP e deve essere attivata per essere utilizzata. La sua attivazione deve essere effettuata nel file **php3.ini**. Ecco cosa fare: nella sezione "**Language Options**" dovete modificare il parametro "**asp_tags**" ed inserire "**On**".

Ecco in breve una prima infarinatura di PHP. Dalla prossima volta incominceremo a vedere qualcosa di un po' più serio!

Lezioni sulla programmazione PHP a cura di Davide Anastasia

Lezione 4 - INCLUDE e REQUIRE

Uno degli utilizzi più classici delle architetture Server Side è quello della creazione di pagine "dinamiche" costituite come mosaici, formati da tanti file che si legano tra loro formando il risultato finale. Essendo la risposta del Server ad una richiesta di pagina .php3 puro codice HTML, questo sistema è un espediente molto elegante per modificare le pagine non intervenendo su tutto il codice ma solo sul file collegato al principale che contiene ciò che deve essere modificato.

Spiego tutto con un esempio:

Utilizzando i FRAME, ormai poco eleganti ma utili in alcuni casi, una pagina così composta:



sarebbe poco bella da vedere. Infatti il frame superiore (che probabilmente conterrà un menù) sarebbe immobile, e di conseguenza difficilmente integrabile in una pagina dalla grafica complessa! Con l'istruzione INCLUDE tutto si risolve perché una pagina costituita in questo modo scrollerebbe con tutta la grafica rimanente. L'effetto è di sicura efficacia e molto utile nell'economia di un grosso sito!

Ora vi spiego i comandi:

Il comando INCLUDE è di facile utilizzazione. Questa è la sua sintassi più semplice:

```
<?php  
include ('file.estensione');  
?>
```

dove "file.estensione" sta per il nome del file da includere (es. include.inc).

La sintassi del comando REQUIRE è la stessa.

E' da notare poi che i comandi INCLUDE e REQUIRE possono essere utilizzati anche nel corso di cicli iterativi che richiamino la stessa funzione. Includendo il file in cui è presente un certo comando PHP esso sarà automaticamente incluso all'interno del ciclo.

Lezioni sulla programmazione PHP a cura di Davide Anastasia

Lezione 5 - Gli operatori

Uno degli aspetti più importanti, anche se molto semplici, d'ogni linguaggio di programmazione sono gli operatori. **Gli operatori sono quei caratteri particolari che indicano all'engine (nel nostro caso) che tipo d'operazione deve svolgere.**

Gli operatori possono essere fondamentalmente di due tipi: **operatori aritmetici** e **operatori logici**. Ma anche d'altri tipi. Ecco qui le tabelle dei principali operatori.

Operatori aritmetici

Questi sono gli operatori aritmetici.

Esempio	Nome	Risultato
$\$a + \b	Addizione	Somma del valore di \$a e quello di \$ b
$\$a - \b	Sottrazione	Rimanente di \$b sottratto ad \$a
$\$a * \b	Moltiplicazione	Prodotto di \$a e \$b
$\$a / \b	Divisione	Diviso di \$a e \$b
$\$a \% \b	Modulo	Rimanente di \$a diviso per \$b

Gli operatori aritmetici eseguono operazioni di puro calcolo matematico che sono utili all'interno di programmi che eseguano questo tipo d'operazioni. Ma è importante notare che gli operatori, di qualunque tipo siano, sono molto importanti sempre all'interno di programmi e che quindi vanno imparati molto bene!

Operatori logici

Gli operatori logici sono simboli che effettuano delle scelte. Se si ha dimestichezza con le porte logiche, è bene specificare che gli operatori logici funzionano esattamente allo stesso modo!

Questi sono gli operatori logici più importanti. Dalla tabella capirete meglio perché si chiamano così.

Esempio	Nome	Risultato
$\$a \text{ and } \b	And	Vero se \$a e \$b sono veri entrambi
$\$a \text{ or } \b	Or	Vero se o \$a o \$b è vero
$\$a \text{ xor } \b	Or	Vero se o \$a o \$b è vero, ma non entrambi
$! \$a$	Not	Vero se \$a non è vero
$\$a \&\& \b	And	Vero se \$a e \$b sono veri entrambi
$\$a \ \ \b	Or	Vero se o \$a o \$b è vero

Come vedete dalla tabella gli operatori logici pongono delle vere e proprie condizioni affinché il "programma" giri.

