Introduzione a PHP

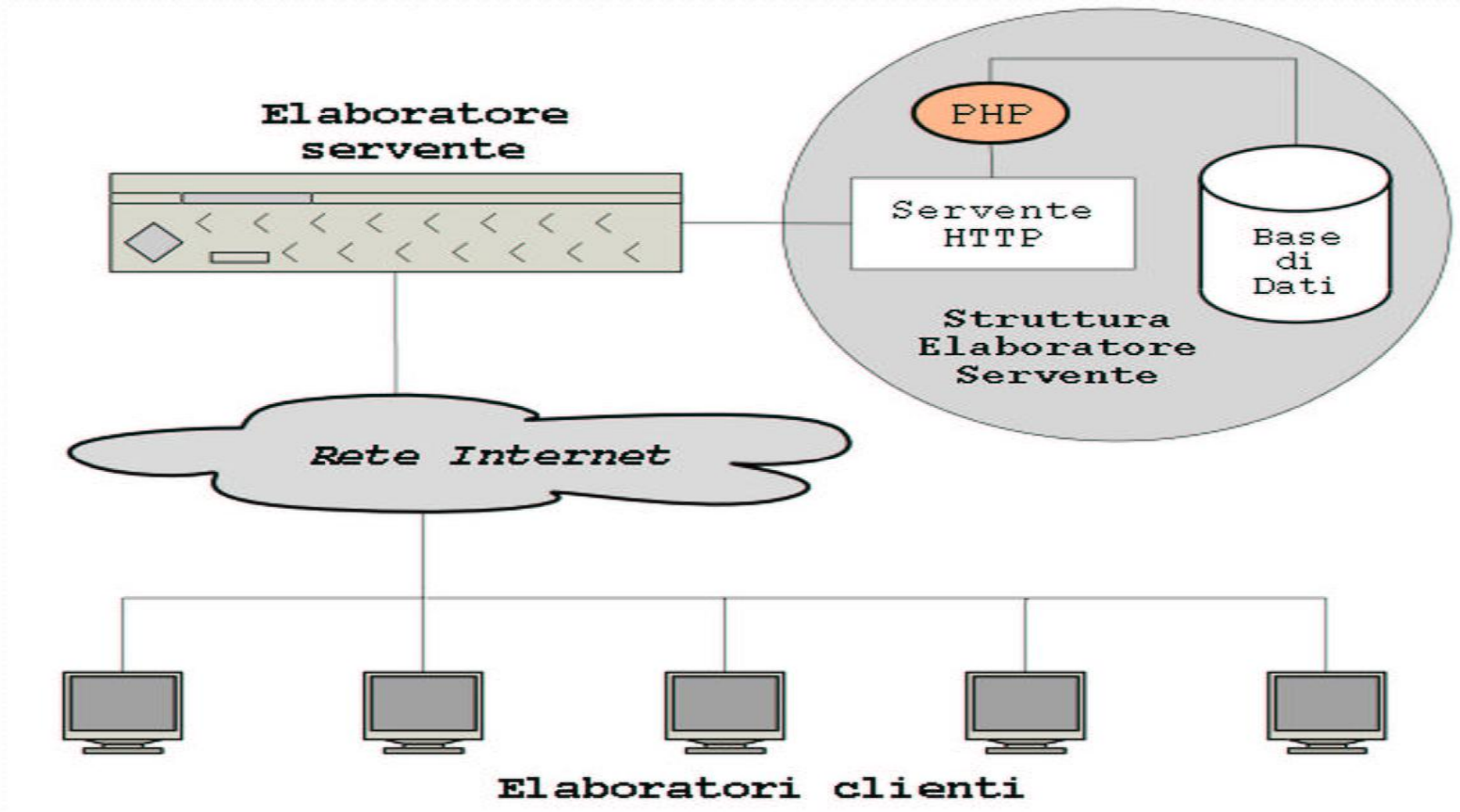
Prof. Francesco Accarino

Cos'è PHP

- «linguaggio script»: il PHP è un vero e proprio linguaggio di programmazione, è importante rendersi conto che l'HTML, ad esempio, non è un linguaggio di programmazione ma un linguaggio di descrizione e formattazione del testo.
- il PHP è un linguaggio interpretato Lato Server.

