Breve manuale Php

Il linguaggio PHP permette di rendere dinamiche le pagine HTML dal lato del "servente". Questo insieme alla sua semplicità di utilizzo, alla somiglianza con la sintassi C e alle innumerevoli funzioni che permettono di interagire con basi di dati e servizi di ogni genere ne hanno decretato il successo in Internet. La diffusione è dovuta alla semplicità di configurazione e alla completa compatibilità con numerosi serventi HTTP. Il PHP permette, in modo semplice e diretto, di far interagire il cliente con le basi di dati ospitate sul servente tramite il semplice utilizzo di un comune navigatore. Tutti i maggiori DBMS sono gestiti da PHP

Le caratteristiche degli operatori aritmetici.

$a + $b	Addizione Somma tra $a e $b
$a - $b	Sottrazione Differenza tra $a e $b
$a * $b	Moltiplicazione Prodotto tra $a e $b
$a / $b	Divisione Quoziente tra $a e $b

Operatori logici

Gli operatori logici gestiti dal PHP sono riportati in tabella .
Le precedenze degli operatori sono riportate nel manuale ufficiale.

'$a and $b'	AND vera se $a e $b sono vere
'$a or $b'	OR vera se $a o $b è vera
'$a Xor $b'	XOR vera se $a o $b è vera ma non entrambe
'!$a'	NOT Negazione. Vera se $a non è vera
'$a && $b'	AND Simile a '**and**' ma con precedenza diversa
'$a || $b'	OR Simile a '**or**' ma con precedenza diversa

L'operatore fondamentale per l'assegnazione di un valore ad una variabile è '**=**'. La sintassi fondamentale è:

```
$a = 123;
$b = $a;
$c = "questa è una stringa";
$a = $a + $b;
$a += $b;
$a -= $b;
$a *= $b;
$a /= $b;
$a = $b = 30;
// o ancora in questo esempio sia $a che $b valgono 6
$a = 3; $b = $a += 3;
26 $a = "ciao a";
27 $a .= " tutti!!!";
```

OPERATORI DI CONFRONTO

'**$a == $b**'	Uguale vera se $a è uguale a $b
'**$a === $b**'	Identico vera se $a è uguale a $b e sono dello stesso tipo
'**$a != $b**'	Diverso vera se $a è diverso da $b
'**$a <> $b**'	Diverso vera se $a è diverso da $b
'**$a !== $b**'	Non Identico vera se $a non è uguale a $b o non sono dello stesso tipo
'**$a < $b**'	Minore vera se $a è minore di $b
'**$a > $b**'	Maggiore vera se $a è maggiore di $b
'**$a <= $b**'	Minore o uguale vera se $a è minore o uguale a $b
'**$a >= $b**'	Maggiore o uguale vera se $a è maggiore o uguale a $b

Struttura IF

```
if(condizione){
        istruzione 1
        istruzione 2
        .
        istruzione n
}
```

```
if(condizione){
        istruzione 1
        istruzione 2
        .
        istruzione n
}else{
        istruzione 1
        istruzione 2
        .
        .
        .
        istruzione m
}
```

Struttura While

```
<html>
<head>
<title>Ciclo while</title>
</head>
<body>
<br> Un ciclo WHILE<br><hr>
<?
$i=1;
while ($i <= 6){ ?>
 <br> <font size="<?=$i?>">
 Questo è un font con size = <?=$i?>
 </font>
<?
$i++;
}
?>
</body>
</html>
```

Struttura FOREACH

Va subito notato che il costrutto '**foreach**' è stato implementato dalla versione 4.0 in poi e quindi non è gestita dalle versioni precedenti del PHP. L'uso del '**foreach**' è indicato per la gestione degli array e mette a disposizione due sintassi principali:

```
foreach($variabile_array as $value){
        istruzioni
        ....
}
```

e:

```
foreach($variabile_array as $key => $value){
        istruzioni
        ...
}
```

il suo funzionamento, a differenza del semplice '**for**', non è molto intuitivo ed è vincolato all'utilizzo di un array. Si nota immediatamente che la sintassi non richiede l'utilizzo di indici né di incrementi, infatti il costrutto opererà su tutti gli argomenti dell'array indistintamente. Dunque questo strumento risulta comodo e versatile quando si ha a che fare con array con chiavi particolari o non consecutive (nel caso di chiavi numeriche).
Si supponga di avere un primo array composto da chiavi intere e consecutive simile al seguente:

```
<?
$array_1 = array (1, "a", 5, "c", "ciao"); // con chiavi da 0 a 4
print_r($array_1);
?>
```

e un secondo composto da chiavi miste e quindi non consecutive come questo:

```
<?
$array_2 = array (
"uno" => 1, // chiave stringa e valore intero
"frutta" => "mela", // chiave stringa e valore stringa
5 => "cinque", // chiave intero e valore stringa
40 => 30 // chiave intero e valore intero
);
print_r($array_2);
?>
```