Come avete notato, per "and" e "or" ci sono due diversi modi di scrittura. Questi sono differenti nella loro precedenza, come vedremo dopo.

Operatori Bitwise

Gli operatori bitwise [parola intraducibile! :-)] permettono di accendere o spegnere specifici bit all'interno d'interi. Per capire meglio guardate la tabella.

Esempio	Nome	Risultato
$\$a \& \b	And	Valido se \$a e \$b sono entrambi definiti
$\$a \b	Or	Valido se o \$a o \$b è definito
$\$a \wedge \b	Xor	Valido se o \$a o \$b è definito, ma non entrambi
$\sim \$a$	Not	Valido se \$a non è definito e viceversa

Come vedete, gli operatori Bitwise "guardano" all'interno delle variabile e scoprono se sono definite. Poi si regolano di conseguenza.

Operatori di comparazione

Esistono anche operatori che effettuano delle comparazioni tra variabili: vediamo.

Esempio	Nome	Risultato
<code>\$a == \$b</code>	Uguale	Vero se il valore di \$a e \$b coincidono
<code>\$a != \$b</code>	Non uguale	Vero se il valore di \$a e \$b non coincidono
<code>\$a < \$b</code>	Più piccolo di...	Vero solo se \$a è più piccolo di \$b
<code>\$a > \$b</code>	Più grande di...	Vero solo se \$a è più grande di \$b
<code>\$a <= \$b</code>	Minore o uguale	Vero se \$a è minore o uguale a \$b
<code>\$a >= \$b</code>	Maggiore o uguale	Vero se \$a è maggiore o uguale a \$b

Esistono poi alcuni operatori speciali. Il primo operatore è `=` e la sua funzione è quella di assegnare il valore ad un variabile (es. `$a = 1`).

L'altro operatore, chiamato di concatenazione, svolge appunto la funzione di concatenare più variabili (es. `$a = "Hello "; $b = $a . "World!"; // now $b = "Hello World!"`). Il suo simbolo è il punto (`.`).

E' da fare un'altra considerazione. Esiste una precedenza negli operatori, ma la tabella di queste precedenza sarebbe troppo lunga da riportare qui, e di fatto poco utile. Per questo vi rimando al manuale ufficiale del PHP per maggiori chiarimenti.

Con questo ho finito e vi aspetto alla prossima per parlare di variabili e array.

Lezioni sulla programmazione PHP a cura di Davide Anastasia

Lezione 6 - Le variabili

Uno degli aspetti base d'ogni linguaggio di programmazione sta nelle variabili. Le variabili del PHP, in modo simile al Perl, ma diversamente dal C, sono molto flessibili e vengono riconosciute automaticamente. La variabile PHP viene definita nel suo contenuto dall'engine. Chi ha una anche leggera infarinatura del C sa che invece lì le variabili devono essere definite. (Chi di voi ricorda il Pascal?!? Beh, in quel caso le variabili andavano ben specificate all'inizio del programma... ed era spesso lì l'errore... ma non perdiamo il filo!). Le variabili possono nel PHP contenere di tutto. Imparerete con il tempo che la variabile del PHP può essere usata come un notes dove inserire una funzione che deve essere richiamata più volte. Ora il discorso potrà sembrare campato in aria, ma quando arriveremo a parlare di database capirete bene di cosa parlo!!

L'inizializzazione di una variabile può essere fatta semplicemente assegnandole un valore (ricordate l'operatore =). Non deve essere definita all'inizio, né tanto meno si deve definire di che tipo è. A volte però può essere utile e necessario farlo! Per assegnare anche il tipo alla variabile questa è la sintassi:

```
$variabile = (int) 1;
```

Per sapere quali sono i tipi di variabili (da sostituire a posto di **int**) consultate il manuale del PHP.

Importante: il simbolo che identifica la variabile è **\$** (che ricorda il denaro!!).

L'inizializzazione di un array è fatta assegnandole un valore... ma in modo diverso!! Ecco l'esempio:

```
$nomi[ ] = "Davide" // Questo valore è identificato come array $nomi[0]  
$nomi[ ] = "Fabio" // Questo valore è identificato come array $nomi[1]  
.... così via!
```

In questo modo quando viene inserito un nuovo valore nell'array, questo diventa l'ultimo. **E' importante ricordare che i membri di un array partono nel conteggio da 0 e non da 1....** è molto importante e ci servirà successivamente!