Cos'è PHP

Il server elabora tramite l'interprete PHP gli script i quali generano le pagine HTML che vengono fornite al cliente tramite il protocollo HTTP.



Modi per uscire dalla modalità HTML

- Esistono vari modi per inserire codice PHP al'interno di una pagina html. I due modi che funzionano sempre sono:

<?php. . .?>

<script language="php">. . .</script>)

Separazione delle istruzioni

- Le istruzioni sono separate come nel C

- ogni istruzione termina con un punto e virgola.

- Il tag di chiusura (?>) implica anche la fine di un'istruzione, perciò le seguenti sono equivalenti:

<?php echo "Questo &grave un test"; ?>

<?php echo "Questo &grave un test" ?>

Si può intervallare HTML e PHP

```
<?php //questo è PHP
if ($expression) { ?>
/*questo è una riga HTML */
<strong>Questa è vera.</strong>
<?php
} else { ?>
<strong>Questa è falsa.</strong>
<?php } ?>
```

Variabili

PHP supporta otto tipi primitivi:

- Quattro scalari:

  boolean, integer, double string

- Due tipi composti:

  array, object

- Due tipi speciali:

  resurce, NUL

Variabili

Il tipo di una variabile non è solitamente stabilito dal programmatore (Come in C o Java), ma è deciso da

PHP in base al contesto in cui la variabile è usata.

Esempio:

```
<?php
$booleano = TRUE; // un boolean
$stringa = "Ciao"; // una string
$int = 12; // un integer
?>
```

Si può forzare la conversione di una variabile in un determinato tipo con l'operatore di cast (come in C o Java) o con la funzione `settype()`; da notare che la stessa variabile può assumere valori diversi in certe situazioni, in base al tipo che è in quel contesto.

Boolean: Può assumere i due valori TRUE o FALSE.

Sintassi: Per introdurre una variabile booleana basta assegnarle il valore TRUE o FALSE (maiuscole o minuscole)

```
<?php
$test = True; // assegna il valore TRUE alla
  variabile $test
?>
```

Conversione in boolean: Per convertire esplicitamente un valore in boolean si usano gli operatori di cast (bool) o (boolean), i seguenti valori sono considerati FALSE:

- il boolean FALSE
- l'integer 0 (zero)
- il float 0.0 (zero)
- la stringa vuota e la stringa "0"
- un array con zero elementi
- un oggetto con zero variabili membro
- NULL

Tutti gli altri valori sono considerati TRUE

Integer

Un Integer è un numero intero di lunghezza dipendente dal sistema operativo, in genere 32 bit con segno, può essere specificato in base 10, 16, 8 eventualmente preceduto dal segno. I numeri che iniziano con una cifra diversa da 0 vengono considerati decimali, se iniziano con 0 ottali, se iniziano con 0x esadecimali

```
<?php
$a = 1234; // numero decimale
$a = -123; // numero negativo
$a = 0123; /* numero ottale (equivalente a
              83 decimale)*/
$a = 0x1A; /* numero esadecimale
              (equivalente a 26 decimale)*/
?>
```

Divisione

Non esiste in PHP un operatore di divisione intera, il risultato è sempre float, l'operatore di cast `(int)` lo tronca, per arrotondare al valore più vicino si usa la funzione `round()`.

```
<?php
var_dump(25/7); //loat(3.5714285714286)
var_dump((int) (25/7)); // int(3)
var_dump(round(25/7)); // float(4)
?>
```

String Una stringa costante può essere specificata

Tra apici (virgolette semplici)

Se volete inserire un apice nella stringa dovete farlo precedere da backslash (`\'`) , il carattere backslash può essere inserito raddoppiandolo(`\\`). In questo formato non possono essere inseriti caratteri speciali come `\n` o `\t`.

