Corso di PHP

5 – Funzioni

Prerequisiti

- Programmazione elementare in Php
- Tecnica top-down
- Concetto matematico di funzione
- Compilazione e link di programmi
- Esecuzione di funzioni
- Uso di parametri

Introduzione

In questa lezione fissiamo l'attenzione su come realizzare un programma modulare, ossia un programma scomposto in **funzioni** o **sottoprogrammi** o **moduli.**

Attraverso la tecnica **top-down**, è possibile rappresentare la soluzione di un problema come composta da sottoproblemi più semplici.

Una **funzione** deve:

- svolgere **un solo compito**
- essere di **uso generale**
- rappresentare la soluzione di un singolo **sottoproblema**

Informazioni generali

In questa Unità didattica mettiamo in pratica la tecnica **top down** l'importanza di suddividere un programma in parti più piccole dette **sottoprogrammi** o **funzioni**: ciò comporta:

- un codice più leggibile
- una progettazione più semplice
- una maggior semplicità
  - nel collaudo
  - nella correzione di eventuali errori
  - nella eventuale modifica del codice.

Il Php come tutti i moderni linguaggi, fornisce al programmatore sia la possibilità di definire proprie **funzioni utente,** sia una ricca libreria di **funzioni predefinite.**

Scrittura di una funzione

In PHP abbiamo la possibilità di **definire delle funzioni** che ci permettono di svolgere determinati compiti in diverse parti del nostro script o anche in script diversi, semplicemente richiamando la porzione di codice relativa, alla quale avremo attribuito un nome che identifichi la funzione stessa.

Vediamo alcuni esempi.

Scrittura di una funzione

Questa funzione ha solo il compito di eseguire una stampa

Intestazione della funzione: il nome *saluti* è deciso dal programmatore. Le parentesi tonde sono obbligatorie

```
<!– saluti.php →
<?php
function saluti()
{
  echo "Ciao, io sono una funzione!<BR>";
}

?>
```

Le funzioni hanno una struttura simile ad un programma e prevedono le parentesi graffe di inizio e di fine, all'interno delle quali si possono usare variabili, istruzioni e strutture di controllo

Istanza di una funzione

```
<!– saluti.php -->
<?php
function saluti()
{
  echo "Ciao, io sono una funzione!<BR>";
}
saluti();
?>
```

Questa è l'**istanza**, ossia la richiesta di esecuzione della funzione

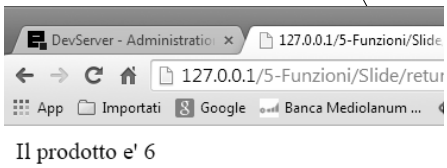
Valore di ritorno

Questa funzione esegue il prodotto di due valori costanti.

```
<!– return.php -->
<?php
function prodotto()
{
  return 2 * 3;
}
echo "Il prodotto e' ".prodotto();
?>
```

La parola chiave **return** indica che la funzione restituisce un valore al programma chiamante.

L'istanza della funzione produce l'esecuzione del calcolo richiesto.



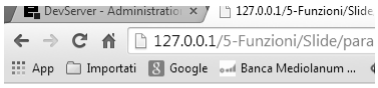
Uso di parametri

Una stessa funzione può operare con valori diversi mediante i **parametri**.

```
<!– parametri.php -->
<?php
function saluti($nome)
{
  echo "Ciao, io sono ".$nome;
}
$n="Anna";
saluti($n);
?>
```

La variabile *$nome* si dice **parametro formale** e il suo valore viene trasmesso alla funzione dal programma chiamante.

L'istanza va fatta passando alla funzione *$n* che prende il nome di **parametro attuale**.



Ciao, io sono Anna

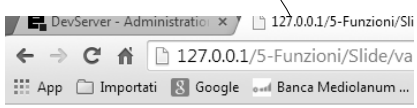
Si noti che in php i parametri vengono passati solo per valore

Variabili locali

All'interno di una funzione si possono usare variabili private, dette **variabilli locali**.

```
<!– var-locali.php -->
<?php
function somma($add1, $add2)
{   $s=$add1 + $add2;
    return $s;
}
$a = 5;
$b = 7;
echo "Somma = ".somma($a, $b);
?>
```

La variabile *$s* è nota solo nella funzione *somma()* e serve come appoggio per il calcolo.