Esistono array anche multidimensionali (e saranno molto utili in futuro... capiteli bene!!), differenti da quelli visti in precedenza che invece erano array monodimensionali. Ecco alcuni esempi di array multidimensionali:

```
# Example 1:  
$a["color"] = "red";  
$a["taste"] = "sweet";  
$a["shape"] = "round";  
$a["name"] = "apple";  
$a[3] = 4;
```

```
# Example 2:  
$a = array(  
"color" => "red",  
"taste" => "sweet",  
"shape" => "round",  
"name" => "apple",  
3 => 4  
);
```

Esempi estrapolati direttamente dal manuale del PHP.

Entrambi questi array costruiscono la medesima struttura, ma sono scritti in maniera differente! Se provassimo ad esempio a richiamare una funzione del genere:

```
echo $a[name];
```

quale sarebbe il risultato?

A video comparirebbe:

```
apple
```

Ora vediamo un array multidimensionale più complesso:

```
$a = array(  
  "apple" => array(  
    "color" => "red",  
    "taste" => "sweet",  
    "shape" => "round"  
  ),  
  "orange" => array(  
    "color" => "orange",  
    "taste" => "sweet",  
    "shape" => "round"  
  ),  
  "banana" => array(  
    "color" => "yellow",  
    "taste" => "paste-y",  
    "shape" => "banana-shaped"  
  )  
);
```

Notate le parti sottolineate... in questo caso si è creato un array multidimensionale più complesso. Per identificare un valore dell'array si devono richiamare 2 dimensioni... così:

```
echo $a["apple"]["taste"];
```

E come output ci sarebbe:

```
sweet
```

Con questo ho concluso questa lezione. Se avete dubbi su questa lezione fate un post sul forum... ok?
Ciao a tutti (e so che siete tanti!!).

Lezioni sulla programmazione PHP a cura di Davide Anastasia

Lezione 7 - Come si installa il PHP 3? Oopss.. il PHP 4!

L'installazione su Apache per Win9x del nuovo PHP 4 è molto simile alla precedente. Infatti, ci sono delle piccole differenze al file di configurazione di Apache rispetto alla versione precedente. Per questo di base partiremo da quella seconda lezione in cui vi ho spiegato come installare il PHP 3, spiegandovi le differenze e mettendo a confronto le operazioni. Le prime operazioni, quelle che riguardavano il file php3.ini (ricordate che adesso quel file è cambiato e si chiama solo php.ini), la sua configurazione, il caricamento dei moduli e la scompattazione della distribuzione in formato binario è sostanzialmente la stessa. Ma c'è da far un appunto. Ho notato che tutti i moduli sono caricati di default anche se sono commentati. Infatti, cercando di attivare il modulo MySQL, ho decommentato quella riga e mi sono ritrovato con l'engine piantato. Ho ricommentato quella riga e tutto ha funzionato. Infatti, anche nella pagina delle info del server (ricordate: `<? phpinfo() ?>`) ho trovato che il supporto MySQL era attivo, nonostante la riga che caricasse il modulo fosse commentata... vabbo', poco male! Se ricordate nella precedente installazione vi ho fatto settare i parametri "extension_dir" e "doc_root" nel file php3.ini. Questa volta vi basterà settare solo nel file php.ini il parametro "extension_dir"... e ricordate quello che vi ho detto sui moduli! Ora tocca alla parte differente rispetto alla versione precedente (ho fatto la rima!). Nel file httpd.conf di Apache le righe da inserire sono leggermente differenti.

Nella versione precedente abbiamo aggiunto questo:

```
ScriptAlias /php3/ "c:/path-to-php-dir/"
AddType application/x-httpd-php3 .php3
AddType application/x-httpd-php3 .phtml
Action application/x-httpd-php3 "/php3/php.exe"
```

Ora dobbiamo aggiungere questo (o basterà rimpiazzarlo se si sta facendo un upgrade):

```
ScriptAlias /php4/ "c:/php4/"
AddType application/x-httpd-php .php3 .php
AddType application/x-httpd-php .phtml
AddType application/x-httpd-php-source .phps
Action application/x-httpd-php "/php4/php.exe"
```



Ora se riprovate a far ripartire il server tutto dovrebbe non funzionare... perché?

