

# Corso di Web Programming

## 8. PHP Parte I

Paolo Milazzo

Dipartimento di Informatica, Università di Pisa

<http://www.di.unipi.it/~milazzo>

[milazzo@di.unipi.it](mailto:milazzo@di.unipi.it)

Corso di Laurea in Informatica Applicata

A.A. 2010/2011

# Sommario

- 1 Introduzione a PHP
- 2 Configurazione di PHP
- 3 La sintassi del linguaggio
  - Variabili ed espressioni
  - Comandi
  - Stringhe
  - Array
  - Funzioni
  - Inclusione di file

# Introduzione a PHP (1)

- PHP è un linguaggio di programmazione “general-purpose” che però viene utilizzato in pratica solo per creare siti web dinamici
- PHP è un acronimo ricorsivo che sta per “PHP: Hypertext Preprocessor”
- E’ un linguaggio interpretato, non compilato
- E’ un linguaggio a cui possono essere aggiunte estensioni che realizzano funzionalità specifiche
  - ▶ e.g. aspetti di sicurezza/crittografia, interazione con altri siti web, ecc...

## Introduzione a PHP (2)

- PHP viene eseguito (interpretato) da un server web
  - ▶ è un linguaggio di scripting server-side
- Il codice PHP viene usato per generare dinamicamente i documenti HTML che il client deve ricevere e visualizzare nel browser
  - ▶ In un sito web statico (non-PHP) il documento HTML viene preso dal file system e inviato direttamente al client
  - ▶ In un sito web dinamico (tipo PHP) i file del web server sono passati all'interprete PHP che genera il documento HTML da inviare al client
- Quindi: l'output di un "programma" PHP è un documento HTML
- Il codice PHP può fare quello che gli pare per creare il documento HTML
  - ▶ Ad esempio: accedere a un database, consultare altri siti web, ecc....

## Introduzione a PHP (3)

- I file PHP sono strutturati come un documento HTML
  - ▶ ma contengono sezioni di codice PHP delimitate da `<?php` e `?>`
- L'interprete PHP effettua il parsing del file (lo scandisce) e sostituisce le sezioni di codice PHP con il codice HTML risultante dalla sua esecuzione
- Tutto ciò che è al di fuori del tag `<?php . . . . ?>` viene ignorato dall'interprete e dato in output così com'è
- Solitamente l'estensione usata per i file PHP è `.php`
- Per “eseguire” lo script bisogna necessariamente passare tramite il web-server:
  - ▶ non è sufficiente aprire il file con il browser
  - ▶ nel caso di un'utilizzo “locale” bisogna avere installato un server HTTP + PHP, e bisogna aprire nel browser un indirizzo tipo:  
`http://localhost/miapagina.php`

## Introduzione a PHP (4)

Un semplice esempio di file PHP (il solito Hello World):

```
<html>
<head><title>Hello World in PHP</title></head>
<body>

<?php
    print "<p>Hello, World!</p>";
    print "<p>Questo e' un frammento di codice PHP.
        Nota che fa parte di un documento HTML standard!</p>;
?>

</body>
</html>
```

E il corrispondente output HTML generato dall'interprete PHP:

```
<html>
<head><title>Hello World in PHP</title></head>
<body>

<p>Hello, World!</p>
<p>Questo e' un frammento di codice PHP.
Nota che fa parte di un documento HTML standard!</p>

</body>
</html>
```

## Introduzione a PHP (5)

- In realtà esistono due alternative per “scrivere” l’output in PHP: `echo` e `print`
  - ▶ `echo` è un comando nativo del linguaggio (forse più veloce di `print`)
  - ▶ `print` è una funzione di libreria, quindi un’espressione (può essere usata in espressioni condizionali: `(a>b?print(a):print(b))`)
- L’uso delle parentesi attorno al messaggio da stampare è opzionale

## Introduzione a PHP (6)

- E' importante sottolineare che l'output di PHP è un documento HTML
- Il client riceverà tale output senza vedere il codice PHP che lo ha originato
- In generale il client non percepisce la differenza tra un sito web statico e un sito web dinamico
  - ▶ in entrambi i casi il client riceve un documento HTML e lo visualizza nel browser



# Configurazione di PHP (1)

- L'interprete PHP può essere configurato in maniera molto dettagliata
  - ▶ Elevato numero di opzioni settabili e possibili estensioni
- I settaggi di default sono specificati nel file `php.ini`
  - ▶ la directory in cui si trova questo file dipende dall'installazione
- La configurazione corrente di php su un web server può essere visualizzata chiamando la funzione PHP `phpinfo()`
- L'output di questa funzione è un documento HTML (intero) contenente una tabellona che riassume tutte le opzioni ed estensioni configurate
- Utile per il mantenimento di un'installazione di PHP e per il debugging di applicazioni
  - ▶ tipicamente si usa appena si è installato PHP (o un'estensione) sul proprio computer per vedere se è tutto OK