I risultati dei due script sono semplicemente:

```
Array ( [0] => 1 [1] => a [2] => 5 [3] => c [4] => ciao )
Array ( [uno] => 1 [frutta] => mela [5] => cinque [40] => 30 )
```

Si supponga ora di non conoscere le chiavi del secondo array perchè assegnate dinamicamente da un codice precedente. In questo caso non si sarebbe in grado di operare sull'array con le strutture cicliche classiche perchè non c'è nessuna relazione tra le chiavi. Mentre nel primo la relazione tra le chiavi è semplice, sono interi successivi a partire da 0. In queste situazioni la struttura '**foreach**' è tanto indispensabile quanto semplice ed immediata da utilizzare.

Esercizi Php

//Utilizzo di Boolean
<?php
$bit = True;
if($bit) {
// $bit è vero
echo " il valore è
vero!";
}
	Stampare tabellina
	<?php
	$a = 2;
	$b = 0;
	$i = 1;
	echo " La tabellina del $a ". "\n";
	while($b<20)
	{
	$b = $a * $i;
	echo "$a x $i = $b\n"
	$i++;
	}
	?>

Le funzioni

Esempio 1

```
<?php
    /*      Questa funzione genera un messaggio di saluto  in base al valore della variabile $ora passatagli
      che contiene l'ora, con valore tra 0 e 24 */
//la variabile $ora viene inizializzata da codice si vedrà in seguito come inserire dati
function saluto($ora){
   if (5 < $ora && $ora <= 12){
     echo "Buon Giorno.";
   }elseif(12<$ora && $ora <= 18){
     echo "Buon Pomeriggio.";
   }elseif(18<$ora && $ora <= 24){
     echo "Buona Sera!";
   }else{
     echo "è tardi!!!!!!?!";
   }
 }
 //---------------- Programma principale
 $orario = date("H");  // estraggo l'ora attuale**
 // Richiamo la funzione e ci aggiungo del testo...
 saluto($orario);
 echo " Come va oggi?!?";
 // non serve riscrivere il codice posso inserire
 // il saluto con una sola RIGA di codice!!!
 echo "<br><br> Ancora un saluto: "; saluto($orario);
?>
```

**La funzione date

Questa funzione, come lascia intendere il suo nome, consente di **formattare una data ed un orario** sulla base delle impostazioni del server sul quale girano i nostri script. Per prima cosa vediamo un semplicissimo esempio:

```
<?php

$R=date("H");

echo($R);

/*H - indica l'ora, in formato 24 ore, con l'eventuale zero iniziale.

Quindi assumerà un valore compreso tra 01 e 24.*/

$R=date("d/m/y");    //mostra data nel formato gg/mm/aa

echo("\n  ".$R);

?>
```

Alcuni parametri utilizzabili

- **Y** - indica l'anno utilizzando quattro cifre. Per esempio 2003.
- **y** - indica l'anno utilizzando le ultime due cifre. Per esempio 03.
- **j** - indica il giorno del mese, senza l'eventuale zero iniziale. Quindi assumerà un valore compreso tra 1 e 31.
- **d** - indica il giorno del mese, con l'eventuale zero iniziale. Quindi assumerà un valore compreso tra 01 e 31.
- **L** - restituisce 1 se l'anno è bisestile, 0 se non lo è.

Esempio 2

```
<?php

  //---------------- Definizione delle funzioni

  function scheda($nome, $cognome){

    /*      Questa funzione ha lo scopo di chiarire il concetto di

      variabile locale e globale. Fare inoltre attenzione

      all'ordine con cui sono passate le variabili

    */

global $tel;//attenzione alla visibilità delle VARIABILI

   echo "<br>Scheda informativa utente: <br><br> ";

   echo "<br>Nome: ".$nome;
```

```
  echo "<br>Cognome: ".$cognome;
  echo "<br>Telefono: ".$tel;
  echo "<br><br>-----------------------<br>";
}
//---------------- Programma principale
$nome = "Gianluca";
$cognome = "Giusti";
$tel = "06/66666666";
  // richiamo la funzione che genera la scheda.
scheda($nome, $cognome);
// il nome delle variabili non importa! Importa invece l'ordine
// con cui vengono passate alla funzione! Ecco la prova:
scheda($cognome, $nome);
```