Perché c'è una differenza molto importante con la versione precedente che non fa funzionare niente se non è svolta a dovere. Infatti ci sono due *.dll da mettere nell'odiata cartella c:\windows\system, altrimenti niente parte. I file sono "Msvcr.dll" e "php4ts.dll", che sono all'interno del pacchetto zip che avete scompattato.

Adesso, riavviando il web server tutto dovrebbe partire. Testate con la solita pagina `<? phpinfo() ?>`.

Tutto ok? Allora, è stato facile? Come sempre!

Il supporto Zend

Come molti di voi avranno notato da quando è sorto lo sviluppo del nuovo PHP 4 una nuova casa produttrice di software sta accompagnando questo sviluppo. E' la Zend. Questa società si è fatta carica della completa riscrittura dell'engine che adesso utilizza un diverso sistema di lavoro che velocizza di molto la restituzione delle pagine al client. Un enorme passo avanti per il PHP che avevo proprio il suo punto debole nella velocità.



Ma non solo, Zend lavora ad altri software che accompagnano l'engine PHP. Ma per questo v'invita a visitare il loro sito:

<http://www.zend.com/>. Vorrei solo soffermarmi un attimo su uno

solo dei componenti aggiuntivi del nuovo PHP. E' lo Zend

Optimizer. Installando questo software sul mio engine ho visto un miglioramento sul già più che ottimo rendimento dell'engine senza acceleratore.

In due righe vi dico come farlo funzionare anche sul vostro engine. Andate al sito della Zend, scaricate lo Zend Optimizer (ricordate che ci sono delle versioni precise per ogni versione del PHP 4) che è gratis e scompattatelo in una cartella a vostra scelta (io ho scelto c:\php4\zend ... giusto per non riempire la root dell'hard disk con troppa roba!). A questo punto prendete questo codice:



```
zend_optimizer.optimization_level=7
zend_extension_ts="C:\php4\Zend\ZendOptimizer.dll"
```

ed inseritelo nel file php.ini... e siete a cavallo! Non dovete fare più niente... è già tutto ok! Il miglioramento in termini di velocità è sensibile e si nota ad occhio! Provare per credere!

Lezioni sulla programmazione PHP a cura di Davide Anastasia

Lezione 8 - Il ciclo IF

Come promesso due lezioni fa (e molti giorni fa... scusate!), in questa lezione parliamo del ciclo IF. Ed iniziamo proprio con questa lezione una serie di tre dedicata ai cicli iterativi. Un argomento interessante ed abbastanza importante, anche se semplice! Ma non tutto ciò che è importante deve essere per forza difficile, vero?

L'istruzione IF-THEN-ELSE è un'istruzione che mette il computer nelle condizioni di decidere che tipo d'operazione compiere in base a delle condizioni da verificare. E' ovvio che utilizza un sistema binario true/false per eseguire le sue scelte! L'istruzione IF (adesso incominciamo a parlare come parla un programmatore!) può essere utilizzata in vari modi.

Il primo metodo consiste nel fare eseguire un'istruzione solo se si verifica una condizione particolare e poi ritornare nella linearità del programma, che viene invece mantenuta se l'IF da come risposta un false. L'immagine semplifica molto il ragionamento.

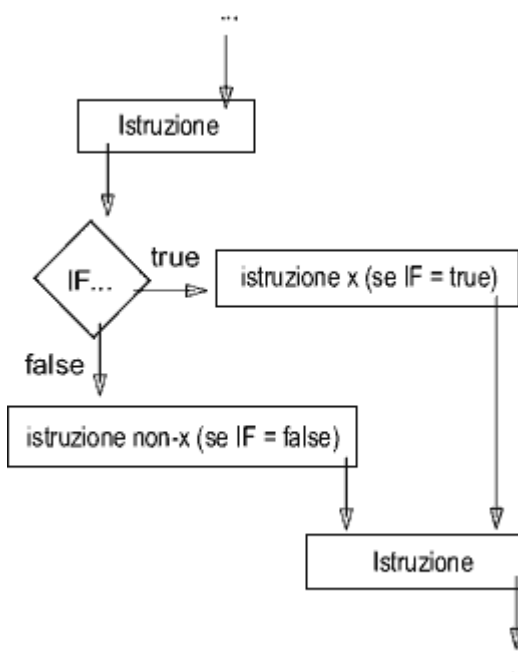
Questo tipo di funzionamento si ottiene inserendo all'interno del listato questa funzione:

```
if (espressione) {  
  istruzioni  
}
```

