PHP 8

LA STORIA

PHP **nasce** nel 1994 con Rasmus Lerdorf **per tracciare il numero di visite all'interno della suo homepage personale**. Da qui il nome del PHP: Personal Home Page, che In seguito si trasformò in Hypertext Preprocessor.

AGGIORNAMENTI

Il progetto venne ripreso successivamente da Rasmus, riscritto e rilasciato integrando alcune funzionalità. A questo punto il **linguaggio** era **diventato** abbastanza **noto presso la community Open Source**, al punto che nel 1998 venne rilasciata la versione 3 che, alla fine dello stesso anno, coprirà circa il 10% dei server presenti in Rete. la versione corrente 5.x uscì nel luglio del 2004.

CARATTERISTICHE E VANTAGGI

LINGUAGGIO

Tipizzazione debole;
Non supporta i puntatori,
Utilizzato principalmente **per applicativi Web**,
Operazioni con le stringhe molto semplici da effettuare
Supporta I/O

Tipi di dati numerici supportati: 32-bit signed int, 64-bit signed int (long integer),
Tipi di dati float supportati: double precision float,
Non supporta array di dimensioni fisse,
Supporta array associativi,
consente di accedere in maniera semplicissima alle richieste HTTP di tipo GET e POST

GET E POST

GET è il metodo con cui vengono richieste la maggior parte delle informazioni ad un Web server, tali richieste vengono veicolate tramite query string, cioè la parte di un URL che contiene dei parametri da passare in input ad un'applicazione (ad esempio www.miosito.com/pagina-richiesta?id=123&page=3).

Il metodo POST, consente di inviare dati ad un server senza mostrarli in query string, è ad esempio il caso delle form.

CARATTERISTICHE E VANTAGGI

Importante è l'accesso in lettura/scrittura ai cookie del browser e il supporto alle sessioni sul server.

I cookie sono delle righe di testo usate per tenere traccia di informazioni relative ad un sito Web sul client dell'utente.

Inoltre anche se nato come linguaggio per Web, potrà essere utilizzato come linguaggio per lo scripting da riga di comando.
Uno dei punti di forza del PHP è la community molto attiva.
Sono migliaia le librerie di terze parti che ampliano le funzionalità di base e, nella maggior parte dei casi, sono tutte rilasciate con licenza Open Source.

Numerosissimi anche i framework sviluppati con questo linguaggio (Zend Framework, Symfony Framework, CakePHP, Laravel..) così come anche i CMS (WordPress, Drupal, Joomla, Magento, Prestashop..).
PHP è multipiattaforma, cioè può essere utilizzato sia in ambienti unix-based (Linux, Mac OSX) che su Windows.
La combinazione più utilizzata, però, resta quella LAMP ovvero Linux come sistema operativo, Apache come Web server, MySQL per i database e PHP

IDE (Integrated Development Environment)

Vengono utilizzati per l'autocompletamento del codice,
Gli IDE **sono software che aiutano i programmatori a sviluppare codice** attraverso l'autocompletamento delle funzioni, dei metodi delle classi, tramite la colorazione del codice o la segnalazione di errori di sintassi senza dover eseguire il codice per individuarli.
Si potrebbe preferire un IDE al semplice editor di testo.
Alcuni degli IDE più noti sono i seguenti, tutti multipiattaforma:

PHPStorm (premium);
Visual Studio Code
Eclipse (free);
Aptana Studio (free);
Zend Studio (premium).

PHPStorm è uno degli IDE più completi attualmente sul mercato.

I TAG E I COMMENTI

TAG DI APERTURA E DI CHIUSURA
Il tag di apertura del linguaggio <?php, mentre l'ultima il tag di chiusura ?>.
L'interprete PHP, infatti, **esegue solo il codice che è contenuto all'interno di questi due delimitatori.**
Questo consente di inserire del codice PHP all'interno di una normale pagina HTML così da renderla dinamica.

```
<?php
/*
In questo script vedremo quali sono gli elementi che compongono un file PHP. Questo ad esempio è un commento multilinea.
*/
// questo invece è un commento su singola riga
# questo è un altro tipo di commento
$nome = "TuoNome";
echo "Il mio nome è " .
$nome;
function stampa_nome($nome) {
    echo"<strong>Ciao " . $nome . "</strong>";
}
stampa_nome($nome);
?>
```

LE VARIABILI

Tipi Di Dato

Il PHP è un linguaggio con tipizzazione debole. Ciò significa che non vi sono regole molto rigide sulla dichiarazione del tipo di variabile. Ad esempio, linguaggi come il C, il C++, il C# o il Java prevedono che si dica sempre di che tipo sarà la variabile.

int a = 5;

Il PHP invece, non richiede alcun tipo e la variabile si dichiara in questo modo:

$a = 5;

L'unica accortezza richiesta è il simbolo del dollaro prima di ogni nome di variabile, utilizzato per segnalare che quell'elemento è una variabile.

E' possibile, inoltre, utilizzare numeri e lettere (ma non solo numeri) nel nome e il carattere underscore _ .