Tra virgolette

Se la stringa è racchiusa tra virgolette (") al suo interno si possono usare i caratteri speciali del linguaggio C, in particolare:

```
\n linefeed (LF or 0x0A (10) in ASCII)
\r carriage return (CR or 0x0D (13) in ASCII)
\t horizontal tab (HT or 0x09 (9) in ASCII)
\\ backslash
\$ dollar sign
\" double-quote
```

La caratteristica più importante delle stringhe racchiuse tra virgolette è che le variabili vengono espanse, cioè sostituite con il loro valore

Stringhe

Accesso ai singoli caratteri di una stringa

I caratteri di una stringa si comportano come gli elementi di un vettore, il primo carattere ha indice 0, l'indice in PHP 4 va racchiuso tra parentesi graffe `{}` anche se si possono usare quelle quadre degli array (sintassi deprecata);

```
<?php
$str = 'Stringa di prova';
$first = $str{0};// Prende il primo carattere di una stringa
$last = $str{strlen($str)-1}; // Prende l'ultimo carattere
di una stringa
?>
```

Funzioni e operatori utili per le stringhe

Le stringhe possono essere concatenate con l'operatore punto '`.`', da notare che il + usato per esempio da Java non funziona. Ci sono inoltre molte funzioni utili per modificare le stringhe.

Conversione in stringa

Si può convertire un valore in stringa usando il cast `(string)` o la funzione `strval()`.

Funzioni per le stringhe

strlen(stringa)

verifica la lunghezza della stringa, cioè il numero di caratteri che la compongono. Restituisce un numero intero.

trim(stringa)

elimina gli spazi all'inizio e alla fine della stringa. Restituisce la stringa modificata.

ltrim(stringa)

elimina gli spazi all'inizio della stringa. Restituisce la stringa modificata.

rtrim(stringa)

elimina gli spazi alla fine della stringa. Restituisce la stringa modificata.

substr(stringa, intero [, intero])

restituisce una porzione della stringa, in base al secondo parametro (che indica l'inizio della porzione da estrarre), e all'eventuale terzo parametro, che indica quanti caratteri devono essere estratti. Se il terzo parametro non viene indicato, viene restituita tutta la parte finale della stringa a partire dal carattere indicato.

Funzioni per le stringhe

strpos(stringa, stringa)

cerca la posizione della seconda stringa all'interno della prima. Ad esempio: strpos('Lorenzo', 're') restituisce 2, ad indicare la terza posizione. Restituisce un intero che rappresenta la posizione a partire da 0 della stringa cercata. Se la seconda stringa non è presente nella prima, restituisce il valore booleano FALSE. La funzione stripos() fa la stessa ricerca senza tenere conto della differenza fra maiuscole e minuscole.

strstr(stringa, stringa)

cerca la seconda stringa all'interno della prima, e restituisce la prima stringa a partire dal punto in cui ha trovato la seconda. strstr('Lorenzo', 're') restituisce 'renzo'. Restituisce una stringa se la ricerca va a buon fine, altrimenti il valore booleano FALSE. La funzione stristr() funziona allo stesso modo ma non tiene conto della differenza fra maiuscole e minuscole.

strtolower(stringa)

converte tutti i caratteri alfabetici nelle corrispondenti lettere minuscole. Restituisce la stringa modificata.

strtoupper(stringa)

converte tutti i caratteri alfabetici nelle corrispondenti lettere maiuscole. Restituisce la stringa modificata. **E mlote altre ancora**

Array

Una array è una specie di "super-variabile" contentente una pluralità di valori invece di uno solo.

Poniamo di voler scrivere una sorta di lista di amici utilizzando appunto una array.

Uno dei modi per **scrivere la nostra array** è il seguente:

```
<?
$amici = array('luca', 'jacopo', 'felice', 'peppo');
?>
```

In sostanza abbiamo dato un nome alla nostra array ed abbiamo inserito i vari elementi tra parentesi (dopo l'indicazione di "array") divisi da una virgola.