Facciamo un esempio stupido:

```
if ($a = 4) {  
  $b = $a/2;  
  $a = $b;  
}
```

Con questo semplice listato abbiamo inserito una condizione. La variabile \$a viene divisa per due nel caso in cui il suo valore sia 4 e viene invece lasciata "andare" senza modifiche nel caso in cui il suo valore sia diverso da 4.

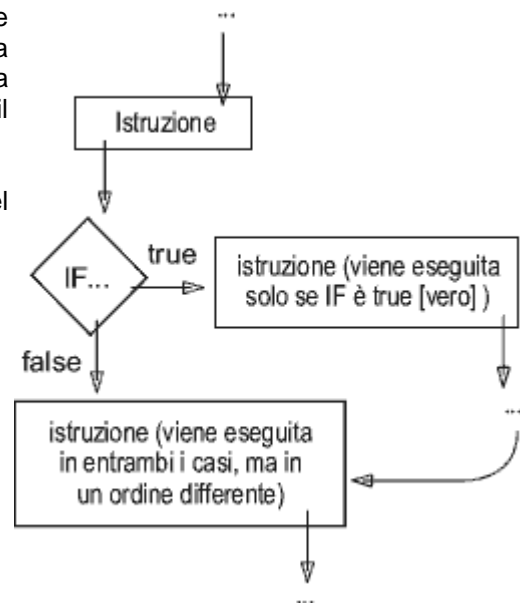


Un altro metodo di usare l'IF è quello classico di effettuare una data operazione x se if = true e di eseguirne un'alternativa se if = false. L'immagine, come al solito spiega meglio il ragionamento.

Questo è il listato base di questa funzione:

```
if (espressione) {  
  istruzioni x  
} else {  
  istruzioni alternative non-x  
}
```

Oltre a questi ci sono altri metodi, che implementano anche l'utilizzo dell'elseif, a cui vi rimando sul manuale del PHP (n'è uscita una nuova versione che aggiunge le



funzioni del nuovo PHP 4 di cui abbiamo parlato la scorsa volta!).

Lezioni sulla programmazione PHP a cura di Davide Anastasia

Lezione 9 - Il ciclo IF - Esempio pratico - Livello: **Principiante**

Adesso vi presento un piccolo esempio pratico sull'uso dell'istruzione IF. Per alcuni di voi che già sanno programmare sembrerà una stupidaggine, ma è utile per chi si avvicina con questo "corso" per la prima volta ad un linguaggio di programmazione.

Ecco l'esempio:

La prima cosa da fare è creare un file **send.php** (o **send.php3**, a seconda di com'è settato il vostro Web Server) in questo modo:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Untitled</title>
</head>
<body>
<font face="Arial" size="2"><b>
<form action="analyzer.php3" method="get">
Immetti un valore per la variabile a ... <input type="Text" name="a"><br>
Immetti un valore per la variabile b ... <input type="Text" name="b"><br>
<input type=submit>
</form>
</b></font>
</body>
</html>
```

Dopo aver creato questo file (che manda i parametri al file analyzer.php3), creiamo il file analyzer.php3:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
<title>Untitled
</head>
<body>

<?
global $a, $b;

if ($a > $b) {
print "a ($a) è più grande di b ($b)";
} elseif ($a == $b) {
print "a ($a) è uguale a b ($b)";
} else {
print "a ($a) è più piccolo di b ($b)";
}
?>

</body>
</html>
```

Lezioni sulla programmazione PHP a cura di Davide Anastasia

Lezione 10 - Il ciclo IF - Esempio pratico - Livello: **Principiante // Medio** Come ti creo un sito con una pagina!!

Ecco qui un altro esempio pratico sull'uso dell'istruzione IF.

