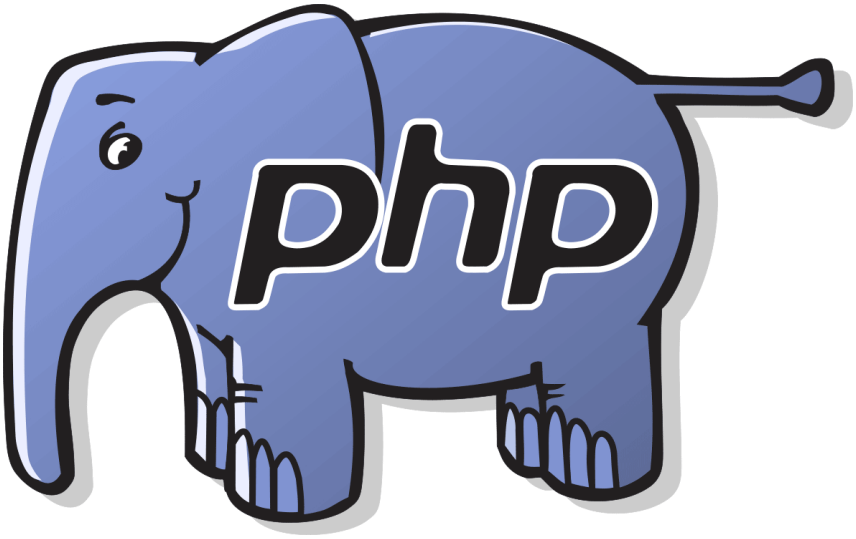

Guida PHP

Prima parte

A cura di

Aldo Guastafierro

Sommario

Premessa

Questo E-book contiene la parte teorica e pratica del linguaggio di programmazione PHP
Molti dei contenuti di questo documento si possono trovare sul sito web che ho predisposto per i miei alunni all'indirizzo www.aldoguastafierro.it/corso_web.
Gli esempi e gli esercizi proposti sono invece collocati a questo indirizzo:
http://www.aldoguastafierro.altervista.org/esercizi/php/esercizi_php/index.html
Questo E-book nasce per offrire un prodotto editoriale totalmente gratuito agli studenti e integrare i contenuti relativi ai linguaggi del web presenti in forma sintetica nei libri di testo.
Molte delle informazioni presenti in questo documento sono liberamente tratte da documenti pubblici e/o pubblicati sul Web.

Qui un elenco dei riferimenti web a cui va il mio personale ringraziamento:

Il blog didattico del prof. Paolo Latella http://paololatella.blogspot.it
http://www.html.it
http://www.mrwebmaster.it
http://www.php.net/manual/en/ref.strings.php
http://www.web-link.it/php/index7.php
http://www.miniscript.it
https://www.w3schools.com/

PHP 5 Introduzione

Gli script PHP vengono eseguiti sul server web.

Che cosa dovreste già sapere

Prima di continuare è necessario avere una conoscenza di base dei seguenti elementi:

- HTML
- CSS
- JavaScript

Che cosa è 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:
Legge ed individua la pagina sul server.
Esegue le istruzioni PHP contenute all'interno della pagina ed esegue tutti i comandi PHP.
Rimanda la pagina risultante al Browser

Che cosa è un file PHP?

- File PHP possono contenere testo, HTML, CSS, JavaScript, PHP e il codice
- Codice PHP vengono eseguiti sul server, e il risultato viene restituito al browser come semplice HTML
- File PHP hanno estensione ".php"

Cosa può fare PHP?

- PHP può generare contenuti attraverso pagina dinamica
- PHP può creare, aprire, leggere, scrivere, cancellare e chiudere i file sul server
- PHP in grado di raccogliere i dati dei moduli
- PHP può inviare e ricevere i cookie
- PHP può aggiungere, eliminare, modificare i dati nel database
- PHP può essere usato per controllare l'accesso utente
- PHP può crittografare i dati

PHP non si è limitato a visualizzare pagine HTM. È possibile trasmettere le immagini, file PDF e perfino filmati Flash. È anche possibile visualizzare qualsiasi testo, come XHTML e XML.

Perché PHP?

- PHP viene eseguito su diverse piattaforme (Windows, Linux, Unix, Mac OS X, etc.)
- PHP è compatibile con quasi tutti i server usati oggi (Apache, IIS, ecc)
- PHP supporta una vasta gamma di database
- PHP è libero. Scaricalo dalla risorsa ufficiale di PHP: www.php.net
- PHP è facile da imparare e funziona in modo efficiente sul lato server

PHP 5 Installazione

Di cosa ho bisogno?

Per iniziare a utilizzare PHP, è possibile:

- Trovare un host web con supporto PHP e MySQL
- Installare un server web sul proprio PC, e quindi installare PHP e MySQL

Utilizzare un host web con supporto PHP

Se il server ha attivato il supporto per PHP non è necessario fare nulla.

Basta creare alcuni file .php, inserirli nella vostra directory web, e il server automaticamente li analizzerà per voi.

Non è necessario compilare nulla o installare strumenti aggiuntivi.

Perché PHP è libero, la maggior parte dei web host offrono un supporto PHP.