E' possibile quindi dichiarare variabili come:
$_var = 4;
$var1 = a;
$_1var = 33;

Non è possibile invece dichiarare variabili come:
$1234 = 33; $1var = a;
poiché iniziano con un numero.

Attenzione: il PHP è un linguaggio case sensitive (fa distinzione tra maiuscole e minuscole), perciò una variabile che inizia con la lettera maiuscola ed un'altra con la lettera minuscola sono considerate differenti.

Esecuzione di script da file

Qualsiasi file PHP può essere eseguito direttamente da Terminale attraverso il comando:
php file.php
Al file è possibile passare anche dei parametri che possono essere utilizzati dallo script.

php index.php eee ddd

Per recuperare i parametri all'interno del nostro script PHP mette a disposizione 2 variabili globali:

1. $argv: un array che contiene gli argomenti;
2. $argc: un intero che indica il numero di argomenti contenuti nell'array $argv.

```php
<?php
/**
 * esempio di riga comando
 */
print 'argc:';
print_r($argc);

print PHP_EOL;
print 'argv:';
print PHP_EOL;
print_r($argv);

/*
argc:3
argv:
Array
(
    [0] => index.php
    [1] => eee
    [2] => ddd
)
*/
```

PHP E HTML

Il codice **contenuto all'interno dei tag** delimitatori di PHP viene **interpretato** e, di conseguenza, **viene stampato soltanto l'output** che noi abbiamo stabilito in sede di stesura del sorgente.

Sono le ore 15:04 del giorno 25/06/2022.

```
<!DOCTYPE html>
<html>
<head>
  <title><?php echo "Titolo della pagina"; ?></title>
</head>
<body>
  Sono le ore <?php echo date('H:i'); ?> del giorno <?php echo date('d/m/Y'); ?>.
</body>
</html>
```

FORMATTAZIONE DELLA PAGINA HTML

IL CARATTERE \n [importante utilizzare con doppio apice "\n"]

\n = PHP_EOL (entrambi producono new line)

Solitamente nei linguaggi di programmazione il carattere \n sta ad **indica**re **un ritorno a capo.**
Anche nel PHP il comportamento è lo stesso, ma bisogna **presta**re **attenzione anche al comportamento dell'HTML**. Il codice:
<?php echo "Ciao TuoNome,\n come stai?"; ?>
verrà infatti visualizzato su una sola riga nonostante il \n.
Visualizzando il sorgente della pagina, invece, il testo risulterà disposto su due righe.
**Nell'HTML il tag corretto per forzare un ritorno a capo è
 e non è sufficiente scrivere del testo su un'altra riga.**

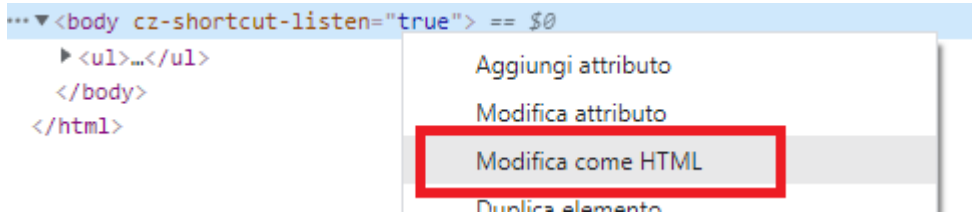
Per avere un funzionamento corretto (e cioè identico a quello atteso), abbiamo quindi bisogno di scrivere:<?php echo "Ciao Simone,
 come stai?"; ?>

FORMATTAZIONE DELLA PAGINA HTML

Se abbiamo bisogno di produrre un consistente quantitativo di markup HTML attraverso il codice PHP, può essere utile formattarlo in modo di aumentarne la leggibilità. Vediamo un esempio:
<?php echo "
\n
list item 1
\n
list item 2
\n
\n"; ?>

Il codice appena visto stamperà tramite il Web browser un codice HTML come il seguente:

list item 1
list item 2



senza \n sarebbe:

Tipi di Dato

PHP supporta I seguenti tipi:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL

Boolean – valori booleani (vero, falso)

Rappresentare i valori “vero” o “falso” all'interno di espressioni logiche.

```php
<?php
    $vero = true;
    $falso = false;
    $vero = 1 && 1;
    $falso = 1 && 0;
    $vero = 1 || 0;
    $falso = 0 || 0;
?>
```

Integer – valori interi

I valori interi sono rappresentati da cifre positive e negative comprese in un intervallo che dipende dall'architettura della piattaforma (32 o 64 bit). I valori minimo e massimo sono rappresentati attraverso la costante PHP_INT_MAX.