Molto spesso ci si trova a combattere con i menù, che spesso in siti di grandi dimensioni (come questo, ad esempio) cambiano spesso. Da una idea nata nel forum (spero di aver interpretato bene la richiesta!) nella prima settimana di Agosto, mi sono messo al lavoro creando un semplice script PHP che potesse da una sola pagina creare un intero sito. Il tutto senza l'utilizzo di Database (che vedremo in seguito!).

L'idea, se pur buona, appare abbastanza semplicistica e può essere la soluzione ottimale per siti di dimensione ridotta (una decina di pagina).

Diamo un'occhiata nel dettaglio:

La prima pagina (quella su cui si compone il sito) è composta da due INCLUDE. Il primo non ha nessuna variabile di controllo (guardando l'allegato ZIP vi renderete conto), mentre il secondo è più interessante e lo riporto qui:

```
<? include($dr.'text'.$c.'.inc') ?>
```

Come vedete all'interno di questo INCLUDE ci sono due variabili: **\$dr** è la variabile definita come la directory in cui avverrà il richiamo del file, mentre la variabile **\$c** definisce il collegamento.

Ma da dove nascono queste variabili?

Queste variabili sono il risultato del primo include, di cui riporto il codice:

```
<table width="150" border="0" cellspacing="0" cellpadding="0">
<tr>
<td><a href="index.php3?s=1&dr=dir1/">Sezione 1</a><br>
<?
/* Controllo della variabile $s */
if ($s == 1) {
echo('<table width="150" border="0" cellspacing="0" cellpadding="0">
<tr>
<td><a href="index.php3?s=1&c=1&dr=dir1/">Link 1-1</a></td>
</tr>
<tr>
<td><a href="index.php3?s=1&c=2&dr=dir1/">Link 1-2</a></td>
</tr>
<tr>
<td><a href="index.php3?s=1&c=3&dr=dir1/">Link 1-3</a></td>
</tr>
</table>');
}
?>
</td>
</tr>
<tr>
<td><a href="index.php3?s=2&dr=dir2/">Sezione 2</a><br>
<?
if ($s == 2) {
echo('<table width="150" border="0" cellspacing="0" cellpadding="0">
<tr>
<td><a href="index.php3?s=2&c=1&dr=dir2/">Link 2-1</a></td>
</tr>
<tr>
<td><a href="index.php3?s=2&c=2&dr=dir2/">Link 2-2</a></td>
</tr>
<tr>
<td><a href="index.php3?s=2&c=3&dr=dir2/">Link 2-3</a></td>
</tr>
</table>');
}
?>
```

```

</td>
</tr>
<tr>
<td><a href="index.php3?s=3&dr=dir3/">Sezione 3</a><br>
<?
if ($s == 3) {
echo('<table width="150" border="0" cellspacing="0" cellpadding="0">
<tr>
<td><a href="index.php3?s=3&c=1&dr=dir3/">Link 3-1</a></td>
</tr>
<tr>
<td><a href="index.php3?s=3&c=2&dr=dir3/">Link 3-2</a></td>
</tr>
<tr>
<td><a href="index.php3?s=3&c=3&dr=dir3/">Link 3-3</a></td>
</tr>
</table>');
}
?></td>
</tr>
</table>

```

Probabilmente ancora non avete capito molto. Mi spiego!

Ogni link (uno in rosso e sottolineato viene posto in risalto. Gli altri sono simili, ma cambiamo per i valori) è caratterizzato da alcune variabili dopo un punto interrogativo (?) che vanno appunto a definire la variabile **\$dr**, la variabile **\$c** e la variabile **\$s**, di cui ancora non abbiamo parlato, ma di cui, se avete guardato questo codice, avete capito il compito. La variabile **\$s** ha il compito di far apparire una parte di codice all'interno di una istruzione IF (al centro di questa lezione!) che appare solo se questa soddisfa certi valori.

Lezioni sulla programmazione PHP a cura di Davide Anastasia

Lezione 11 - Il ciclo WHILE - WHILE / DO

Questa volta parliamo di altri due cicli iterativi o costrutti di controllo che in realtà si possono sommare in un unico controllo. Alla fine spiegheremo le differenze!