Impostazione PHP sul proprio PC

Tuttavia, se il server non supporta PHP, è necessario:

- Installare un server web
- Installare PHP
- Installare un database, come MySQL

Il sito ufficiale di PHP (PHP.net) ha istruzioni di installazione per PHP: http://php.net/manual/en/install.php

PHP 5 Sintassi

Uno script PHP viene eseguito sul server, e il risultato viene inviato al browser in formato HTML.

PHP Sintassi base

Uno script PHP può essere posizionato in qualsiasi punto del documento.

Uno script PHP inizia con **<? Php?** E termina con **?>**:

```
<?php
// PHP code goes here
?>
```

L'estensione di file predefinito per i file PHP è ".php".

Un file PHP normalmente contiene i tag HTML, e codice di scripting PHP.

Qui di seguito, abbiamo un esempio di un semplice file PHP, con uno script PHP che utilizza una funzione PHP "eco" per visualizzare il testo "Ciao Mondo!" su una pagina web:

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>
```

Commenti in PHP

Un commento in codice PHP è una linea che non viene letta/eseguita come parte del programma. Il suo unico scopo è quello di essere letto da qualcuno che sta guardando il codice.

I commenti possono essere utilizzati per:

- Lasciare agli altri la possibilità di capire cosa si sta facendo
- Ricordare a voi stessi quello che si è fatto.

```
<!DOCTYPE html>
<html>
<body>

<?php
// This is a single-line comment

# This is also a single-line comment

/*
This is a multiple-lines comment block
that spans over multiple
lines
*/

// You can also use comments to leave out parts of a code line
$x = 5 /* + 15 */ + 5;
echo $x;
?>

</body>
</html>
```

Esegui l'esempio

PHP case-sensitive

In PHP, tutte le parole chiave (per esempio se, altro, mentre, eco, ecc), classi, funzioni e funzioni definite dall'utente non sono case-sensitive.

Nell'esempio qui sotto, tutti e tre le istruzioni echo indicati sono legali:

Esempio

```
<!DOCTYPE html>
<html>
<body>

<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>

</body>
</html>
```

Esegui esempio

Nell'esempio che segue, solo la prima istruzione visualizzerà il valore della variabile $color (questo è perché $color, $COLOR, e $coLOR sono trattati come tre diverse variabili):

Esempio

```
<!DOCTYPE html>
<html>
<body>

<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>

</body>
</html>
```

Esegui esempio

PHP 5 Variabili

Le variabili sono "contenitori" per la memorizzazione di informazioni.

Creazione (dichiarante) Variabili PHP

In PHP, una variabile inizia con il segno $, seguito dal nome della variabile:

Esempio

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

Dopo l'esecuzione delle dichiarazioni di cui sopra, la variabile **$txt** conterrà il valore **Ciao mondo!** La variabile **$x** conterrà il valore **5**, e la variabile **$y** conterrà il valore **10.5** .

Nota: Quando si assegna un valore di testo a una variabile, mettere le virgolette intorno al valore.

Nota: A differenza di altri linguaggi di programmazione, PHP non ha alcun comando per dichiarare una variabile. Il tipo di dato viene creato nel momento in cui si assegna un valore ad essa. Si dice che PHP è un linguaggio di scripting poco tipizzato

Pensate a variabili come contenitori per l'archiviazione dei dati.

Variabili PHP

Una variabile può avere un nome breve (come x e y) o un nome più descrittivo (età, name, total volume).

Regole per variabili PHP:

- Una variabile inizia con il simbolo $, seguito dal nome della variabile
- Un nome di variabile deve iniziare con una lettera o il carattere di sottolineatura
- Un nome di variabile non può iniziare con un numero
- Un nome di variabile può contenere solo caratteri e sottolineature alfanumerici (AZ, 0-9 e _)
- I nomi delle variabili sono case-sensitive ($età e $ETA' sono due variabili differenti)

Ricorda che in PHP nomi delle variabili sono case-sensitive!

Variabili di output

Il costrutto **eco** di PHP viene spesso utilizzato per visualizzare i dati in uscita sullo schermo.

L'esempio seguente mostra come testo e una variabile vengono visualizzaate:

Esempio

```
<?php
$txt="W3Schools.com";
echo "I love $txt!";
?>
```

L'esempio seguente produce lo stesso output nell'esempio precedente:

Esempio

```
<?php
$txt = "W3Schools.com";
echo "I love " .$txt . "!";
?>
```

Il seguente esempio emetterà la somma di due variabili:

Esempio

```
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

PHP è un linguaggio debolmente tipizzato

Nell'esempio di cui sopra, si noti che non abbiamo dovuto dire a PHP che tipo di dati la variabile è.

PHP converte automaticamente la variabile per il tipo di dati corretto, a seconda del suo valore.

In altri linguaggi come C, C ++ e Java, il programmatore deve dichiarare il nome e il tipo della variabile prima di utilizzarla.

Variabili PHP Scope

In PHP, le variabili possono essere dichiarate in qualsiasi parte del programma.

L'ambito di una variabile è la parte dello script dove la variabile può fare riferimento / utilizzato.