aggiungere mail ed indirizzo ho due possibilità

1. **Inserire var globali**
2. **Passare altre due var.**

Funzioni Ricorsive - Stampare il fattoriale n!

```
<?php
  function fattoriale($numero){
    if($numero <= 1){
      return 1;   // abbandona la ricorsione
    }else{
      return $numero*fattoriale($numero-1);  //la funzione richiama se stessa
    }
  }
  //---------------- Programma principale
echo "<br>Calcolo del fattoriale:<br>";

 for($i=2; $i<=10; $i++){
   echo "<br>Il fattoriale di $i è: ".fattoriale($i);
  }
```

trovare il MCD tra due numeri (inseriti nel programma es. a=2345, b=81)

con l'algoritmo DI EUCLIDE

Rappresentazione dell'algoritmo (pseudocodice)

Ripeti finché a<>b

Se a > b Sostituisci ad a il valore (a- b)

Altrimenti

Sostituisci ad b il valore (b-a)

Fine se

Fine Ripeti

Mostra il MCD che è pari ad a … (OPPURE b)

Gli array

Gli array possono essere creati tramite il costrutto 'array()' oppure tramite la valorizzazione degli elementi. A differenza dei classici linguaggi di programmazione, il PHP permette di indicizzare gli array non solo mediante interi non negativi, ma anche tramite stringhe. Di seguito la sintassi per creare un array tramite il costrutto 'array()'.

```
0 <?
1 array( [chiave =>] valore
2 , ...
3 )
4 // la chiave può essere un intero non negativo o una stringa
5 // il valore può essere qualunque
6 ?>
```

La chiave è contenuta tra le parentesi quadre perchè può essere omessa, se non viene specificata viene incrementato il valore intero. La sintassi di associazione è molto semplice ed intuitiva, per le chiavi intere positive: 'chiave => valore', mentre per le chiavi di tipo stringa vanno aggiunte le doppie virgolette '"chiave" => valore'. Per visualizzare il contenuto dell'array è possibile utilizzare la semplice istruzione 'print_r($array)'. Negli esempi seguenti si vedrà come questa funzione visualizza l'array.

```
0 <?
1 $a = array( "a", "b", 44, "d", "e");
2 print_r($a);
3 ?>
```

L'istruzione '**print_r($a)**' visualizzerà sullo schermo la struttura e il contenuto dell'array nel seguente modo:

```
Array ( [0] => a [1] => b [2] => 44 [3] => d [4] => e )
```

L'indice dei valori è stato incrementato automaticamente. È bene tenere sempre presente che le chiavi degli array partono da 0 e non da 1.

C'è la possibilità di specificare solo alcune chiavi e lasciare la gestione degli indici omessi all'interprete.

```
0 <?
1 $a = array( "a", "b", "c", "d", 8=>"e", 4=>"f", "g", 3=>"h");
2 print_r($a);
3 ?>
```

Questo il risultato dello script precedente:

```
Array ( [0] => a [1] => b [2] => c [3] => h [8] => e [4] => f [9] => g )
```

Un array può essere creato anche senza l'utilizzo del costrutto '**array()**', in questo caso la sintassi ricorda quella dei più comuni linguaggi di programmazione.

```
0 <?
1 $a[] = "a";
2 $a[] = "b";
3 $a[] = 44;
4 $a[] = "d";
5 $a[] = "e";
6 print_r($a);
7 ?>
```

Il secondo esempio può essere tradotto in questo modo:

```
0 <?
1 $a[] = "a";
2 $a[] = "b";
3 $a[] = "c";
4 $a[] = "d";
5 $a[8] = "e";
6 $a[4] = "f";
7 $a[] = "g";
8 $a[3] = "h";
9 print_r($a);
10 ?>
```

I due procedimenti per la creazione degli array sono equivalenti.