32 bit:

Con segno: da −2.147.483.648 a +2.147.483.647

Senza segno: da 0 a +4.294.967.295

64 bit:

Con segno: da −9.223.372.036.854.775.808 a +9.223.372.036.854.775.807
Senza segno: da 0 a +18.446.744.073.709.551.615

```
<?php
   $intero = 1;
   $intero = 1231231;
   $intero = -234224;
   $intero = 1 + 1;
   $intero = 1 - 1;
   $intero = 3 * 4;
?>
```

```
Creare un file test.php
<?php
print PHP_INT_MIN . ", " .
PHP_INT_MAX;

.. e eseguirlo ..
>php test.php
```

```
-9223372036854775808, 9223372036854775807
```

Float o double – numeri in virgola mobile

Il tipo float in PHP, conosciuto anche come double, è un numero a virgola mobile positivo o negativo. Può essere specificato tramite il punto . o, in alternativa, un esponente.

I valori con cifre decimali sono rappresentati mediante l'utilizzo del **. e non** della **,** come nel nostro sistema.

```
<?php
   $float = 10.3;
   $float = -33.45;
   $float = 6.1e6; // 6,1 * 10^6 => 6.100.000
   $float = 3E-7;  // 3 * 10^-7  =>
3/10.000.000 = 0,0000003
?>
```

Bisogna prestare attenzione a **non usare questo tipo di dato per operazioni con valori precisi**, come nel caso in cui si ha a che fare con valute.

In quel caso bisognerebbe utilizzare invece le funzioni di **BCMath**.

```php
$float1 = 123.30; // Sintassi con il punto
$float2 = 120e-3; // Sintassi con esponente

echo "$float1\n"; // 123.30
echo "$float2\n"; // 0.12
```

String – testi o stringhe di caratteri

Una stringa è un insieme di caratteri.

In PHP non è necessario definire la dimensione massima della stringa ma in ogni momento si può modificare il contenuto senza prima verificare se la variabile può contenere una determinata stringa.
Una stringa può essere **delimitata dal carattere ' (apice singolo) o dal carattere " (doppio apice).**
Il delimitatore utilizzato per determinare l'inizio di una stringa deve essere utilizzato anche per indicarne il suo termine.

```
<?php
   $stringa = "ciao come stai?";
   $stringa = 'ciao come stai?';
   $stringa = "lei mi ha chiesto 'come stai?'";
   $stringa = 'lei mi ha chiesto "come stai?"';
   //stringhe non valide perché contengono lo stesso carattere di apertura
   //all'interno della stringa
   $stringa_non_valida = "lei mi ha chiesto "come stai?"";
   $stringa_non_valida = 'lei mi ha chiesto 'come stai?';
   //utilizzare il backslash, \, per impiegare il carattere di apertura all'interno della stringa
   //tale operazione viene definita escaping delle stringhe
   $stringa_valida = "lei mi ha chiesto \"come stai?\"";
   $stringa_valida = 'lei mi ha chiesto \'come stai?\';
?>
```

Stampare stringhe con echo e print

Il costrutto **echo consente di stampare a schermo una o più stringhe**.

Essendo appunto **un costrutto**, e non una funzione, **non necessità di parentesi per essere richiamato.**

```
echo 'Questa è una stringa';
echo "Questa è una stringa";
echo "Questa è una stringa " . "concatenata";
echo "Questa è un'altra stringa ", "concatenata";
```

```
$nome = "TuoNome";
echo "Il mio nome è " . $nome;
echo "Il mio nome è ", $nome;
$array = array("TuoNome", "Luca");
echo "Il primo elemento dell'array si chiama ",
$array[0];
```

Stampare stringhe con echo e print

Possiamo anche inserire le variabili direttamente all'interno della stringa, senza concatenarle.

```
$nome = "TuoNome";
echo "Il mio nome è $nome";
```

In questo caso è necessario delimitare la stringa con il doppio apice " altrimenti non verrà interpretata la variabile.

Per accedere ad un elemento dell'array all'interno della stringa, oltre al doppio apice, abbiamo bisogno di delimitare l'accesso all'array con le parentesi graffe {}

```
$array = array("TuoNome", "Luca");
echo "Il primo elemento dell'array si chiama {$array[0]}";
```

echo in HTML e sintassi abbreviata

Un altro modo per utilizzare il costrutto echo è quello di utilizzare la sua sintassi abbreviata.

è necessario **aggiungere un = subito dopo l'apertura del tag PHP per stampare automaticamente il valore della variabile**.
Questa specifica alternativa, per funzionare correttamente, necessita che la direttiva short_open_tag sia abilitata all'interno del nostro php.ini in caso di versioni inferiori alla 5.4.0 del PHP.

Dalle versioni successive alla 5.4 , invece, <?= è sempre disponibile.

```
<?php
   //definizione variabili
   $nome = "TuoNome";
?>
... codice html ...
<p>Benvenuto <?=$nome?></p>
```

Stampare stringhe con print

Print è un altro costrutto di PHP utilizzato **per stampare** stringhe sullo standard output.

È leggermente meno performante del costrutto echo, ma il funzionamento è molto simile.
La differenza sostanziale tra i due costrutti è che quest'ultimo **ha un valore di ritorno booleano mentre il costrutto echo non ha valore di ritorno**.