## Configurazione di PHP (3)

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

### PHP Version 5.2.3-1ubuntu6.3



<b>System</b>	Linux grenadine 2.6.18-xenU #3 SMP Thu Jan 10 15:56:11 CET 2008 i686
<b>Build Date</b>	Jan 10 2008 09:24:13
<b>Server API</b>	Apache 2.0 Handler
<b>Virtual Directory Support</b>	disabled
<b>Configuration File (php.ini) Path</b>	/etc/php5/apache2
<b>Loaded Configuration File</b>	/etc/php5/apache2/php.ini
<b>Scan this dir for additional .ini files</b>	/etc/php5/apache2/conf.d
<b>additional .ini files parsed</b>	/etc/php5/apache2/conf.d/curl.ini, /etc/php5/apache2/conf.d/gd.ini, /etc/php5/apache2/conf.d/mysql.ini, /etc/php5/apache2/conf.d/mysqli.ini, /etc/php5/apache2/conf.d/pdo.ini, /etc/php5/apache2/conf.d/pdo_mysql.ini, /etc/php5/apache2/conf.d/pspell.ini, /etc/php5/apache2/conf.d/tdy.ini
<b>PHP API</b>	20041225
<b>PHP Extension</b>	20060613
<b>Zend Extension</b>	220060519
<b>Debug Build</b>	no
<b>Thread Safety</b>	disabled
<b>Zend Memory Manager</b>	enabled
<b>IPv6 Support</b>	enabled

# Overview

- La sintassi di PHP è simile a quella di C, C++ e Java (ma senza tipi)
- Tutto il codice PHP va messo all'interno di tag `<?php .... >`
  - ▶ il tag `<?php .... >` può essere usato più volte all'interno dello stesso file
  - ▶ tutto ciò che si trova al di fuori di tali tag viene dato in output così com'è
- Come C, C++ e Java:
  - ▶ I comandi sono separati da `;`
  - ▶ PHP è case-sensitive (distingue tra maiuscole e minuscole)
- PHP supporta commenti a singola linea e multilinea con la stessa sintassi di C++ e Java:

```
<?php
// Questo e' un commento che verra' trascurato
// dall'interprete PHP....

/* .... tanto quanto questo commento,
   che puo' occupare tranquillamente
   piu' linee
*/
?>
```

# Le variabili (1)

- Come in ogni linguaggio di programmazione imperativo, in PHP si ci sono le variabili
- Differentemente dal solito il nome di una variabile deve iniziare con \$, seguito da una lettera o un \_
  - ▶ Esempi: \$username, \$color, \$i, \$\_nome....
- Non esiste un costrutto di dichiarazione di variabili
  - ▶ Ossia, **non c'è bisogno di dichiarare una variabile prima di iniziare a usarla!**
- I tipi di dato è come se non esistessero!
  - ▶ Una variabile **può essere assegnata a valori di tipo diverso in momenti diversi**
- L'operatore di assegnamento di una variabile è = (come in C, C++ e Java)

## Le variabili (2)

- L'uso di \$ all'inizio delle variabili consente di definire “variabili variabili”, ad esempio:

```
$a = b;  
$b = 1;  
echo $$a;
```

produce come risultato 1 !!!

- Potente, ma pericoloso:
  - ▶ può indurre a errori di programmazione
  - ▶ espone a rischi di sicurezza se si usano dati ottenuti dall'utente

## Le variabili (3)

- Lo scope di una variabile in PHP è la pagina stessa
- Ogni variabile esiste solo per lo script (pagina) in cui è definita e alla fine dell'esecuzione viene distrutta
- Si può verificare se una variabile è stata inizializzata con la funzione `isset`:

```
isset($var)
```

restituisce `true` se `$var` è stata precedentemente assegnata, `false` altrimenti

- E' possibile distruggere una variabile precedentemente assegnata tramite la funzione `unset`:

```
unset($var)
```

# Le espressioni

- Gli operatori che si possono usare nelle espressioni sono sostanzialmente gli stessi di C/C++/Java:
  - ▶ Operatori aritmetici: +, -, \*, /, %, ++, --
  - ▶ Operatori di confronto: ==, !=, <, >, <=, >=
  - ▶ Operatori logici: &&, ||, !