PHP ha tre diversi modalità di gestire le variabili:

- Locale
- globale
- statico

Ambito globale e locale

Una variabile dichiarata **al di fuori** di una funzione ha una portata globale e si può accedere solo al di fuori di una funzione:

Esempio

```
<?php
$x = 5; // global scope

function myTest() {
    // using x inside this function will generate an error
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

Una variabile dichiarata **all'interno di** una funzione ha un ambito locale e si può accedere solo all'interno di tale funzione:

Esempio

```
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
?>
```

Si possono avere variabili locali con lo stesso nome in diverse funzioni, perché le variabili locali sono riconosciuti solo dalla funzione in cui sono dichiarate.

PHP e parola chiave global

La parola chiave **global** viene utilizzato per accedere a una variabile globale dall'interno di una funzione.

Per fare questo, utilizzare la parola chiave global prima delle variabili (all'interno della funzione):

Esempio

```php
<?php
$x = 5;
$y = 10;

function myTest() {
    global $x, $y;
    $y = $x + $y;
}

myTest();
echo $y; // outputs 15
?>
```

Esegui esempio

PHP può memorizza anche tutte le variabili globali in un array chiamato $GLOBALS [*index*]. L' *indice* contiene il nome della variabile. Questo array è accessibile anche dall'interno funzioni e può essere utilizzato per aggiornare direttamente variabili globali.

L'esempio di cui sopra può essere riscritto in questo modo:

Esempio

```php
<?php
$x = 5;
$y = 10;

function myTest() {
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
?>
```

Esegui esempio

PHP la parola chiave static

Normalmente, quando si esegue una funzione, tutte le sue variabili sono soppresse. Tuttavia, a volte vogliamo che una variabile locale non deve essere cancellata. Ne abbiamo bisogno per un ulteriore lavoro.

Per fare questo, si utilizza la parola chiave **static** prima che si dichiara la variabile:

Esempio

```
<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}

myTest();
myTest();
myTest();
?>
```

Poi, ogni volta che la funzione viene chiamata, quella variabile avrà ancora le informazioni che conteneva dall'ultima volta che la funzione è stato chiamato.

Nota: La variabile è ancora locale alla funzione.

PHP 5 "eco" e funzione print

In PHP ci sono due modi per ottenere l'output: **eco e print**.

In questo tutorial usiamo eco (e stampare) in quasi ogni esempio. Quindi, questo capitolo contiene un pò più informazioni su queste due istruzioni di output.

Php echo e print

eco e print sono due istruzioni che fanno più o meno la stessa cosa. Entrambe sono utilizzati per i visualizzare dati e informazioni in uscita.

Le differenze sono piccole: **echo** è un costrutto e quindi non si comporta come una funzione e non ha alcun valore di ritorno, mentre **print** ha un valore di ritorno di 1 in modo che possa essere usato nelle espressioni per controllare la stampa. Echo può prendere parametri multipli (anche se tale uso è raro) mentre print può avere al proprio interno un argomento. Echo è marginalmente più veloce di print.

L'istruzione echo PHP

La dichiarazione echo può essere usata con o senza parentesi: echo().

Visualizza testo

L'esempio seguente mostra output di testo con il comando echo (si noti che il testo può contenere markup HTML):

Esempio

```
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?>
```

Visualizzazione di variabili

L'esempio seguente mostra come output testo e variabili con l'istruzione echo:

Esempio

```php
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;

echo "<h2>$txt1</h2>";
echo "Study PHP at $txt2<br>";
echo $x + $y;
?>
```

L'istruzione print PHP

La funzione print può essere utilizzata con o senza parentesi: print ().

Visualizza testo

L'esempio seguente mostra il testo di output con il comando print (si noti che il testo può contenere markup HTML):

Esempio

```php
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>
```

PHP 5 tipi di dati

Tipi di Dati in PHP

Le variabili possono memorizzare dati di tipo diverso, e diversi tipi di dati possono fare cose diverse.

PHP supporta i seguenti tipi di dati:

- String
- Integer
- Float (numeri in virgola mobile)
- Boolean
- Array
- Object
- NULL
- Resource

PHP String

Una stringa è una sequenza di caratteri, come "Ciao mondo!".

Una stringa può essere qualsiasi testo all'interno virgolette. È possibile utilizzare virgolette singole o doppie:

Esempio

```php
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

PHP Integer

Un tipo di dati integer è un numero non decimale tra -2.147.483.648 e 2.147.483.647.

Regole per gli interi:

- Un numero intero deve avere almeno una cifra
- Un numero intero non deve avere un punto decimale
- Un numero intero può essere positivo o negativo
- Interi possono essere specificati in tre formati: decimale (10-based), esadecimale (16-based - prefisso 0x) o ottale (base 8 - prefisso 0)

Nel seguente esempio $x è un numero intero. La funzione PHP var_dump () restituisce il tipo e il valore dei dati:

Esempio

```
<?php
$x = 5985;
var_dump($x);
?>
```

PHP Float

Un flot (numero a virgola mobile) è un numero con punto decimale o un numero in forma esponenziale.

Nel seguente esempio $x è un float. La funzione PHP var_dump () restituisce il tipo e il valore dei dati:

Esempio

```
<?php
$x = 10.365;
var_dump($x);
?>
```

PHP booleano

Un valore booleano rappresenta due possibili stati: vero o falso.

```
$x = true;
$y = false;
```

Booleani sono spesso usati nei test condizionale.

PHP Array

Un array memorizza valori multipli in una singola variabile.