Un'altra differenza non meno importante è che **print non permette di separare con la virgola diverse stringhe da stampare, ma consente unicamente di concatenarle tra di loro.**

OK:
```
echo "Questa è un'altra stringa ", "concatenata";
```

ERR:
```
print "Questa è un'altra stringa ", "concatenata";
```

Vari Esempi:

```
print "Questa è una stringa";
print "Questa è una stringa " . "concatenata";
// il seguente esempio produrrà un errore
print "Questa è un'altra stringa ", "concatenata";
$nome = "TuoNome";
print "Il mio nome è $nome";
$array = array(" TuoNome", "Luca");
print "Il primo elemento dell'array si chiama {$array[0]}";
```

Backslash

Il backslash \ deve essere utilizzato come carattere di escape quando si vuole includere nella stringa lo stesso tipo di carattere che la delimita;

se mettiamo un backslash davanti ad un apice doppio in una stringa delimitata da apici singoli (o viceversa), anche il backslash entrerà a far parte della stringa stessa.

Il backslash viene usato anche come “escape di sé stesso” nei casi in cui si desideri includerlo esplicitamente in una stringa.

```
<?php
   echo "Questo: \"\\\" è un backslash"; //
stampa: Questo: "\" è un backslash
   echo 'Questo: \'\\\' è un backslash'; //
stampa: Questo: '\' è un backslash
   echo "Questo: '\' è un backslash";    //
stampa: Questo: '\' è un backslash
   echo "Questo: '\\' è un backslash";   //
stampa: Questo: '\' è un backslash
?>
```

PER DELIMITARE UNA STRINGA

Per delimitare una stringa, sarà possibile **inserire delle variabili all'interno della stessa**.
In questo caso la variabile verrà automaticamente interpretate dal PHP e convertita nel valore assegnato ad essa (“esplosione della variabile”):
Utilizzare il " doppio apice

```php
<?php
    $nome = "TuoNome";
    $stringa = "ciao $nome, come stai?";
//ciao TuoNome, come stai?
    $stringa = 'ciao $nome, come stai?';
//ciao $nome, come stai?
//NON Funziona
?>
```

NOTAZIONE HEREDOC

La notazione heredoc, **utilizzata per specificare stringhe molto lunghe**, consente di **delimitare una stringa con i caratteri seguiti da un identificatore**;
a questo scopo in genere si usa **EOD**, ma è solo una convenzione, è possibile utilizzare qualsiasi stringa composta di caratteri alfanumerici e underscore, di cui il primo carattere deve essere non numerico (la stessa regola dei nomi di variabile).

Tutto ciò che segue questo delimitatore viene considerato parte della stringa, fino a quando non viene ripetuto l'identificatore seguito da un punto e virgola.

Attenzione: **l'identificatore di chiusura deve occupare una riga a sé** stante, deve iniziare a colonna 1 e non deve contenere nessun altro carattere (nemmeno spazi vuoti) dopo il punto e virgola.

La sintassi heredoc risolve i nomi di variabile così come le virgolette. Rispetto a queste ultime, con tale sintassi abbiamo il vantaggio di poter includere delle virgolette nella stringa senza farne l'escape:

```
<?php
$nome = "TuoNome";
$stringa = <<<EOD
Il mio nome è $nome
EOD;
echo $stringa;

$contenuto = <<<DATA;
Tutto questo contenuto
<div>verrà inserito nella
variabile </div>
DATA;

DATA; dalla 7.3 può non
essere ad inizio riga
```

ARRAY

Un array (o vettore) può essere considerato **come una matrice matematica.**

In PHP esso rappresenta una variabile complessa che restituisce una mappa ordinata basata sull'associazione di coppie chiave => valore.

La chiave di un array può essere un numero o di tipo stringa, mentre il valore può contenere ogni tipo di dato, compreso un nuovo array.

```
<?php
   $array = array(); //array vuoto
   $array = [];      //array vuoto nella nuova
sintassi
   $array = array('a', 'b', 'c');
   $array = array(1, 2, 3);
?>
```

Array

Possiamo descrivere un array come una matrice matematica.
In termini più semplici può essere considerato **una collezione di elementi ognuno identificato da un indice**; **gli indici** di un array **possono essere numerici o stringhe**.
Gli elementi dell'array possono contenere al proprio interno qualsiasi tipo di dato, compresi oggetti o altri array.
Un array può essere definito in due modi equivalenti:

```
$array = array();
$array = [];
```

Esempio:

```
$frutti = array('banana', 'pesca', 'lampone');
$frutti = ['banana', 'pesca', 'lampone'];
```

Array e la funzione print_r()

In ogni momento **possiamo visualizzare il contenuto di un array con la funzione print_r(),** utile in caso di debug.
Un array è quindi un contenitore di elementi contraddistinti da un indice.

Se l'indice non viene esplicitato di default è di tipo numerico 0-based.

0-based sta ad indicare che l'indice dell'array inizierà da 0 anziché da 1.