Somma = 12

Nel programma chiamante si può assegnare la funzione ad una variabile

$calcolo = somma($a, $b);

Variabili globali

Le funzioni usate all'esterno di una funzione si dicono **variabilli globali** e per poter essere usate nella funzione occorre la parola chiave **global**.

```
<!– var-globali.php -->
<?php
function stampa($var1)
  {
    global $a;
    echo $a;
  }
$a = 'ciao a tutti';
$b = 'buongiorno';
stampa($b);
?>
```

$a può essere stampata nella funzione solo se si dichiara **global**, altrimenti si ha errore (variabile indefinita).

$b non viene stampata dalla funzione, in quanto non dichiarata **global**.

Ritorno di più valori

Se una funzione deve restituire più valori è possibile farli ritornare in un array.

```
<!– return-array.php -->
<?php
function multipli($num)
 {
   $doppio = $num * 2;
   $triplo = $num * 3;
   $quintuplo = $num * 5;
   $ris = array($doppio, $triplo, $quintuplo);
   return $ris;
}
$m= multipli(5);
list($a, $b, $c)= $m;
echo $a." ".$b." ".$c;
?>
```

$doppio, *$triplo* e *$quintuplo* sono usate come variabili locali. Il risultato *$ris* è un array con i 3 valori calcolati.

$list ricostruisce le 3 variabili calcolate dalla funzione per poi stamparle.

Funzioni su variabili

Funzioni per variabili e costanti	
define(*nome*, *valore* [, *case-sens*])	Definisce la costante *nome* e *valore*.
defined (*x*)	Controlla se esiste la costante. Restituisce **true** o **false**.
max(*x*, *y*, *z*, ...)	Confronta più variabili e restituisce quella di valore massimo.
min(*x*, *y*, *z*, ...)	Confronta più variabili e restituisce quella di valore minimo.
echo (*variabile*);	Stampa *variabile*
empty (*variabile*)	Controlla se *variabile* è nulla (stringa vuota, valore zero, variabile non definita o di valore NULL). Risponde con **true** o **false**.
isset (*variabile*)	Controlla se la variabile è definita (se è inizializzata o se diversa dal valore NULL.). Risponde con **true** o **false**.
is_null (*variabile*)	Controlla se la variabile è NULL (errore 'notice' se la variabile non è definita.). Risponde con **true** o **false**.
is_int (*variabile*) **is_integer** (*variabile*)	Controlla se la variabile è di tipo intero. Risponde con **true** o **false.**
is_long (*variabile*)	Controlla se *variabile* è intero lungo. Risponde con **true** o **false**.
is_float (*variabile*)	Controlla se *variabile* è numerica. Risponde con **true** o **false**.
is_double (*variabile*)	Controlla se *variabile* è numerica. Risponde con **true** o **false**.
is_real (*variabile*)	Controlla se *variabile* è numerica. Risponde con **true** o **false**.
is_string (*variabile*)	Controlla se *variabile* è una stringa. Risponde con **true** o **false**.
is_array (*variabile*)	Controlla se *variabile* è un array. Risponde con **true** o **false**.
is_numeric (*variabile*)	Controlla se *variabile* è alfanumerica. Risponde con **true** o **false**.
gettype (*variabile*)	A seconda di *variabile*, dà *boolean*, *integer*, *double*, *string*, *array*.
settype (*variabile*, *tipo*)	Imposta *variabile* a *tipo*
var_dump (*variabile*)	Restituisce al browser informazioni su *variabile* (nome e contenuto)
print_r (*variabile*)	Stampa *variabile* a video. Risponde con **true** o **false**.
unset (*variabile*)	Elimina la variabile. Non restituisce valori.

Funzioni su variabili

- esempi

```
$b = empty($a);              //$a non è ancora definita, quindi $b sarà vero
$a = 5;  $b = isset($a);     //vero
$b = is_float($a);           //falso: $a è un intero
$b = is_string($a);          //falso
$a = '5';  $b = is_int($a);  //falso: $a ora è una stringa
$b = is_string($a);          //vero
$b = is_numeric($a);         //vero: la stringa ha un contenuto numerico
$c = gettype($b);            //$c prende il valore 'boolean'
unset($a);                   //eliminiamo la variabile $a;
$b = is_null($a);            //vero, ma genera errore
if (empty($a)) {   print ('$a è vuota o non definita!');  }
else {  print ('$a contiene un valore'); }
```