Nel seguente esempio $cars è un array. La funzione PHP var_dump () restituisce il tipo e il valore dei dati:

Esempio

```
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>
```

PHP Object

Un oggetto è un tipo di dati che memorizza dati e informazioni sul come elaborare tali dati.

In PHP, un oggetto deve essere esplicitamente dichiarato.

In primo luogo dobbiamo dichiarare una classe di oggetti. Per questo, usiamo la parola chiave class. Una classe è una struttura che può contenere proprietà e metodi:

Esempio

```
<?php
class Car {
    function Car() {
        $this->model = "VW";
    }
}

// create an object
$herbie = new Car();

// show object properties
echo $herbie->model;
?>
```

PHP NULL Valore

Null è un tipo di dati speciale che può avere un solo valore: NULL.

Una variabile di tipo di dati NULL è una variabile che non ha alcun valore assegnato.

Suggerimento: Se una variabile viene creato senza un valore, esso viene automaticamente assegnato un valore NULL.

Le variabili possono anche essere svuotate impostando il valore NULL:

Esempio

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

PHP Resource

Il tipo speciale resource non è un tipo di dati effettivo. È la memorizzazione di un riferimento a funzioni e risorse esterne a PHP.

Un esempio comune di utilizzare il tipo di dati resource è una chiamata al database.

Noi non parliamo del tipo di risorsa qui, dal momento che è un argomento avanzato.

PHP 5 stringhe

Una stringa è una sequenza di caratteri, come "Ciao mondo!".
In questo capitolo vedremo alcune funzioni di uso comune per manipolare le stringhe.

Ottenere la lunghezza di una stringa

La funzione PHP strlen () restituisce la lunghezza di una stringa.

L'esempio seguente restituisce la lunghezza della stringa "Ciao mondo!":

Esempio

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

L'output del codice precedente sarà: 12.

esempio

Contare il numero di parole in una stringa

La funzione PHP str word count () conta il numero di parole in una stringa:

Esempio

```
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

L'output del codice precedente sarà: 2.

esempio

Invertire una stringa

La funzione PHP strrev() inverte una stringa:

Esempio

```
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

L'uscita del codice qui sopra sarà:! Dlrow olleh.

Cercare posizione di un testo specifico all'interno di una stringa

La funzione strpos cerca un testo specifico all'interno di una stringa.

Se viene trovata una corrispondenza, la funzione restituisce la posizione primo carattere della parola cercata. Se non viene trovata alcuna corrispondenza, verrà restituito FALSE.

L'esempio che segue ricerca il testo "mondo" nella stringa "Ciao mondo!":

Esempio

```
<?php
echo strpos("Ciao mondo!", "mondo"); // outputs 6
?>
esempio
```

L'output del codice precedente sarà: 6.

Suggerimento: La prima posizione carattere in una stringa è 0 (non 1).

Sostituisci testo all'interno di una stringa

La funzione PHP str_replace () sostituisce alcuni caratteri con alcuni altri caratteri in una stringa.

L'esempio seguente sostituisce il testo "mondo" con "Dolly":

Esempio

```
<?php
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
?> //L'uscita del codice qui sopra sarà: Hallo Dolly!
```

PHP 5 Costanti

Le costanti sono come le variabili, tranne che una volta che vengono definiti non possono essere cambiate.

Una costante è un identificatore (nome) per un valore semplice. Il valore non può essere modificato durante l'esecuzione dello script.

Un nome di costante valida inizia con una lettera o un trattino basso (nessun segno $ davanti al nome della costante).

Nota: A differenza delle variabili, le costanti sono automaticamente globali all'interno dell'intero script.

Creare una costante PHP

Per creare una costante, utilizzare la funzione define ().

Sintassi

```
define(name, value, case-insensitive)
```

parametri:

- *Nome* : Specifica il nome della costante
- *Valore* Specifica il valore della costante
- *-case insensitive* : Specifica se il nome della costante dovrebbe essere maiuscole e minuscole. Il valore predefinito è falso

L'esempio che segue crea una costante con un nome **maiuscolo e minuscolo**:

Esempio

```
<?php
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
```

L'esempio che segue crea una costante con un nome **case-insensitive**:

Esempio

```
<?php
define("GREETING", "Welcome to W3Schools.com!", true);
echo greeting;
?>
```

Costanti sono globali

Le costanti sono automaticamente globale e possono essere utilizzati in tutto l'intero script.

L'esempio seguente utilizza una costante all'interno di una funzione, anche se definito all'esterno della funzione:

Esempio

```
<?php
define("GREETING", "Welcome to W3Schools.com!");

function myTest() {
    echo GREETING;
}

myTest();
?>
```

PHP 5 Operatori

Operatori PHP

Operatori sono utilizzati per eseguire operazioni su variabili e valori.

PHP divide gli operatori nei seguenti gruppi:

- operatori aritmetici
- operatori di assegnazione
- Gli operatori di confronto
- operatori di incremento / decremento
- operatori logici
- operatori di stringa
- operatori Array

PHP Operatori aritmetici

Gli operatori PHP aritmetiche vengono utilizzati con valori numerici per eseguire operazioni aritmetiche comuni, come l'addizione, sottrazione, moltiplicazione ecc

PHP Operatori di assegnazione

Gli operatori di assegnazione PHP sono utilizzati con valori numerici per inserire il valore di una variabile.