Il ciclo **WHILE** serve per eseguire un blocco di istruzioni al verificarsi di una determinata condizione. Ecco qui il codice base:

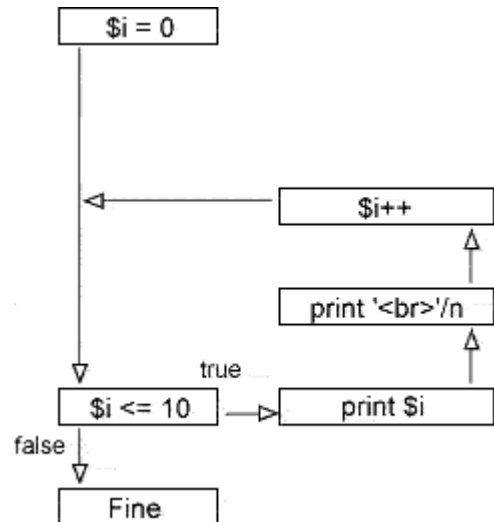
```
while (espressione) {  
  istruzione  
}
```

...ed un piccolo classico esempio:

```
<?  
$i = 0;  
while ($i <= 10)  
{  
  print $i;  
  print "<br>\n";  
  i++;  
}  
?>
```

...che visualizza i primi dieci numeri naturali.

Nell'immagine a lato ho rappresentato il diagramma di flusso di questa istruzione.



Ma esiste anche un altro metodo con cui è possibile realizzare l'istruzione While: è il sistema **WHILE - DO**. La sua sintassi è leggermente differente. Eccola:

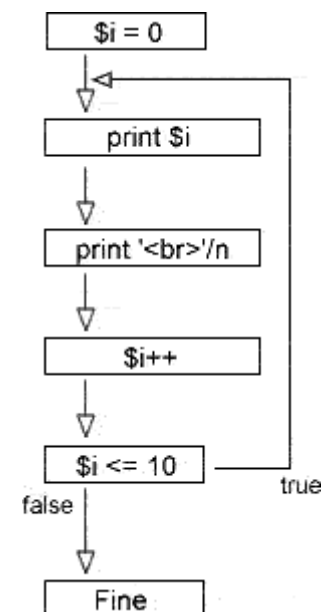
```
do {  
  espressione  
}  
while (istruzione)
```

Ecco qui il precedente esempio riscritto con la nuova sintassi:

```
<?  
$i = 0;  
do  
{  
  print $i;  
  print "<br>\n";  
  i++;  
} while ($i <= 10);  
?>
```

Come si comprende bene dai diagrammi di flusso la differenza dei due cicli sta nel fatto che nel primo ciclo l'istruzione non viene effettuata neanche una volta se al primo "giro" si riceve un valore "false", mentre nel secondo caso le istruzioni vengono comunque eseguite almeno una volta.

Questa differenza, da non trascurare, si trova nella posizione dell'istruzione di controllo. Infatti nel primo caso il controllo avviene a monte mentre nel secondo caso il controllo avviene a valle.



Con questo vi saluto e alla prossima...

Lezioni sulla programmazione PHP a cura di Davide Anastasia

Lezione 12 - Il ciclo FOR

Quello di questa (?) lezione è il ciclo operativo FOR, il più flessibile e potente dei costrutti di controllo del PHP. Non vi faccio il diagramma di flusso (come ho fatto precedentemente) perché il costrutto logico della scrittura è lo stesso del costrutto WHILE, ed infatti il controllo dell'espressione è eseguito a monte. Ecco qui la sintassi del ciclo:

```
for (expr1; expr2; expr3) {  
  istruzione  
}
```

in cui expr1 è una istruzione che viene valutata prima di eseguire il ciclo, expr2 è l'espressione che viene valutata per decidere se il ciclo deve continuare o meno, expr3 è l'operazione che viene eseguita al termine di ogni ciclo.

Facciamo un esempio pratico. Eseguiamo con il ciclo FOR lo stesso "programma" che abbiamo creato con WHILE. Ecco il listato:

```
<?  
  for ($i=0; i<=10; i++)  
  {  
    print  
    $i;  
    print "<br> \n";  
  }  
?>
```

Ecco qui finita la prima parte dedicata ai costrutti di controllo.
Nella prossima parleremo di MySQL e la sua connessione con il PHP...

