PHP – Un’introduzione

Dispense per il corso di

Linguaggi e Traduttori 2003

Facoltà di Economia
Università di Trento

Paolo Bouquet

Cos'è il PHP?

E' un linguaggio di *scripting server side*

- La differenza tra lato client e lato server sta tutta nel modo e da chi viene interpretata una pagina Web quando essa viene caricata.
- quando un server Web predisposto per il PHP riceve una richiesta dal browser di un client iniziano una serie di operazioni: Il server:

    1. Legge ed individua la pagina sul server.
    2. Esegue le istruzioni PHP contenute all'interno della pagina ed esegue tutti i comandi PHP.
    3. Rimanda la pagina risultante al Browser.

Vantaggi di PHP

- PHP è un linguaggio molto semplice da utilizzare, a cominciare dalla sintassi derivata direttamente da veri linguaggi di programmazione come C/C++, Perl, Java.
- Forse la vera forza del PHP sta nella gestione dei database, con poche righe di codice è possibile accedere qualsiasi database, estrapolare i dati che ci interessano e inserirli nella pagina Web.
- Un altro punto a favore del PHP è la sua natura OpenSource, quindi gratuita.
- Infine il PHP gira su tutti in principali Web server ed in linea di massima non dobbiamo apportare nessuna modifica al codice quando lo spostiamo da un Web server ad un altro.

Primo esempio

Affinché l'interprete PHP riesca a distinguere all'interno del codice il linguaggio da interpretare ed eseguire (PHP) dall'HTML occorre utilizzare dei TAG particolari. Ecco un semplice esempio:

```
<html>
  <body>
    <h1>
      <?php echo “Hello World!!”; ?>
    </h1>
  </body>
</html>
```

Risultato

Al client arriverà il seguente file html:

```
<html>
  <body>
    <h1>Hello World!! </h1>
  </body>
</html>
```

PHP info

Un altro esempio è il comando che ci dà informazioni sulla configurazione del server. Se dentro al file test.php si mette la seguente linea:

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

si ottengono tutte le informazioni sulla configurazione del web server e del php su cui si sta lavorando.

HTTP Server Agent

Lo script:

```
<html>
<body>
Il server agent che stai usando &egrave;:
<center>
<?php echo $_SERVER["HTTP_USER_AGENT"]; ?>
</center>
</body>
</html>
```

HTTP Server Agent

.... produrrà (sul mio PC) il seguente output:

```
<html>
<body>
Il server agent che stai usando &egrave;:
  <center>
    Mozilla/4.0 (compatible; MSIE 5.01;
  Windows NT 5.0)
  </center>
</body>
</html>
```

Variabili

Le variabili sono come dei “cassetti” in cui uno può mettere dei valori.

Ogni variabile ha dunque un nome e un valore.

Il nome di una variabile in PHP:

- è sempre prefissato dal simbolo del dollaro ($)
- contiene caratteri alfanumerici e inizia con una lettera o con il simbolo “_”

In PHP, una variabile è creata automaticamente ogni volta che le si assegna un valore (*dichiarazione implicita*)

Assegnazione di valore a variabili

In PHP, una variabile è creata automaticamente ogni volta che le si assegna un valore (*dichiarazione implicita*)

Per esempio, con il comando:

```
$a = 5;
```

Facciamo due cose contemporaneamente:

- Creiamo una nuova variabile con nome $a
- Le assegnamo il valore “5”

Tipi di dato

I tipi di dato supportati da PHP si distinguono in:

- tipi scalari:
  - numeri (interi e a virgola mobile)
  - stringhe
  - booleani
- tipi compositi:
  - array
  - oggetti

Esempi

Assegnamento di variabili:

- numeri: `$a = 5; $p_greco = 3.14;`
- stringhe: `$a = “Hello world!”;`
- booleani: `$t = TRUE; $f = FALSE;`
- array (esplicito):

```
$giorni = array(“lun”, “mar”, “mer”, “gio”, “ven”, “sab”, “dom”);
```

- array (implicito):

```
$giorni[0] = “lun”;
$giorni[1] = “mar”;
…
$giorni[6] = “dom”;
```

Assegnamento *by value* e *by reference*

E’ possibile assegnare il valore di una variabile a un’altra variabile in due modi diversi:

1. *by value*: “copiando” nella seconda variabile il valore della prima (vedi esempio 9-1):

```
$a = 10;
$b = $a;
```