L'operatore di assegnazione di base in PHP è "=". Ciò significa che l'operando a sinistra assume il valore dell'espressione di assegnazione sulla destra.

Gli operatori di assegnazione rispettano le stesse regole di altri linguaggi di programmazione come ad esempio il linguaggio C e Java.

PHP Operatori di confronto

Gli operatori di confronto PHP vengono utilizzati per confrontare due valori (numerici o stringa):

PHP Operatori di incremento / decremento

Gli operatori di incremento PHP sono utilizzati per incrementare il valore di una variabile.

Gli operatori PHP decremento vengono utilizzati per diminuire il valore di una variabile.

Operatori logici PHP

Gli operatori logici di PHP sono utilizzati per combinare istruzioni condizionali.

PHP Operatori di stringa

PHP ha due operatori che sono appositamente progettati per le stringhe.

operatore	codice php	tipo di variabile
assegnazione	$a = "pluto";	qualsiasi
addizione	$a = $b + $c;	numerico
sottrazione	$a = $b - $c;	numerico
moltiplicazione	$a = $b*$c;	numerico
divisione	$a = $b/$c;	numerico
modulo (resto divisione)	$a = $b%$c;	numerico
incremento di 1	$a++; *è come* $a = $a + 1;	numerico
decremento di 1	$a--; *è come* $a = $a -1;	numerico
uguaglianza	5==5	qualsiasi
disuguaglianza	5!=4	qualsiasi
maggiore, maggiore o uguale	$a>$b $a>=$b	qualsiasi
minore, minore o uguale	$a<$b $a<=$b	qualsiasi
and	($a<1) and ($b>0)	boolean
or	($a<1) or ($b>0)	boolean
not	!$a	boolean
concatenazione	$a="Bella"."monte"	stringa

PHP 5 if ... else ... elseif

Le istruzioni condizionali vengono utilizzate per eseguire diverse azioni in base alle condizioni diverse.

PHP Istruzioni condizionali

Molto spesso, quando si scrive codice, si desidera eseguire diverse azioni per condizioni diverse. È possibile utilizzare istruzioni condizionali nel codice per fare questo.

In PHP abbiamo le seguenti istruzioni condizionali:

- **if** - esegue codice se la condizione è vera
- **se ... else** - esegue codice se la condizione è vera e un altro codice se questa condizione è falsa
- **se ... elseif .... else** - esegue codici diversi per più di due condizioni
- **switch** - seleziona uno dei molti blocchi di codice da eseguire

---

PHP - L'istruzione if

L'istruzione if esegue un codice se la condizione è vera.

Sintassi

```
if (condition) {
    code to be executed if condition is true;
}
```

L'esempio che segue l'uscita volontà "Buona giornata!" se l'ora corrente (ORA) è inferiore a 20:

Esempio

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
}
?>
```

PHP - If ... ELSE

l'if .... else esegue del codice se la condizione è vera e un altro codice se questa condizione è falsa.

Sintassi

```
if (condition) {
    code to be executed if condition is true;
} else {
    code to be executed if condition is false;
}
```

L'esempio che segue visualizza " `Have a good day!"` se l'ora corrente è inferiore a 20, e "`Have a good night!"` altrimenti:

Esempio

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

PHP - l'if ... elseif .... else

l'if .... elseif ... else esegue codici diversi per più di due condizioni.

Sintassi

```
if (condition) {
    code to be executed if this condition is true;
} elseif (condition) {
    code to be executed if this condition is true;
} else {
    code to be executed if all conditions are false;
}
```

L'esempio che segue visualizza " Have a good morning!" se l'ora corrente è inferiore a 10, e " Have a good day!" se l'ora corrente è inferiore a 20. Altrimenti sarà uscita "Have a good night!";

Esempio

```
<?php
$t = date("H");

if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

L'istruzione switch PHP

L'istruzione switch è utilizzata per eseguire diverse azioni in base alle condizioni diverse.

Sintassi

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from all labels;
}
```

Questo è come funziona: In primo luogo abbiamo una sola espressione *n* (il più delle volte una variabile), che viene valutata una volta. Il valore dell'espressione viene confrontato con i valori per ciascun caso nella struttura. Se c'è una corrispondenza, viene eseguito il blocco di codice associato a quel caso. Utilizzare **break** per evitare che il codice esegua l'istruzione successiva automaticamente. L' **impostazione predefinita** istruzione viene utilizzata se non viene trovata alcuna corrispondenza.

```
<?php
$favcolor = "red";
switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

PHP Cicli

Spesso quando si scrive codice, si desidera che lo stesso blocco di codice da eseguire più e più volte di fila. Invece di aggiungere diversi codici linee quasi uguali in uno script, possiamo usare i cicli per eseguire un compito come questo.

In PHP, abbiamo le seguenti dichiarazioni di loop:

- **while** - loop attraverso un blocco di codice finché la condizione specificata è vera
- **do ... while** - loops attraverso un blocco di codice, una volta, e poi ripete il ciclo fino a quando la condizione specificata è vera
- **for** - loop attraverso un blocco di codice per un numero specificato di volte
- **foreach** - loop attraverso un blocco di codice per ogni elemento in un array

PHP il ciclo while

Il ciclo while esegue un blocco di codice finché la condizione specificata è vera.