# Comandi

Anche i comandi del linguaggio sono come in C/C++/Java:

- Comandi di selezione:

- ▶ `if ($trovato) {.....}`
- ▶ `if ($trovato) {.....} else {.....}`
- ▶ `switch ($msg) {`
  - `case "a" : .....`
    - `break;`
  - `case "b" : .....`
    - `break;`
  - `default : .....``}`

- Comandi di iterazione:

- ▶ `while (!$trovato) {.....}`
- ▶ `do {.....} while (!$trovato)`
- ▶ `for ($i=0;$i<10;$i++) {.....}`



## Le stringhe (1)

- In PHP le stringhe possono essere delimitate da virgolette (ossia ") o da apici (ossia ')
- Quando si usano gli apici (') la stringa viene interpretata letteralmente (esattamente com'è scritta)
- Quando si usano le virgolette (") le eventuali variabili contenute nella stringa vengono sostituite con il loro valore

```
// esempio sull'uso delle virgolette e degli apici
$food = "patate"
$virgolette = "Io adoro mangiare $food.";
$apici = 'Io invece adoro mangiare $food.';
print "<p>$virgolette</p>";
print "<p>$apici</p>";
```

Risultato:

Io adoro mangiare patate.

Io invece adoro mangiare \$food.

## Le stringhe (2)

- L'operazione di concatenazione di stringhe è il punto: .

```
$nome = "Mario";  
$cognome = "Rossi";  
$nomeintero = $nome . " " . $cognome;
```

- Altre operazioni sulle stringhe includono:
  - ▶ `strlen($s)` – restituisce la lunghezza della stringa `$s`
    - ★ `strlen("Hello world!")` restituisce 12
  - ▶ `strpos($s1,$s2)` – restituisce la posizione di `$s2` in `$s1`, o `false` se non presente
    - ★ `strpos("Hello world!","world")` restituisce 6
  - ▶ `trim($s)` – elimina gli spazi all'inizio e alla fine di `$s`
    - ★ `trim(" Hello world! ")` restituisce "Hello world!"
  - ▶ ....

# Gli array (1)

- PHP offre due tipi di array (in realtà il secondo è un caso particolare del primo):

- ▶ **array numerici** come in C/C++/Java che associano valori a indici numerici

chiavi	1	2	3	4
valori	"pippo"	"pluto"	"paperino"	"topolino"

- ▶ **array associativi** che associano valori a **chiavi**, possibilmente non numeriche

chiavi	"a"	"b"	"c"	"d"
valori	"pippo"	"pluto"	"paperino"	"topolino"

- Il costrutto per la creazione di un nuovo array è `array(...)`, e per associare una chiave ad un valore si usa `=>`:

- ▶ Array numerico:  
`$an = array("pippo","pluto","paperino","topolino");`
- ▶ Array associativo: `$aa = array("a" => "pippo", "b" => "pluto", "c" => "paperino", "d" => "topolino");`

## Gli array (2)

- L'accesso all'array può avvenire tramite l'indice o la chiave:
  - ▶ `echo $an[0];`
  - ▶ `echo $aa["a"];`
- Un array può anche essere creato assegnandone un elemento:
  - ▶ `$an[0] = "pippo";`
  - ▶ `$aa["a"] = "pippo";`
- Nel caso di un array numerico, se non si specifica l'indice il nuovo elemento verrà aggiunto in coda all'array (l'array è creato se non esiste):
  - ▶ `$an[] = "pippo";`
- Per iterare su tutti gli elementi di un array si può usare il costrutto `foreach`:
  - ▶ `foreach ($myarray as $var) { ... }`
  - ▶ `foreach ($myarray as $index => $var) { ... }`che scorre l'array `$myarray` associando alla variabile `$var` i valori contenuti (e alla variabile `$index` le chiavi corrispondenti)

## Gli array (3)

Esempio: mostra i personaggi contenuti in un array in una tabella HTML:

```
<table border="1">
  <tr><th>Personaggio </th></tr>
  <?php
    $pers = array("pippo","pluto","paperino","topolino");
    foreach ($pers as $nome) {
      print "<tr><td>$nome</td></tr>\n";
    }
  ?>
</table>
```

Con array associativo:

```
<table border="1">
  <tr><th>Chiave</th><th>Personaggio </th></tr>
  <?php
    $pers = array("a"=>"pippo","b"=>"pluto",
                  "c"=>"paperino","d"=>"topolino");
    foreach ($pers as $idx => $nome) {
      print "<tr><td>$idx</td><td>$nome </td></tr>\n";
    }
  ?>
</table>
```

## Gli array (4)

- Alcune funzioni utili per lavorare con gli array:
  - ▶ `count($a)` – restituisce il numero di elementi di un array (simile al metodo `length` di Java e JavaScript)

```
if (count($pers)=0) {  
    echo "Errore, l'array e' vuoto!";  
}
```

- ▶ `in_array(val, $a)` – verifica se il valore `val` è presente nell'array

```
$nome = "pluto";  
if (in_array($nome, $pers))  
    echo "Il personaggio $nome e' presente nell'array";
```