Gli array possono avere chiavi miste, come nell'esempio seguente, in cui alcune sono intere e alcune stringhe.

```
0 <?
1 $a["rosso"] = "a";
2 $a[] = "c";
3 $a[8] = "e";
4 $a["nero"] = "f";
5 $a[] = "g";
6 print_r($a);
7 ?>
```

Ecco il risultato:

```
Array ( [rosso] => a [0] => c [8] => e [nero] => f [9] => g )
```

Esercizi con i vettori

Primo esercizio

```
<?php

        $tp=array(17.5, 19.2, 21.8, 21.6, 17.5);
        $tp[5]=20.2;
        $tp[6]=16.6;

        for($i=0;$i<=6;$i++)
        {
                echo"$tp[$i]<br>";
        }

?>
```

Secondo esercizio

```
<?php

        $nome=array("Paolo","Marco","Giovanna","Maurizio","Francesca","Maria");
        $eta=array(47,21,42,54,51,17);

        for($i=0;$i<=5;$i++)
        {
                echo"$nome[$i], $eta[$i] anni<br>";

        }

?>
```

Array Associativi (esempio diverso dell'esercizio)

Si tratta di un array i cui elementi sono accessibili mediante nomi, quindi stringhe anziché indici puramente numerici. Questo non comporta però l'obbligo di utilizzare solo un tipo di indice: alcuni elementi dell'array possono avere un indice numerico, altri un indice di tipo stringa.
Segue un esempio in linguaggio PHP:

```
$auto["marca"] = 'FIAT';
$auto["modello"] = '500L';
$auto["colore"] = 'Blu';
$auto["anno"] = 1956;
$auto["revisionata"] = true;
```

L'indice, racchiuso tra le parentesi quadre è l'elemento attraverso il quale è possibile accedere al valore corrispondente dell'array. In PHP poi, come con altri linguaggi interpretati, gli array possono avere valori di tipo diverso. Nell'esempio abbiamo valori di tipo stringa, intero, booleano.

Scrivere un programma con un array associativo contenente Provincia - sigla della regione

Esercizio in Php senza modulo form

```
<?php
// array con indice numerico
echo(" array con indice numerico");
echo("<br>");
$regioni[0]="Roma";
$regioni[1]="Milano";
$regioni[2]="Napoli";
$regioni[3]="Bologna";
$regioni[4]="Torino";

$N=count($regioni);
for($xx=0;$xx<$N;$xx++)
{
        echo($regioni[$xx]);
        echo("<br>");
}
echo("<br>");
echo(" array associativo");
```

```
echo("<br>");
$regioAS["la"]="Roma";
$regioAS["lo"]="Milano";
$regioAS["ca"]="Napoli";
$regioAS["er"]="Bologna";
$regioAS["pi"]="Torino";
$N=count($regioAS);

for ($i = 0; $i < $N; $i++) {
        $Elemento = (each($regioAS));
        echo"<BR>chiave: $Elemento[key]";
        echo" valore: $Elemento[value]";
}
        echo("<br>");

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

Output

```
array con indice numerico
Roma
Milano
Napoli
Bologna
Torino

array associativo

chiave: la valore: Roma
chiave: lo valore: Milano
chiave: ca valore: Napoli
chiave: er valore: Bologna
chiave: pi valore: Torino
Key=la, Value=Roma
Key=lo, Value=Milano
Key=ca, Value=Napoli
Key=er, Value=Bologna
Key=pi, Value=Torino
```

Utilizzare i Form

acquisire un numero intero e visualizzare i suoi divisori

Pagina html

```
<html>
	<head>
		<title>Divisori</title>
	</head>
	<body bgcolor="lightblue">
		<!-- acquisire un numero intero e visualizzare i suoi divisori -->
		<h2>Digitare un numero intero</h2><p>
			<form action="divisori.php" method="post">
				<input type="text" size="10" maxlength="10"
name="numint"><p>
				<input type="submit" value="Invia">
				<input type="reset" value="Cancella">
			</form>
	</body>
</html>
```

pagina php

```
<?php
	echo"<html><body bgcolor='lightblue'>";
	//inizializzazione della variabile
	$numint=$_POST['numint'];

	if($numint!=floor($numint) || ($numint<=0))
		echo "Errore! Il numero digitato non &egrave; intero o non &egrave; positivo.
Ridigitare.";
	else {
		for($i=1;$i<=$numint;$i++) {
			if($numint % $i==0)
				echo  "Il numero $i &egrave; divisore di $numint.<br>";

		}
	}
?>
```

Creare un programma in php che controlli formalmente se sono stati digitati i 13 elementi del codice fiscale

pagina html

```
<html>
	<head>
		<title>Codice Fiscale</title>
	</head>
	<body>
```

```
        <form action="codicefiscale.php" method="POST">
        <h2>Controllo del codice fiscale</h2><p>
        <input type="text" size="13" maxlength="13" name="codice"> Inserire qui il codice
fiscale<br>
        <input type="submit" value="invia"><input type="reset" value="cancella">
        </form>
        </body>
</html>
```