Come abbiamo potuto vedere nell'output dell'esempio precedente, il primo elemento dell'array (banana) possedeva infatti l'indice con valore 0.

```
$frutti = ['banana', 'pesca', 'lampone'];
print_r($frutti);
```

L'esempio restituirà un output come il seguente:

```
Array
(
[0] => banana
[1] => pesca
[2] => lampone
)
```

Array ad indici

sono array con indice numerico
è possibile accedere ai singoli elementi attraverso il proprio indice.

L'esempio stamperà Il mio frutto preferito è il lampone perché ho deciso di stampare il terzo (indice 2 partendo da 0) elemento dell'array.

```
$frutti = ['banana', 'pesca', 'lampone'];
echo "Il mio frutto preferito è il " . $frutti[2];
```

Array associativi

Un altro modo molto di utilizzare gli array è quello di **sfruttare gli indici come stringhe** anziché come valore numerico.
Nell'array appena definito **ogni elemento è caratterizzato da una coppia chiave -> valore** in cui la chiave (o indice) è il nome di una persona e il valore è il suo anno di nascita.

```
$annoDiNascita = [
'TuoNome' => '1986',
'Gabriele' => '1991',
'Giuseppe' => '1992',
'Renato' => '1988'
];
```

Per stampare la data di nascita di uno degli utenti:

```
echo "L'anno di nascita di Gabriele è " . $annoDiNascita['Gabriele'];
```

Usando la funzione print_r() sull'array definito in precedenza, eseguendo quindi il comando print_r($annoDiNascita); otterremo il seguente output:

```
Array
(
[TuoNome] => 1986
[Gabriele] => 1991
[Giuseppe] => 1992
[Renato] => 1988
)
```

LE FUNZIONI

LE FUNZIONI

Una funzione viene dichiarata con la keyword function, seguita dal nome della funzione e, tra parentesi tonde, dal nome dei parametri necessari alla funzione. Il codice della funzione è delimitato da due parentesi graffe { ... }

```
<?php
/*In questo script vedremo quali sono gli elementi che compongono un file PHP. Questo ad esempio è un commento multilinea.
*/
// questo invece è un commento su singola riga
# questo è un altro tipo di commento
$nome = "TuoNome";
echo "Il mio nome è " . $nome;
function stampa_nome($nome) {
    echo "<strong>Ciao " . $nome ."</strong>";
}
stampa_nome($nome);
?>
```

PSR-12

A function declaration looks like the following. Note the placement of parentheses, commas, spaces, and braces:

```
function fooBarBaz($arg1, &$arg2, $arg3 = [])
// NO BLANK LINE
{
// NO BLANK LINE
    // function body
// NO BLANK LINE
}
```

Quindi:

```
function fooBarBaz($arg1, &$arg2, $arg3 = [])
{
    // function body
}
```

LE FUNZIONI

Nel caso in cui, invece, **dovesse ritornare qualcosa, si utilizza la keyword** return.

Ad esempio, supponiamo di voler scrivere **una funzione che sommi due parametri** e restituisca il risultato, il codice PHP necessario sarà:

```
function somma($a, $b) {
  $somma = $a + $b;
  return $somma;
}
$somma = somma(3,5); // $somma sarà uguale ad 8
```

Funzioni in PHP

I parametri passati in ingresso ad una funzione possono anche essere opzionali se definiamo un valore di default.
Supponiamo che il secondo parametro della funziona sia opzionale e, se non definito, sia di default 10, potremmo a quel punto richiamare la nostra funzione passando solo il primo parametro.

Automaticamente PHP sommerà al nostro primo parametro il valore 10

```
function somma($a, $b = 10) {
    return $a + $b;
}
```

Il codice della funzione è lo stesso, l'unica cosa che cambia è il valore di inizializzazione della variabile $b.
Eseguendo il codice:
echo "La somma di 5 e 10 è: " . somma(5);
L'output sarà:
La somma di 5 e 10 è: 15

Le Costanti in PHP define('NOME', "valore")

una porzione di memoria destinata a **contenere un dato** caratterizzato dal fatto di essere **immutabile durante l'esecuzione di uno script**.

Sulla base di quanto appena detto le costanti possono essere quindi considerate come **l'esatto opposto di una variabile**, inoltre, sempre a differenza delle variabili**, le costanti divengono automaticamente globali per l'intero script nel quale vengono definite**.
Una costante **può essere definita in PHP attraverso l'impiego del costrutto define che accetta due parametri in ingresso**, questi ultimi sono: **il nome della costante e il valore associato ad essa** in fase di digitazione del codice.

Le costanti in PHP vengono **dichiarate attraverso dei nomi che iniziano con una lettera o con l'underscore**, per esse **non è** quindi **previsto l'impiego del carattere iniziale $** come avviene invece per le variabili.

PSR-1
Per convenzione i nomi delle costanti vengono scritti interamente in maiuscolo.
Il valore contenuto da una costante può essere **soltanto di tipo scalare o null**.
I **tipi di dato scalare sono**: interi, float, stringhe e booleani

```
define('COSTANTE', 'stringa');
define('POST_PER_PAGINA', 10);
define('TEMPLATE_EMAIL', 'template.html');
```

```
define('NOME',123);
print NOME;
//stampa 123
```

Le espressioni in PHP

Il valore di una qualsiasi espressione può essere assegnato ad una variabile.