2. *by reference*: creando un “link” tra il valore della seconda variabile e il valore della prima (vedi esempio 9-2)

```
$a = 10;
$b = &$a;
```

Tipi di base: stringhe

Una stringa può essere specificata in 3 modi:

1. Single quoted (‘)
2. Double quoted (“)
3. Heredoc syntax

Stringhe: single quoted

Le stringhe possono essere introdotte tra singoli apici:

```
<?php echo ‘Stringa'; ?>
```

Le stringhe cosi introdotto possono estendersi su più righe:

```
<?php echo ‘Si possono usare la forma single quoted per
    definire stringhe su più righe'; ?>
```

Si può usare la controbarra (\) per utilizzare simboli speciali:

```
<?php echo ‘Mario disse: “Torno all\’una"';
      echo ‘Hai cancellato C:\\*.*?'; ?>
```

Il singolo apice non permette di introdurre nuove righe:

```
<?php echo ‘Questo non produce: \n una nuova riga'; ?>
```

All’interno di singolo apice le variabili non vengono espanse:

```
<?php $variabili = 23456;
      echo ‘Neanche le $variabili vengono espanse'; ?>
```

Stringhe: double quoted

Le stringhe introdotte tra doppi apici permettono di:

- Utilizzare più simboli di escape:
  - \n : nuova riga
  - \t: tabulazione
  - \r: return
  - \”: carattere doppio apice
- Espandere le variabili:

```
<?php $a = 10;
      echo “Il valore di \$a è $a”; ?>
```

- Concatenere stringhe:

```
<?php $a = “Universita”; $b = “di”; $c = “Trento”;
      echo “$a” . “$b” . “$c”; ?>
```

Stringhe: Heredoc

Esiste anche una terza sintassi per introdurre stringhe, detta Heredoc:

```
<?php
        $a = <<<EOD
        Ecco una stringa
        su più righe
        con sintassi heredoc
        EOD;
 ?>
```

Stringhe: lunghezza

La funzione `strlen` permette di calcolare la lunghezza di una stringa:

```
$a = “prova”;
echo ‘strlen(“prova”) = ‘ . strlen($a);
```

L’output sarà:

```
strlen(“prova”) = 5
```

Array

Il PHP supporta sia gli array scalari che gli array associativi.

In PHP, un array di valori può essere esplicitamente creato definendone gli elementi oppure la sua creazione può avvenire inserendo valori all'interno dell'array, ad esempio:

```
$a = array (“qui", “quo", “qua");
```

crea l'array definendo esplicitamente gli elementi dell'array, al contrario dell'esempio che segue:

```
$a[0] = “qui";
$a[1] = “quo";
$a[2] = “qua";
```

Array

Se invece, per aggiungere elementi ad un array (supponiamo che sia quello precedentemente creato) si utilizzano le parentesi quadre vuote, i dati vengono accodati all'array; ad esempio:

```
$a[] = “pippo";
$a[] = “pluto";
```

In questo caso, l'array si allunga di 2 elementi e risulta:

```
$a[0] = “qui";
$a[1] = “quo";
$a[2] = “qua";
$a[3] = “pippo";
$a[4] = “pluto";
```

Array

Gli array associativi si basano invece su coppie "name-value"; un esempio potrebbe essere:

```
$a = array(
"nome" => "Mario",
"cognome" => "Rossi",
"email" => "mario@rossi.com",
);
```

Array

E' interessante la possibilità della funziona array di annidare le entries, come nell'esempio che segue:

```
$a = array(
"primo" => array(
"nome" => "Mario",
"cognome" => "Rossi",
"email" => "mario@rossi.com",
),
"secondo" => array(
"nome" => "Marco",
"cognome" => "Verdi",
"email" => "mario@verdi.com"));
```

Eseguire su questo array un comando del tipo:

```
<? echo $a["secondo"]["email"]; ?>
```

visualizzerà "mario@verdi.com"

Operatori aritmetici

I principali operatori aritmetici sono i seguenti:

`$a + $b`	Somme	Sum of $a and $b.
`$a - $b`	Sottrazione	Difference of $a and $b.
`$a * $b`	Moltiplicazione	Product of $a and $b.
`$a / $b`	Divisione	Quotient of $a and $b.
`$a % $b`	Resto	Remainder of $a divided by $b.