Sintassi

```
while (condition is true) {
    code to be executed;
}
```

L'esempio seguente imposta dapprima una variabile $x a 1 ($ x = 1). Poi, il ciclo while continuerà a funzionare fintanto che $x è inferiore o uguale a 5 ($ x <= 5). $x aumenterà di 1 ogni volta che il ciclo viene eseguito ($x++):

Esempio

```
<?php
$x = 1;

while($x <= 5) {
    echo "The number is: $x <br>";
    $x++;
}
?>
```

PHP ciclo do ... while

Con il ciclo do ... while sarà sempre eseguito il blocco di codice una volta, sarà poi verificata le condizioni, e ripetuto il ciclo while finchè la condizione specificata è vera.

Sintassi

```
do {
    code to be executed;
} while (condition is true);
```

L'esempio seguente imposta dapprima una variabile $x variabile a 1 ($x = 1). Poi, il ciclo do while scriverà un messaggio in output, e quindi incrementare la variabile $x con 1. Quindi controlla la condizione ($x è inferiore o uguale a 5?), e il ciclo continuerà aad essere e eseguito fintanto che $x è inferiore o uguale a 5:

Esempio

```
<?php
$x = 1;

do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

Si noti che in un ciclo **do while** la condizione viene verificata dopo l'esecuzione del primo ciclo di istruzioni. Ciò significa che il do while esegue le istruzioni almeno una volta, anche se la condizione è falsa la prima volta.

L'esempio che segue imposta la variabile $x 6, quindi si esegue il ciclo, **e quindi la condizione viene verificata**:

Esempio

```
<?php
$x = 6;

do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

PHP 5 ciclo for

PHP for cicli eseguire un blocco di codice per un numero specificato di volte.

Il ciclo for viene utilizzato quando si sa in anticipo quante volte lo script dovrebbe funzionare.

Sintassi

```
for (init counter; test counter; increment counter) {
    code to be executed;
}
```

parametri:

- *contatore init* : Inizializza il valore del contatore del ciclo
- *contatore di test* : valutata per ogni iterazione del ciclo. Se TRUE, il ciclo continua. Se FALSE, il ciclo termina.
- *incremento del contatore* : aumenta il valore del contatore del ciclo

L'esempio seguente mostra i numeri da 0 a 10:

Esempio

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

PHP il ciclo foreach

Il ciclo foreach funziona solo su array, ed è utilizzato per eseguire il ciclo per ogni coppia chiave/valore in un array.

Sintassi

```
foreach ($array as $value) {
    code to be executed;
}
```

Per ogni iterazione del ciclo, il valore dell'elemento dell'array corrente viene assegnato a $valore e il puntatore viene spostato da uno, fino a raggiungere l'ultimo elemento dell'array.

Il seguente esempio illustra un ciclo che visualizza i valori dell'array proposto ($colori):

Esempio

```
<?php
$colori = array("red", "green", "blue", "yellow");

foreach ($colors as $valore) {
    echo "$value <br>";
}
?>
```

PHP 5 Funzioni

Il vero potere di PHP proviene dalle sue funzioni; ha più di 1000 funzioni incorporate.

PHP Funzioni definite dall'utente

Oltre alle funzioni predefinite, PHP ci permette di creare nostre funzioni.

Una funzione è un blocco di istruzioni che possono essere usate ripetutamente in un programma.

Una funzione non verrà eseguita immediatamente quando una pagina viene caricata.

Una funzione viene eseguita da una chiamata alla funzione.

Creazione e definizione di funzioni in PHP

Una funzione definita definita dall'utente inizia con la parola chiave "function":

Sintassi

```
function functionName() {
    code to be executed;
}
```

Suggerimento: Dare alla funzione un nome che riflette ciò che la funzione fa!

I nomi delle funzioni non sono case-sensitive.

Nell'esempio seguente, creiamo una funzione denominata "writeMsg ()". La parentesi graffa aperta ({) indica l'inizio del codice della funzione e la parentesi graffa di chiusura (}) indica la fine del codice della funzione. La funzione visualizza "Ciao mondo!". Per chiamare la funzione, basta scrivere il suo nome:

Esempio

```
<?php
function writeMsg() {
    echo "Hello world!";
}

writeMsg(); // call the function
?>
```

PHP Argomenti funzione

Le informazioni possono essere passata alle funzioni tramite argomenti. Un argomento è proprio come una variabile.

Gli argomenti vengono specificati dopo il nome della funzione, all'interno delle parentesi. È possibile aggiungere il numero di argomenti che vuoi, basta separarli con una virgola.

Il seguente esempio ha una funzione con argomento ($fname). Quando la funzione familyName() viene invocata, si passa come argomento un nome (es Jani), e il nome è usato all'interno della funzione, che visualizza il nome passato ed il cognome predefinito. In questo esempio si visualizzano i componenti di una famiglia invocando più volte la stessa funzione passando nomi diversi.

Esempio

```
<?php
function familyName($fname) {
    echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

Il seguente esempio ha una funzione con due argomenti ($fname e $year):

Esempio

```
<?php
function familyName($fname, $year) {
    echo "$fname Refsnes. Born in $year <br>";
}

familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

PHP con argomento valore d i dafault

L'esempio seguente mostra come utilizzare un parametro di default. Se chiamiamo la funzione setHeight () senza argomenti si assume il valore di default come argomento:

Esempio

```
<?php
function setHeight($minheight = 50) {
    echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

Funzioni PHP con valori di return

Per consentire una funzione restituisce un valore, utilizzare l'istruzione di ritorno:

Esempio

```
<?php
function sum($x, $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```

PHP 5 Array

Un array memorizza valori multipli in una singola variabile:

Esempio

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Che cosa è un array?