Funzioni su stringhe

Funzioni stringa	
. (*punto*)	Concatenazione di stringhe
bin2hex (*stringa*)	Restituisce il codice esadecimale dei caratteri corrispondenti alle lettere della *stringa*.
chr (*codicecar*)	Restituisce il carattere associato al *codicecar*
explode (*str1*, *str2* [*numero*])	Converte la stringa in un array, i cui elementi sono le sottostringhe di str2 separate da str1 (che può essere uno spazio). *numero* (opzionale) indica quanti elementi può contenere l'array; se inferiore al numero di elementi, l'ultimo conterrà il resto della stringa. Restituisce un array.
floatval (*stringa*)	Converte una stringa in numero in doppia precisione
intval (*stringa*)	Converte *stringa* in numero
is_numeric (*stringa*)	Restituisce **true** se *stringa* rappresenta un numero
ltrim (*stiringa*)	Restituisce la stringa eliminando eventuali spazi a sinistra.
rtrim (*stringa*)	Restituisce la stringa eliminando eventuali spazi a destra.
strcmp (*str1*, *str2*)	Restituisce un valore che indica il risultato di un confronto di stringhe
strlen (*stringa*)	Restituisce il numero di caratteri che compongono una stringa.
strpos (*stringa1*, *stringa2*)	Restituisce la posizione della prima occorrenza di *stringa2* all'interno di *stringa2*, oppure **false**.
strstr (*stringa1*, *stringa2*)	Controlla se *stringa2* è contenuta in *stringa1*. Se presente, restituisce la sottostringa che inizia da *stringa2*. Se *stringa2* non è presente restituisce **false**.
strtolower (*stringa*)	Converte *stringa* in minuscolo.
strtoupper (*stringa*)	Converte *stringa* in maiuscole
strval (*numero*)	Restituisce una rappresentazione *numero* in forma di stringa
substr (*stringa*, *inizio*[, *num*])	Restituisce *num* caratteri di *stringa*, partendo dalla posizione *inizio* (se *inizio* è -1 ci troviamo sull'ultimo carattere)
trim (*stringa*)	Restituisce la stringa eliminando eventuali spazi prima e dopo.

Funzioni su stringhe

- esempi

```
$a = 'IERI ERA DOMENICA';
$b = strtolower($a);                /* $b diventa 'ieri era domenica */
strlen('abcd');                     // restituisce 4
trim(' Buongiorno a tutti ');       // restituisce 'Buongiorno a tutti'
substr('Buongiorno a tutti', 4);    /* 'giorno a tutti' (inizia dal quinto) */
substr('Buongiorno a tutti', 4, 6); /* 'giorno'(6 car. partendo dal quinto) */
substr('Buongiorno a tutti', -4);   // 'utti' (ultimi quattro)
substr('Buongiorno a tutti', -4, 2); /* 'ut' (2 car. a partire dal quartultimo) */
substr('Buongiorno a tutti, 4, -2); /* 'giorno a tut' (dal quinto al terzultimo) */
```

Funzioni su stringhe

- esempi

strpos('Domani è domenica', 'm'); //2 (prima 'm' trovata)
strstr('Domani è domenica', 'm'); /* 'mani è domenica' (parte dalla prima 'm') */
strtoupper('Buongiorno a tutti'); //'BUONGIORNO A TUTTI'
explode(',','Alberto,Mario,Giovanni'); /* suddivide la stringa in un array, separando un elemento ogni volta che trova una virgola; avremo quindi un array di tre elementi: ('Alberto','Mario','Giovanni')*/
explode(',','Alberto,Mario,Giovanni',2); /* in questo caso l'array può contenere al massimo due elementi, per cui nel primo elemento andrà 'Alberto' e nel secondo il resto della stringa: 'Mario,Giovanni' */