Nel caso proposto di **seguito il valore della variabile è dato dal risultato dell'espressione che si trova a destra dell'operatore di assegnazione (=)**

Questa sintassi consente di avere più espressioni all'interno di una singola istruzione.

```
$a = 3 + 3; //il valore di $a è il risultato dell'espressione: $a = 6
```

OPERATORI ARITMETICI DI PHP

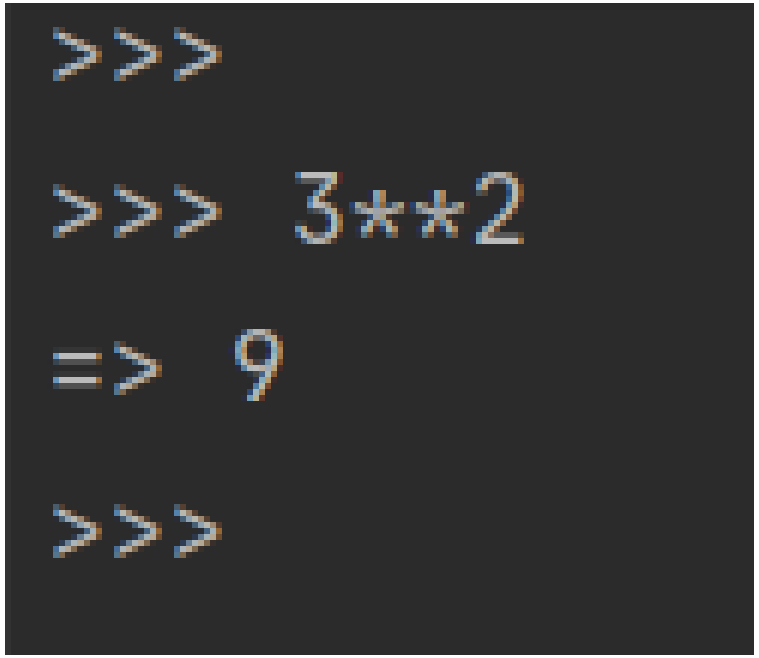
Altri operatori messi a disposizione da PHP per effettuare operazioni aritmetiche.

Quelli più semplici:

```
$a = 1 + 1; // operatore di somma
$b = 1 - 1; // operatore di sottrazione
$c = 1 * 1; // operatore di moltiplicazione
$d = 2 / 1; // operatore di divisione
```

Operatore Elevazione a Potenza **

**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power



Le espressioni in PHP

Ci sono **operatori che possono avere priorità maggiore e che quindi vengono eseguiti prima di altri.**

L'operatore di incremento ++ aumenta unitariamente il valore della variabile a cui è applicato.

Al termine della valutazione, quindi, verrà stampato a schermo il valore 2.

L'operatore può essere applicato prima (pre-incremento) o dopo (post-incremento) rispetto alla variabile di riferimento.

```
$i = 1;
echo ++$i;
//stampa 2;
```

A seconda della posizione cambia la priorità e, di conseguenza, il risultato dell'operazione.

Nel caso in cui l'operatore venga posizionato dopo la variabile, il suo incremento verrà posticipato al termine della valutazione dell'espressione.

In quest'ultimo esempio verrà stampato a schermo il valore 1

```
$i = 1;
echo $i++;
//stampa 1
```

Operatori Aritmetici

Gli operatori aritmetici possono essere utilizzati non solo tra valori scalari ma anche con variabili e funzioni.
il risultato di un'espressione viene assegnato come risultato alla variabile stessa.
La variabile $b, quindi, viene modificata sommando il suo valore iniziale al valore di $a

```
$a = 1 + 1;
$b = 4 - 2;
$b = $b + $a; // 2 + 2 = 4
```

Operatori di assegnazione combinati

La stessa espressione appena vista può essere riscritta utilizzando gli operatori di assegnazione combinati:

Tali operatori possono essere utilizzati con le operazioni aritmetiche + - * / %

```
$b += $a; // equivale a $b = $b + $a
```

```
$b += $a; // $b = $b + $a
$b -= $a; // $b = $b - $a
$b *= $a; // $b = $b * $a
$b /= $a; // $b = $b / $a
$b %= $a; // $b = $b % $a
```

```
$str .= "aggiunto alla stringa";
```

Operatori di incremento e decremento

La differenza della posizione dell'operatore determina la priorità con cui viene eseguito l'incremento o il decremento.

Se si trova a sinistra viene eseguita prima di valutare il resto dell'espressione;
se si trova a destra viene eseguita dopo aver valutato l'espressione.

```
$i = 0;
++$i; // incrementa di 1 la variabile $i
$i++; // incrementa di 1 la variabile $i
--$i; // decrementa di 1 la variabile $i
$i--; // decrementa di 1 la variabile $i
```

Operatori di confronto

Gli operatori di confronto consentono di determinare il valore di due espressioni in base al confronto tra i loro valori.