Un array è una variabile speciale, che può contenere più di un valore alla volta.

Se si dispone di un elenco di elementi (un elenco di nomi di automobili, per esempio), la memorizzazione delle vetture in singole variabili potrebbe essere la seguente:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

Questo metodo potrebbe andar bene con pochi dati. Cche cosa succederebbe se invece di avere 3 auto, se ne avrebbero 300?

La soluzione è quella di creare un array!

Un array può contenere molti valori sotto un unico nome, ed è possibile accedere ai valori facendovi riferimento attraverso un identificativo denominato indice.

Creare un array in PHP

In PHP, esiste la funzione array() per crearlo:

```
array();
```

In PHP, ci sono tre tipi di array:

- **Array indicizzati** - Array con un indice numerico
- **Array associativi** - array con chiavi di nome
- **Array multidimensionali** - Array che contengono uno o più array

PHP Array indicizzati

Ci sono due modi per creare array indicizzati:

L'indice può essere assegnato automaticamente (indice inizia sempre a 0), in questo modo:

```
$cars = array("Volvo", "BMW", "Toyota");
```

o l'indice può essere assegnato manualmente:

```
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

L'esempio seguente crea un array indicizzato di nome $cars, assegna tre elementi ad esso, e quindi stampa un testo che contiene i valori del'array:

Esempio

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Ottenere la lunghezza di un array – la funzione count()

La funzione count () viene utilizzata per restituire la lunghezza (numero di elementi) di un array:

Esempio

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
```

Ciclo for per array indicizzato

Per scorrere e stampare tutti i valori di un array indicizzato, è possibile utilizzare un ciclo for, in questo modo:

Esempio

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
```

PHP Array Associativi

array associativi sono array che utilizzano chiavi denominate che si assegnano a loro.

Ci sono due modi per creare un array associativo:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

o:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

Le chiavi di nome possono quindi essere utilizzati in uno script:

Esempio

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Ciclo per scorrere un array associativo

Per scorrere e stampare tutti i valori di un array associativo, è possibile utilizzare un ciclo foreach, in questo modo:

Esempio

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

PHP 5 Ordinamento Array

Gli elementi di un array possono essere ordinati in ordine alfabetico o numerico, decrescente o crescente.

PHP - Ordina funzioni per gli array

In questo capitolo, si passerà attraverso le seguenti funzioni sorta di matrice PHP:

- sort () - matrici di ordinamento in ordine ascendente
- matrici di ordinamento in ordine decrescente - rsort ()
- asort () - sort array associativi in ordine crescente, in base al valore
- ksort () - sort array associativi in ordine crescente, in base alla chiave
- arsort () - sort array associativi in ordine decrescente, in base al valore
- krsort () - sort array associativi in ordine decrescente, in base alla chiave

Ordina array in ordine crescente - sort ()

L'esempio seguente ordina gli elementi dell'array $cars in ordine alfabetico crescente:

Esempio

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);
?>
```

L'esempio seguente ordina gli elementi dell'array $numeri in ordine numerico crescente:

Esempio

```
<?php
$numeri = array(4, 6, 2, 22, 11);
sort($numeri);
?>
```

Ordina array in ordine decrescente - rsort ()

L'esempio seguente ordina gli elementi dell'array $vetture in ordine alfabetico decrescente:

Esempio

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
rsort($cars);
?>
```

L'esempio seguente ordina gli elementi dell'array $numeri in ordine progressivo:

Esempio

```
<?php
$numeri = array(4, 6, 2, 22, 11);
rsort($numeri);
?>
```

Ordina Array (Ordine crescente), in base al valore - asort ()

L'esempio seguente ordina un array associativo in ordine crescente, in base al valore:

Esempio

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
asort($age);
?>
```

Ordina Array (Ascendente Ordine), Secondo Key - ksort ()

L'esempio seguente ordina un array associativo in ordine crescente, in base alla chiave:

Esempio

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
ksort($age);
?>
```

Ordina Array (Ordine decrescente), in base al valore - arsort ()

L'esempio seguente ordina un array associativo in ordine decrescente, in base al valore:

Esempio

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
arsort($age);
?>
```

Ordina Array (Ordine decrescente), Secondo Key - krsort ()

L'esempio seguente ordina un array associativo in ordine decrescente, in base alla chiave:

Esempio

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
krsort($age);
?>
```

PHP 5 variabili globali - superglobals

Superglobals sono stati introdotti in PHP 4.1.0, e sono variabili built-in che sono sempre disponibili in tutti gli ambiti.

Variabili PHP globali - superglobali

Diverse variabili predefinite in PHP sono "superglobals", il che significa che essi sono sempre accessibili, indipendentemente dal campo d'applicazione - ed è possibile accedervi da qualsiasi funzione, classe o file senza dover fare nulla di speciale.

Il variabili PHP superglobale sono:

- $ GLOBALS
- $ _SERVER
- $ _REQUEST
- $ _POST
- $ _GET
- $ _FILES
- $ _ENV
- $ _COOKIE
- $ _SESSION

Questo capitolo spiegherà alcune delle superglobals, e il resto verrà spiegato nei capitoli successivi.

PHP $GLOBALS

$GLOBALS è una variabile globale super-PHP che viene utilizzato per accedere alle variabili globali da qualsiasi parte del script PHP (anche all'interno di funzioni o metodi).

Memorizza tutte le variabili globali di PHP in un array chiamato $ GLOBALS [*index*]. L' *indice* contiene il nome della variabile.

L'esempio seguente mostra come utilizzare il super variabile globale $ GLOBALS:

Esempio

```
<?php
$x = 75;
$y = 25;

function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

Nell'esempio sopra, poiché z è una variabile presente nell'array $GLOBALS, è anche accessibile dall'esterno della funzione!

PHP $_SERVER

$_SERVER è una variabile globale super-PHP che contiene informazioni sulle intestazioni, i percorsi ei luoghi di script.

L'esempio seguente mostra come utilizzare alcuni degli elementi in $ _SERVER:

Esempio

```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

La seguente tabella elenca gli elementi più importanti che possono andare dentro $ _SERVER:

Elemento/Codice	**Descrizione**
$_SERVER['PHP_SELF']	Returns the filename of the currently executing script
$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using
$_SERVER['SERVER_ADDR']	Returns the IP address of the host server

$_SERVER['SERVER_NAME']	Returns the name of the host server (such as www.w3schools.com)
$_SERVER['SERVER_SOFTWARE']	Returns the server identification string (such as Apache/2.2.24)
$_SERVER['SERVER_PROTOCOL']	Returns the name and revision of the information protocol (such as HTTP/1.1)
$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST)
$_SERVER['REQUEST_TIME']	Returns the timestamp of the start of the request (such as 1377687496)
$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string
$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request
$_SERVER['HTTP_ACCEPT_CHARSET']	Returns the Accept_Charset header from the current request (such as utf-8,ISO-8859-1)
$_SERVER['HTTP_HOST']	Returns the Host header from the current request

$_SERVER['HTTP_REFERER']	Returns the complete URL of the current page (not reliable because not all user-agents support it)
$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol
$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page
$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page
$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web server
$_SERVER['SCRIPT_FILENAME']	Returns the absolute pathname of the currently executing script
$_SERVER['SERVER_ADMIN']	Returns the value given to the SERVER_ADMIN directive in the web server configuration file (if your script runs on a virtual host, it will be the value defined for that virtual host) (such as someone@w3schools.com)
$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by the web server for communication (such as 80)

$_SERVER['SERVER_SIGNATURE']	Returns the server version and virtual host name which are added to server-generated pages
$_SERVER['PATH_TRANSLATED']	Returns the file system based path to the current script
$_SERVER['SCRIPT_NAME']	Returns the path of the current script
$_SERVER['SCRIPT_URI']	Returns the URI of the current page

PHP $_REQUEST

PHP $ _REQUEST viene utilizzato per raccogliere i dati dopo la presentazione di un modulo HTML.

L'esempio seguente mostra un modulo con un campo di input e un pulsante di invio. Quando un utente invia i dati cliccando su "Invia", i dati del modulo viene inviato al file specificato nell'attributo action del tag <form>. In questo esempio, si segnala a questo stesso file di dati del modulo per l'elaborazione. Se si desidera utilizzare un altro file PHP per elaborare i dati dei moduli, sostituirlo con il nome di vostra scelta. Poi, possiamo usare il super variabile globale $_REQUEST per raccogliere il valore del campo di input:

Esempio

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_REQUEST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>

</body>
</html>
```

PHP $_POST

PHP $_POST è ampiamente usato per raccogliere i dati dei moduli dopo aver inviato un form HTML con method = "post". $_POST è anche ampiamente utilizzata per passare variabili.

L'esempio seguente mostra un modulo con un campo di input e un pulsante di invio. Quando un utente invia i dati cliccando su "Invia", i dati del modulo viene inviato al file specificato nell'attributo action del tag <form>. In questo esempio, si segnala per il file stesso per l'elaborazione dei dati del modulo. Se si desidera utilizzare un altro file PHP per elaborare i dati dei moduli, sostituirlo con il nome di vostra scelta. Poi, possiamo usare il super variabile globale $ _POST per raccogliere il valore del campo di input:

Esempio

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_POST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>

</body>
</html>
```

PHP $_GET

PHP $_GET può anche essere utilizzato per raccogliere i dati dei moduli dopo aver inviato un form HTML con method = "get".

$_GET anche in grado di raccogliere i dati inviati nella URL.

Supponiamo di avere una pagina HTML che contiene un collegamento ipertestuale con i parametri:

```
<html>
<body>

<a href="test_get.php?subject=PHP&web=W3schools.com">Test $GET</a>

</body>
</html>
```

Quando un utente fa clic sul link "Test $GET", i parametri "soggetto" e "web" sono inviati a "test_get.php", e si può quindi accedere ai propri valori in "test_get.php" con $_GET.

L'esempio seguente mostra il codice in "test_get.php":

Esempio

```
<html>
<body>

<?php
echo "Study " . $_GET['subject'] . " at " . $_GET['web'];
?>

</body>
</html>
```