Operatori di confronto

I principali operatori di confronto sono i seguenti:

- `$a == $b` Uguaglianza (`TRUE` se `$a` è uguale a `$b`)
- `$a === $b` Identità (`TRUE` se `$a` è identico a `$b`, e sono dello stesso tipo (solo PHP 4!)
- `$a != $b` Diseguaglianza (`TRUE` se `$a` non è uguale a `$b`)
- `$a <> $b` Come sopra
- `$a !== $b` Non indentità (`TRUE` se `$a` non è identico a `$b`, o non sono dello stesso tipo (solo PHP 4)
- `$a < $b` Minore di (`TRUE` se `$a` strettamente minore di `$b`)
- `$a > $b` Maggiore di (`TRUE` se `$a` è strettamente maggiore `$b`)
- `$a <= $b` Minore o uguale (`TRUE` se `$a` è minore o uguale a `$b`)
- `$a >= $b` Maggiore o uguale (`TRUE` se `$a` è maggiore o uguale a `$b`)

Operatori logici

I principali operatori logici sono i seguenti:

- `$a and $b` And TRUE se $a e $b sono TRUE.
- `$a or $b` Or TRUE se $a o $b sono TRUE.
- `$a xor $b` Xor TRUE se $a o $b sono TRUE (ma non entrambi).
- `!$a` Not TRUE se $a non è TRUE.
- `$a && $b` And TRUE se $a e $b sono TRUE.
- `$a || $b` Or TRUE se $a o $b sono TRUE.

Nota: I due operatori per congiunzione e disgiunzione hanno diversa precedenza.

Strutture di controllo

- if - endif
- else
- elseif
- while
- for-each
- switch

IF

If permette di eseguire un blocco di codice se avviene (o non avviene) una determinata condizione; la sua sintassi è:

```
if (condizione) {
   statement }
```

Ad esempio:

```
$a = 2;
$b = 2;
if ($a == $b) {
echo "\$a è uguale a \$b e valgono $a.\n";
}
```

endif

If può essere usato anche senza graffe se si utilizza il comando “endif”:

Ad esempio:

```
$a = 2;

$b = 2;

if ($a == $b)

echo "\$a è uguale a \$b e valgono $a.\n";

endif;
```

if-else

“else” viene in completamento di “if”: con if, infatti, stabiliamo che succeda qualcosa all'avverarsi di una condizione; con else possiamo stabilire cosa accade nel caso questa non si avveri. Un esempio potrebbe essere:

```
$a = 2;
$b = 3;
if ($a == $b) {
echo "\$a è uguale a \$b e valgono $a.\n";
} else {
echo "\$a è diversa da \$b.\n\$a vale $a
mentre \$b vale $b.\n";
}
```

if-elseif-else

elseif permette di specificare casi non definiti da "if:

```
if ($a == $b) {
   echo "\$a è uguale a \$b.\n";
   }

elseif ($a != $b) {
   echo "\$a è diversa da \$b.\n";
   }

elseif (!$a) {
   echo "\$a non esiste.\n";
   }

elseif (!$b) {
   echo "\$b non esiste.\n";
   }

else {
   echo “Non so che dire … \n";
   }
```

while

La condizione "while" si comporta esattamente come in C; la sintassi di base è:

```
while (espressione) statement
```

Per esempio:

```
/* $a viene incrementato e visualizzato */
/* finchè il suo valore non supera "5" */
$a = 1;
while ($a <= 5) {
print $a++;
}
```

for

Il "for" permette di eseguire dei loop. La sintassi è la seguente:

```
for (expr1; expr2; expr3) statement
```

Per esempio:

```
for ($a = 0 ; $a <=10 ; $a++) {
print $a;
}
```

visualizzerà i numeri da "0" a "10". Nelle tre espressioni fra parentesi abbiamo definito che:

- $a ha valore "0“ (valutato una sola volta all’inizio del loop);
- $a è minore o uguale a "10";
- $a è incrementata di un'unità.