Operator	Name
==	Equal
===	Identical
!=	Not equal
<>	Not equal
!==	Not identical
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<=>	Spaceship

If, else, istruzioni condizionali in PHP

Nel corso delle lezioni precedenti **è stato possibile analizzare alcuni frammenti di codice in cui venivano eseguite operazioni in base a determinate condizioni.**

Questo è possibile grazie alle istruzioni condizionali che ci consentono di avere comportamenti differenti all'interno del nostro codice in base a specifiche condizioni.

if

Dopo la keyword if, deve essere indicata fra parentesi un'espressione da valutare (condizione). Questa espressione sarà valutata in senso booleano, cioè il suo valore sarà considerato vero o falso.
Dunque, **se la condizione è vera, l'istruzione successiva verrà eseguita**, altrimenti sarà ignorata.

Se l'istruzione da eseguire nel caso in cui la condizione sia vera **è una sola, le parentesi graffe sono opzionali**

```
if( <condizione> ) <istruzione>
```

```
if ($nome == 'Marco') {
  print "Marco";
}
```

```
if ($nome == 'Marco')
  print "Marco";
```

```
if ($i == 1) :
    print "si 1";
else :
    print "no 1";
endif;
```

Condizioni booleane

Se si valuta una stringa piena "ciao" **è considerato vero (TRUE).**

In sostanza, PHP considera come **falsi**:

- il valore numerico 0, nonché una **stringa che contiene "0"**;
- una **stringa vuota**;
- un **array con zero elementi**;
- **un valore NULL**, cioè una variabile che non è stata definita o che è stata eliminata con unset(), oppure a cui è stato assegnato il valore NULL esplicitamente.

Qualsiasi altro valore è per PHP un valore **vero**.
Quindi qualsiasi numero, intero o decimale purché diverso da "0", qualsiasi stringa non vuota, se usati come espressione condizionale saranno considerati veri.

```
if ("ciao") {
    print "è true";
}
```

If e Operatori di uguaglianza (==) e assegnamento (=)

$a = 7;

```php
<?php
if ($a = 4) echo "$a è uguale a 4!";
// 4 è uguale a 4!
```

Attenzione = è una assegnazione e non un confronto

viene stampato solamente perché $a è un numero != 0 e quindi vero

else

Per eseguire istruzioni se la condizione è falsa abbiamo bisogno del costrutto else
La parola chiave **else,** che **significa "altrimenti", deve essere posizionata subito dopo la parentesi graffa di chiusura** del codice previsto per il caso "vero" (o dopo l'unica istruzione prevista, se non abbiamo usato le graffe).

Anche per else valgono le stesse regole: niente punto e virgola, parentesi graffe obbligatorie se dobbiamo esprimere più di un'istruzione, altrimenti facoltative.

Ovviamente **il blocco di codice specificato per else verrà ignorato quando la condizione è vera, mentre verrà eseguito se la condizione è falsa**.

```
if (<condizione>) {
   <codice>
} else {
   <codice>
}
```

if – else - Istruzioni nidificate

Le istruzioni if possono essere nidificate una dentro l'altra, consentendoci così di creare codice di una certa complessità:

```
if ($nome == 'Luca') {
    if ($cognome == 'Rossi') {
        echo "Luca Rossi è di nuovo fra noi";
    } else {
        echo "Abbiamo un nuovo Luca!";
    }
} else {
    echo "ciao $nome!";
}
```

I Cicli PHP, for, while e do

I cicli sono un altro degli elementi fondamentali di qualsiasi linguaggio di programmazione in quanto **permettono di eseguire determinate operazioni in maniera ripetitiva.**
Si tratta di una necessità che si presenta molto frequente.

FOR

Si ipotizzi di voler mostrare i multipli da 1 a 10 di un numero intero, ad esempio “5”.

La prima soluzione disponibile è basata sull'impiego del ciclo for.

Questo costrutto, simile a quello usato in altri linguaggi, utilizza la parola chiave **for, seguita,** fra parentesi**, dalle istruzioni per definire il ciclo**; **successivamente** si racchiudono **fra parentesi graffe tutte le istruzioni che devono essere eseguite ripetutamente**.

```
for ($mul = 1; $mul <= 10; ++$mul) {
$ris = 5 * $mul;
echo "5 * $mul = $ris <br/>";
}
```

For

Le tre istruzioni inserite *fra le parentesi tonde e separate da punto e virgola* **vengono trattate in questo modo:**

- **la prima viene eseguita una sola volta, all'inizio del ciclo;**
- **la terza viene eseguita alla fine di ogni iterazione del ciclo;**
- **la seconda deve essere una condizione, e viene valutata prima di ogni iterazione del ciclo;**

Quando la condizione risulta falsa, l'esecuzione del ciclo viene interrotta e il controllo passa alle istruzioni presenti dopo le parentesi graffe.