- ▶ `array_merge($a1, ..., $aN)` – unisce tutti gli array elencati in un unico array

```
$pers1 = array("pluto", "paperino");  
$pers2 = array("topolino", "paperone");  
$pers = array_merge($array1, $array2);
```

- ▶ `sort($a)` e `rsort($a)` – ordinano l'array passato come parametro (modificandolo) in ordine crescente e decrescente

## Gli array (5)

- Alcune funzioni utili per lavorare con gli array:

- ▶ `explode(sep, str)` e `implode(sep, $a)` – convertono una stringa in un array e viceversa per mezzo di un carattere separatore:

```
$personaggi = "pippo topolino paperino paperone";  
$pers_array = explode(" ", $personaggi);  
$personaggi_virgola = implode(",", $pers_array);  
echo $personaggi_virgola;
```

Il risultato è `pippo,topolino,paperino,paperone`.

- ▶ `print_r($a)` – stampa gli elementi dell'array come un'unica stringa `$a` (se associativo, stampa anche le chiavi)

# Funzioni (1)

- Come in JavaScript, le funzioni si definiscono tramite la parola chiave `function`.
- Le funzioni possono prevedere argomenti di cui non è necessario specificare il tipo.
- Le funzioni possono prevedere un valore di ritorno da restituire tramite `return`.
- Le variabili usate dentro a una funzione sono locali alla funzione stessa.
- Di default una funzione non vede le variabili create fuori della funzione stessa. Per poter accedere a una variabile “globale” bisogna usare la dichiarazione `global` dentro la funzione

```
$a = 22;  
function f() {  
    global $a;  
    echo $a;  
}
```



## Funzioni (2)

Esempi di funzioni:

- Una funzione che restituisce un valore:

```
function add($x,$y) {  
    return $x+$y;  
}
```

- Una funzione senza valori di ritorno che stampa l'intestazione della pagina web:

```
function intestazione() {  
    print "<h1>Il mio sito</h1>\n";  
    print "<hr>";  
    print date("l F d, Y");  
    print "<hr>";  
}
```

- Chiamate di funzioni:

```
$somma = add(5,10);  
intestazione();
```

# Espressioni Regolari

- PHP mette a disposizione diverse librerie di funzioni per lavorare con espressioni regolari.
- La libreria più importante è la PCRE library (Perl-Compatible Regular Expression), le cui funzioni principali sono le seguenti:
  - ▶ `preg_match (regex, str)` – restituisce 1 o 0 a seconda che l'espressione regolare `regex` (espressa come stringa) venga matchata almeno una volta nella stringa `str`
    - ★ Esempio: `preg_match('/[ab]+/', 'ababcbb')` restituisce 1;
  - ▶ `preg_match_all (regexp, str, $matches)` – restituisce il numero di match di `regex` in `str`, e inserisce tutti i match nell'array `$matches` passato come parametro
    - ★ Esempio: `preg_match('/[ab]+/', 'ababcbb', $matches)` restituisce 1; inoltre `$matches[0] = abab` e `$matches[1] = bb`
  - ▶ `preg_replace (regex, str1, str2)` – restituisce una stringa in cui tutti i match di `regex` in `str2` sono sostituiti con `str1`
    - ★ Esempio: `preg_replace('/[ab]+/', 'zzz', 'ababcbb')` restituisce `zzzczzz`;

# Inclusione di file (1)

- Per condividere e riutilizzare porzioni di script e funzioni si usa frammentare il codice in più file in modo da poterlo includere quando serve.
- In PHP abbiamo due funzioni che importano il codice da un file esterno:
  - ▶ `include(nomefile)` – importa il codice contenuto nel file indicato, senza interrompersi se il file non esiste
  - ▶ `require(nomefile)` – importa il codice contenuto nel file indicato, interrompendo l'esecuzione se il file non esiste
- Le due funzioni possono essere usate ovunque nel codice, ma comunque prima dei riferimenti fatti a variabili o funzioni contenuti nel file da includere

## Inclusione di file (2)

- Le funzioni per l'inclusione di files consentono di realizzare le "Server Side Includes": porzioni di documenti HTML comuni a più pagine (e.g. menu, intestazioni di pagine, ecc...)

```
<html>
<body>
<div class="leftmenu">
  <?php include("menu.php"); ?>
</div>

<h1>Welcome to my home page.</h1>
<p>Some text.</p>
</body>
</html>
```

dove menu.php potrebbe essere:

```
<a href="/default.php">Home</a>
<a href="/tutorials.php">Tutorials</a>
<a href="/references.php">References</a>
<a href="/examples.php">Examples</a>
<a href="/about.php">About Us</a>
<a href="/contact.php">Contact Us</a>
```