Quindi, per ogni valore di $a a partire da "0" fino a "10" $a viene visualizzato.

foreach

Il PHP 4 permette di eseguire loop su array in modo semplificato usando il costrutto "foreach“. La sintassi è la seguente:

```
foreach (array_expression as $value) statement
foreach (array_expression as $key => $value)
  statement
```

Per esempio:

```
$arr = array("one", "two", "three");
foreach ($arr as $value) {
      echo "Value: $value<br>\n";
}
```

foreach

```
/* Esempio: key and value */
$a = array ( "one" => 1, "two" => 2, "three" => 3, "seventeen"
   => 17 );
foreach($a as $k => $v) {
print "\$a[$k] => $v.\n"; }


/* Esempio: arrays multi-dimensionali*/
$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";
foreach($a as $v1) {
foreach ($v1 as $v2) {
print "$v2\n";
} }
```

switch-case-default

"Switch" permette di sostituire una serie di "if" sulla stessa espressione e, ovviamente, di agire dipendentemente dal valore di questa:

```
switch($a) {
   case 1:
      print("a è uguale a uno");
      break;
   case 2:
      print("a è uguale a due");
      break;
   default:
      print("a non vale né uno né due");
}
```

FORM HTML

Immaginiamo di avere la seguente form HTML:

```
<FORM action="http://localhost/password.php"
  method="post">

<INPUT type="TEXT" name="utente">

<INPUT type="SUBMIT" name="Invia"
  value="Spedisci">

</FORM>
```

Accedere alle variabili di una FORM HTML

Ci sono vari modi di accedere a variabili da una form HTML. Per esempio, se si utilizza il metodo POST, ci sono i seguenti modi:

```
<?php
// Available since PHP 4.1.0
print $_POST['utente'];
print $_REQUEST['utente'];
import_request_variables('p', 'p_');
print $p_utente;

// Available since PHP 3.
print $HTTP_POST_VARS['utente'];

// Available if the PHP directive register_globals = on. As of
// PHP 4.2.0 the default value of register_globals = off.
// Using/relying on this method is not preferred.
print $utente; ?>
```

Esempio

```
<?php


if (isset($_POST[utente]) {
      echo "Benvenuto <b>$_POST[utente]</b>,
  divertiti";
}
else {
echo “Torna indietro e inserisci il nome
  utente!";
}


?>
```

Apertura file

Quando si apre un file, bisogna specificare le seguenti informazioni:

- Se lo si vuole aprire in sola lettura, in sola scrittura, o in lettura e scrittura
- Se si vuole che il puntatore sia posizionato all'inizio o alla fine del file
- Cosa fare se il file non esiste

Apertura file

In PHP, un file si apre con il comando `fopen(filename, mode)`, dove il mode può essere uno dei seguenti:

Mode	Descrizione
‘r’	Solo lettura, puntatore all’inizio del file
‘r+’	Lettura e scrittura, puntatore all’inizio del file
‘w’	Solo scrittura, puntatore all’inizio del file. Se il file non esiste, tenta di crearlo
‘w+’	Lettura e scrittura, puntatore all’inizio del file. Se il file non esiste, tenta di crearlo
‘a’	Solo scrittura, puntatore alla fine del file. Se il file non esiste, tenta di crearlo
‘a+’	Lettura e scrittura, puntatore alla fine del file. Se il file non esiste, tenta di crearlo

Lettura da file

In PHP, un file si legge con il comando `fread(risorsa,lunghezza)`, dove la lunghezza dice fino a che punto si vuole leggere il file.

Per esempio:

```
<?php
$nomefile = "..\Esempi-XML\sms15.xsl";
$handle = fopen ($nomefile, "r");
$contenuto = fread ($handle, filesize $nomefile));
fclose ($handle);
?>
```

Scrittura su file

In PHP, un file si scrive con il comando `fwrite(risorsa, stringa)`.

Esempio di scrittura in cima al file:

```
<?php

$nomefile="gatta.txt";

$testo="Tanto va la gatta al lardo che ci
  lascia lo zampino";

$handle = fopen($nomefile, 'w');

fwrite($handle, $testo);

fclose($handle);

?>
```

```
fwrite($handle, “<HTML>\n”);
fwrite($handle, “<BODY>\n”);
fwrite($handle, “<TABLE>\n”);
fwrite($handle, “<TR>\n”);
fwrite($handle, “<TD>Nome</TD><TD>$_POST[nome]</TD>\n”);
fwrite($handle, “</TR>\n”);
fwrite($handle, “</TABLE>\n”);
fwrite($handle, “</BODY>”);
fwrite($handle, “</HTML>”);
```

Scrittura su file - 2

Esempio di scrittura in fondo al file (append):

```
<?php

$nomefile="gatta.txt";

$testo="Tanto va la gatta al lardo che ci
  lascia lo zampino";

$handle = fopen($nomefile, ‘a');

fwrite($handle, $testo);

fclose($handle);

?>
```