PHP associa automaticamente a ciascuno dei valori che abbiamo elencato un indice numerico, a partire da 0. Quindi, in questo caso, 'luca' assumerà l'indice 0, 'jacopo' l'indice 1, e così via.

Array

Il risultato sarebbe stato lo stesso se avessimo scritto la nostra array in questa maniera:

```
<?
   $amici[0] = 'luca';
   $amici[1] = 'jacopo';
   $amici[2] = 'felice';
   $amici[3] = 'peppo';
   ?>
```

Come è evidente nel primo modo l'indice veniva assegnato in automatico da PHP, mentre in questo secondo caso lo abbiamo esplicitato noi.

Array

l'indice può essere anche di tipo stringa.

In questo caso si parla di array associativi.

Facciamo un esempio di una array contenente i dati di un ipotetico cliente:

```
<?
$cliente['azienda'] = 'Tiscali';
$cliente['nome'] = 'Roberto';
$cliente['cognome'] = 'Siena';
?>
```

Quando l'indice è di tipo stringa si usano gli apici anche all'interno della parentesi quadra!

Lavorare con gli array

Per riferirsi ad un singolo elemento dell'array

```
print "Ciao " . $amici[1]; //questa linea di codice stampa "Ciao jacopo"
```

Per aggiungere un nuovo elemento alla nostra array (elemento che verrà posizionato in fondo agli altri già presenti) si usa:

$amici[ ] = 'daniele'; //questa linea aggiunge l'elemento "daniele" alla array

Qualora volessimo sostituire uno degli elementi.

Poniamo ad esempio di voler sostituire l'elemento "felice" con un nuovo elemento "marcello". Ecco come fare:

```
$amici[2] = 'marcello'; //questa linea sostitisce l'elemento "felice" con "marcello"
```

Array di grandi dimensioni

Nel nostro esempio abbiamo usato una array di dimensioni molto ridotte, ma in realtà capità spesso di avere di fronte array molto lunghe e può essere utile conscere il numero degli elementi che la compongono, in questo caso soccorre la funzione count() che useremo così:

$max_num = count($amici);

Questa funzione è molto utile se vogliamo usare un ciclo for.

```
<?
$amici = array('luca', 'jacopo', 'felice', 'peppo', 'gino', 'mario', 'antonio', 'roberto', 'massimo', 'giuseppe', 'matteo', 'silvio', 'michele', 'franco', 'guido', 'piero');

$max_num = count($amici);

for ($counter=1; $counter<=$max_num; $counter++)
{
print $amici[$counter] . "<br>";
}
?>
```

Il ciclo foreach

Riprendiamo la array amici dei precedenti esempi e costruiamo un ciclo usando, appunto, foreach:

```
<?
    $amici = array('luca', 'jacopo', 'felice', 'peppo', 'gino', 'mario',
    'antonio', 'roberto', 'massimo', 'giuseppe', 'matteo', 'silvio',
    'michele', 'franco', 'guido', 'piero');

    foreach($amici As $IndiceAmico => $NomeAmico)
    {
    print $IndiceAmico . " - " . $NomeAmico . "<br>";
    }
    ?>
```

Dopo foreach, all'interno della parentesi, abbiamo specificato innanzitutto il nome della nostra array, in secondo luogo abbiamo specificato due nuove variabili: la prima contenente l'indice dell'elemento e la seconda contenente il suo valore.

Variabili Predefinite

Esistono alcune variabili automaticamente definite dall'interprete PHP e quindi disponibili in ogni script

- $_SERVER: array associativo che contiene informazioni fornite dal web server all'interprete PHP, tra cui
  - "PHP_SELF": nome dello script PHP in esecuzione
  - "SERVER_NAME": hostname del server
  - "SERVER_ADDR": indirizzo IP del server
  - "REMOTE_ADDR": indirizzo IP del client
  - informazioni contenute nell'header della richiesta HTTP del client, ad esempio
    - "HTTP_USER_AGENT": user agent (browser) usato dal client
    - "HTTP_REFERER": URL della pagina che ha portato il client alla pagina corrente