Lezioni sulla programmazione PHP a cura di Davide Anastasia

Lezione 13 - Installare MySQL

L'utilizzo dei database in ambienti WEB si è molto rafforzato in questo periodo, periodo in cui la gestione di pagine dinamiche che si "autocreano" in relazione alle interrogazioni che provengono dal client si è rafforzata. Sicuramente i primi database sono stati quelli delle BBC, poi i motori di ricerca (con database molto grandi) e poi gli altri. Da ciò sono nati anche linguaggi di scripting come ASP... e soprattutto il PHP che hanno fatto della loro forza l'interfacciamento con i database. Come sappiamo anche il Perl, il linguaggio più usato per i CGI, può eseguire operazioni su database, ma ha bisogno di moduli esterni. Questo non è il caso del PHP, ed io in questa lezione vi insegnerò come installare un database Server sul vostro computer per le vostre prove in locale... senza pagare la sanguisuga Telecom! :-))

Nel sondaggio che ho proposto sul forum la risposta più gettonata nella scelta del database Server è stata MySQL (con 3 voti) che ha vinto largamente sulle altre possibilità (tutte a zero!). Ringrazio quelle tre persone che hanno preso sul serio il mio sondaggio!



Adesso partiamo spiegando come fare...

Do per scontato che Apache ed il supporto PHP 4 (o PHP 3) siano installati correttamente e quindi perfettamente funzionanti!

La prima cosa da fare è modificare nel file php.ini la sezione che riguarda MySQL. Quindi andate in quella sezione e copiate questa configurazione:

```
mysql.allow_persistent = On  
mysql.max_persistent = -1  
mysql.max_links = -1
```

```
mysql.default_port = 3306  
mysql.default_host = localhost  
mysql.default_user = root
```

Gli altri parametri lasciateli vuoti o così come sono.

Se adesso richiamate la pagina `<? phpinfo(); ?>` troverete le informazioni sulla connessione MySQL che avete inserito.

A questo punto non vi rimane che installare il Database server. Andate al sito <http://www.tcx.se/> e trovate la versione di MySQL per Win32. Scaricatela (se sarà possibile la inseriremo nella nostra sezione download).

Quando avete finito il download potrete anche disconnettervi... la connessione non serve più!

A questo punto dovete lanciare l'installazione. Il programma si installa in **c:\mysql** : lasciate stare questa patch... non vi complicate la vita!

Quando avete finito l'installazione per far partire il server dovete avviare il programma **C:\mysql\bin\mysqld.exe** (potete utilizzare l'estensione **--install** con WinNT).

Adesso il database Server è perfettamente funzionante. Per accedere alla shell di MySQL (dove potrete poi mandare i comandi in SQL al DBMS) dovete avviare il programma **mysql.exe**.

Ma c'è anche un'altra soluzione: **phpMyadmin**

phpMyadmin è un tool che permette di eseguire tutte le operazioni eseguibili tramite shell attraverso un'interfaccia grafica di facile comprensione e che non richiede di fatto la conoscenza di SQL. Potrete trovare questo tool sul sito <http://www.phpwizard.com/>

Dopo averlo scaricato, dovrete scompattare lo zip in una cartella del Web Server e modificare il file **config.inc.php3** ... ecco le modifiche da eseguire:

```
$cfgServers[0]['host'] = 'localhost';  
$cfgServers[0]['port'] = ""; // Leave blank for default port  
$cfgServers[0]['adv_auth'] = false;  
$cfgServers[0]['stduser'] = 'root';  
$cfgServers[0]['stdpass'] = "";  
$cfgServers[0]['user'] = 'root';  
$cfgServers[0]['password'] = "";  
$cfgServers[0]['only_db'] = ""; // if set to a db-name, only this db is accessible  
$cfgServerDefault = 1; // default server  
$cfgConfirm = true;
```

```
$cfgPersistentConnections = false;  
$cfgMysqladmin = "C:\mysql\bin\MySqlManager.exe";  
require("italian.inc.php3");
```

Adesso provate ad eseguire il tool tramite il Web Server.

Se tutto è andato bene dovrebbe funzionare, altrimenti ricontrollate tutta la configurazione!

Se continuate ad avere problemi fatemi un post sul forum del PHP... ok?

Alla prossima...