Ovviamente è possibile che tale condizione risulti falsa fin dal primo test: in questo caso, le istruzioni contenute fra le parentesi graffe non saranno eseguite nemmeno una volta.

Il formato standard è quindi quello che utilizza le parentesi tonde per definire un "contatore": con la prima istruzione lo si inizializza, con la seconda lo si confronta con un valore limite oltre il quale il ciclo deve terminare, con la terza lo si incrementa dopo ogni esecuzione.

```
for ($mul = 1; $mul <= 10; ++$mul) {
    $ris = 5 * $mul;
    echo "5 * $mul = $ris <br/>";
}
```

PSR-12

A for statement looks like the following. Note the placement of parentheses, spaces, and braces.

```
<?php

for ($i = 0; $i < 10; $i++) {
    // for body
}
```

Le richieste HTTP (GET E POST)

La specifica HTTP definisce 9 tipi di metodi alcuni dei quali non sono però usati o supportati da PHP; i più diffusi restano sicuramente GET e POST.

GET è il metodo con cui vengono richieste la maggior parte delle informazioni ad un Web server, tali richieste vengono veicolate tramite query string, cioè la parte di un URL che contiene dei parametri da passare in input ad un'applicazione.

Il metodo POST, invece, consente di inviare dati ad un server senza mostrarli in query string, è ad esempio il caso dei form

Le richieste HTTP (GET E POST)

Iniziamo con il metodo GET. Sicuramente è il più semplice e il più immediato.

È consigliato soprattutto in quelle richieste in cui è utile salvare nell'URL i parametri richiesti.

Per poter accedere ai parametri in GET di una richiesta HTTP proveniente da un form di ricerca avremo bisogno di due file:
form.html e
search.php.

Analizziamo innanzitutto il codice del file form.html che conterrà il form:

```
<!DOCTYPE html>
<html>
<head>
   <title></title>
</head>
<body>
   <form action="search.php">
     <input type="text" name="author" placeholder="Inserisci autore" />
     <input type="submit" value="Cerca" />
   </form>
</body>
</html>
```

Ricevere un parametro $_GET

```
$author = $_GET['author']; $author = filter_var($author,
FILTER_SANITIZE_STRING);
```

FILTER_SANITIZE_STRING DEPRECATO DALLA VERSIONE 8 USARE htmlspecialchars

```
$new = htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);
echo $new; // &lt;a href=&#039;test&#039;&gt;Test&lt;/a&gt;
```

POST

Il metodo POST si differenza da GET in quanto i parametri della richiesta non vengono passati in query string e quindi non possono essere tracciati nemmeno negli access log dei web server.

Caso d'uso comune di una richiesta in POST è un form che invia dati personali, come in una registrazione.
Vediamo quindi come accedere ai parametri POST con un esempio di registrazione tramite username e password.
Anche in questo caso abbiamo bisogno di due file: form.html e register.php.

Il codice è simile all'esempio precedente. Le differenze sostanziali sono la presenza di due campi username e password e, soprattutto, l'aggiunta dell'attributo method nel tag form.
Quando abbiamo bisogno di effettuare una richiesta POST è necessario specificare il metodo nel form.

Il file contenente il form:

```
<!DOCTYPE html>
<html>
<head>
   <title></title>
</head>
<body>
   <form action="register.php" method="post">
      <input type="text" name="username"
placeholder="Inserisci lo username" /><br>
      <input type="password" name="password"
placeholder="Inserisci la password" /><br>
      <input type="submit" value="Registrati" />
   </form>
</body>
</html>
```

Recevere un parametro $_POST

```
$username = $_POST['username'];
$password = $_POST['password'];
$username = filter_var($username, FILTER_SANITIZE_STRING); $password = filter_var($password, FILTER_SANITIZE_STRING); if (!$username || !$password) { $error = 'Username e password sono obbligatori'; }

if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
  }
```

post e validate

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
  $emailErr = "Invalid email format";
}
```

HTML CSS E PHP e Tema Child in WP

WordPress chiamato "mio-tema«

1. Vai alla directory dei temi di WordPress: /wp-content/themes/
2. Crea una nuova cartella per il tema child. Per esempio, puoi chiamarla mio-tema-child.
3. ll'interno della cartella del tema child (mio-tema-child), crea un file chiamato style.css
4. Nella cartella del tema child, crea un file chiamato functions.php

Tema child pronto

style.css

```
/*
Theme Name: Mio Tema Child
Template: mio-tema
Author: Il tuo nome
Description: Un tema child per il tema Mio Tema
Version: 1.0
*/
```

functions.php

```
<?php
function mio_tema_child_enqueue_styles() {
    // Carica lo stile del tema genitore
    wp_enqueue_style('mio-tema-parent-style',
get_template_directory_uri() . '/style.css');

    // Carica lo stile del tema child (dopo il tema genitore)
    wp_enqueue_style('mio-tema-child-style',
get_stylesheet_directory_uri() . '/style.css', array('mio-tema-
parent-style'));
}
add_action('wp_enqueue_scripts',
'mio_tema_child_enqueue_styles');
```