Variabili predefinite

- $_SESSION: array associativo che contiene informazioni sulle sessioni
- $_COOKIE: array associativo contenente le variabili passate allo script tramite i cookie HTTP
- $_GET: array associativo contenente le variabili passate dal client tramite il metodo GET
- $_POST: come sopra, ma riferito al metodo POST
- E' da ricordare che queste variabili sono disponibili solo se è stato abilitato il tracking delle variabili nel file di configurazione php.ini (track_vars=On) oppure all'interno dello script con la direttiva "php_track_vars"

Istruzioni e operatori

- La sintassi di molte istruzioni PHP è uguale al linguaggio C – assegnazioni, blocchi di istruzioni, if, for, do ... while, while, switch ... case, break, continue, return
- Anche la sintassi dei commenti e di quasi tutti gli operatori (aritmetici, logici, di confronto, etc.) è uguale al C

- L'operatore . permette di concatenare due stringhe

- Costrutto list() per l'assegnazione simultanea di più variabili

```
$info = array( "alfa", "beta", "gamma" );
list($val1, $val2, $val3) = $info;
```

Passaggio di dati

Una delle principali possibilità offerte dai linguaggi di scripting lato server è quella di generare contenuti (dinamicamente) sulla base delle richieste degli utenti.
Questa interattività si realizza soprattutto attraverso le variabili **GET e POST** che consentono, appunto, agli utenti di passare al server le loro richieste o preferenze attraverso i form (i classici moduli html) o querystring.

Esistono, infatti, due diversi modi per passare dati al server: il metodo POST (generalmente usato nei form) ed il metodo GET (generalmene usato nelle querystring). Vediamoli separatamente.

Con il **metodo GET** i dati vengono passati direttamente all'interno dell'indirizzo web della pagina, il quale si presenterà accompagnato da un punto di domanda (**?**) seguito dai dati organizzati in coppie nome/valore (qualora vi siano diverse coppie queste saranno legate tra loro dal simbolo **&**).

Facciamo un esempio di utilizzo del metodo GET realizzando una semplice querystring:

http://www.sito.com/automobili.php?marca=fiat&modello=panda

Il metodo GET

PHP memorizza i dati passati attraverso la querystring all'interno dell'array **$_GET**.

Per recuperarne il valore all'interno della pagina "automobili.php":

```
<?
//Recupero il valore del parametro "tipo"
$marca_auto = $_GET['tipo'];

//Recupero il valore del parametro "modello"
$modello_auto = $_GET['modello'];

//Ora stampo semplicemente a video il risultato

echo "Hai scelto una " . $marca_auto . " modello "
. $modello_auto;
?>
```

Il metodo POST

Il **metodo POST** viene utilizzato per inviare i dati ad una applicazione PHP tramite i form (moduli html).
Prima di parlare nello specifico della variabile $_POST conviene fare un piccolo ripasso di Html e vedere (brevemente) come funzionano i moduli.

il tag <form> viene sempre accompagnato da due attributi fondamentali, che sono: "method" e "action".

Method può avere come valore sia GET (che genera una querystring) che POST.

Action ha come valore il percorso dell'applicazione a cui saranno inviati i dati. Facciamo un esempio:

```
<form method="post" action="applicazione.php">
Nome: <input type="text" name="nome">
        <input type="submit"  value="invia">
</form>
```

Il metodo POST

Diversamente dal metodo GET, il metodo POST invia i dati in maniera da non essere direttamente visibili per l'utente, attraverso la richiesta HTTP che il browser invia al server.

Tornando ll'esempio, per recuperare il valore del campo "nome" all'interno della nostra applicazione PHP useremo la variabile $_POST. Ecco come:

```
<?
//Recupero il valore del parametro "nome"
$nome_utente = $_POST['nome'];

//Ora stampo semplicemente a video il risultato
echo "Ciao " . $nome_utente;
?>
```

(Esempi: Maggiore_POST e Maggiore_GET