Funzioni su array

Funzioni su array	
array_key_exists (*key*, *array*)	Controlla se *key* è presente tra le chiavi di *array*. Restituisce **true** o **false**.
array_pop (*array*)	Toglie da *array* l'ultimo elemento e lo restituisce.
array_push (*array*, *val1*, *val2*…)	Unisce più valori ad un *array*. Restituisce il numero degli elementi che compone il nuovo *array*.
array_shift (*array*)	Toglie da *array* il primo elemento e lo restituisce.
array_search (*val*, *array*)	Se *val* è presente in *array*, restituisce la chiave, oppure **false**.
array_reverse (*array*, **true** | **false**)	Inverte l'ordine degli elementi di *array*. Se impostiamo **true** confermiamo le chiavi dell'array originarie. Restituisce un array.
array_unshift (*array*, *val1*, *val2*,...)	Immette i *val1*, *val2*, …all'inizio di *array*. Restituisce il numero degli elementi che compone il nuovo array.
arsort (*array*)	Ordina le chiavi degli elementi in ordine decrescente (non le modifica).
asort (*array*)	Ordina le chiavi degli elementi in ordine crescente (non le modifica).
count (*array*)	Conta il numero di elementi di array. Restituisce un numero.
current (*array*)	Restituisce il primo valore dell'array. Restituisce una stringa.
in_array (*val*, *array*)	Controlla se *val* è presente in *array*. Restituisce **true** o **false**.
is_array (*array*)	Controlla se *array* è una variabile array. Restituisce una stringa.
rsort (array)	Modifica l'array originario ordinando gli elementi con chiavi numeriche in ordine decrescente partendo da zero (resetta gli indici)
sizeof (*array*)	Restituisce il numero degli elementi nell'array.
sort (*array*)	Modifica l'array originario ordinando gli elementi con chiavi numeriche in ordine crescente partendo da zero (resetta gli indici)

Funzioni su array
- esempi

```
$arr = array('Luca', 'Giovanni', 'Matteo', 'Paolo', 'Antonio', 'Marco', 'Giuseppe');
$n = count($arr);                    // $n vale 7
$arr1 = array_reverse($arr);         /* $arr1 avrà gli elementi di arr invertiti */
echo $arr[1], '<br>';                // 'Giovanni'
echo $arr1[1], '<br>';               // 'Marco'
sort($arr);                          /* $arr viene ordinato */
$a = in_array('Giovanni', $arr);     // $a è vero (TRUE)
$a = in_array('Francesco', $arr);    // $a è falso (FALSE)
$ultimo = array_pop($arr);           // $ultimo è 'Paolo' (li avevamo ordinati!)
$ultimo = array_pop($arr);           /* ora $ultimo è 'Matteo' */
$primo = array_shift($arr);          // primo è 'Antonio'
```


Funzioni su array
- esempi

```
$a = array_unshift($arr, $ultimo, $primo);  /* 'Matteo' e 'Antonio' vengono
                                               reinseriti in testa all'array; $a riceve il
                                               valore 6 */
$stringa = implode(' ', $arr);              /* $stringa diventa 'Matteo Antonio
                                               Giovanni Giuseppe Luca Marco' */
                                            // Creiamo un array con chiavi associative:
$famiglia = array('padre' => 'Claudio', 'madre' => 'Paola', 'figlio' => 'Marco',
'figlia' => 'Elisa');
$fam1 = $famiglia;                          /* creiamo una copia del nostro array per
                                               poter fare esperimenti */
rsort($fam1);                               /* ora $fam1 sarà 'Paola', 'Marco', 'Elisa',
                                               'Claudio', con chiavi da 0 a 3 */
```

Funzioni su array

- esempi

```
$fam1 = $famiglia;                    //ripristiniamo l'array originale
arsort($fam1);                        /* di nuovo $fam1 sarà 'Paola',
                                      'Marco', Elisa', 'Claudio', ma
                                      ciascuno con la sua chiave
                                      originale ('madre', 'figlio', 'figlia',
                                      'padre') */
$a = array_key_exists('figlia', $fam1);   // $a è TRUE
$a = array_key_exists('zio', $fam1);      //$a è FALSE
$a = array_search('Claudio', $fam1);      //$a è 'padre'
$a = array_search('Matteo', $fam1);       //$a è FALSE
```


Funzioni su data

Funzioni data e ora	
date ("j/n/Y")	Restituisce la data corrente di sistema (**Y** - anno su 4 cifre, **y** - anno su 2 cifre)
date ("j")	Restituisce un numero intero compreso tra 1 e 31 inclusi che rappresenta il giorno del mese della *data* (**d** - giorno del mese su due cifre, **j** - giorno del mese senza lo zero)
date ("F")	Restituisce un numero intero tra 1 e 12 inclusi, che rappresenta il **mese** dell'anno di *data* (**n** - mese numerico senza lo zero, **m** - mese numerico su 2 cifre, **F** - mese testuale in lingua inglese **M** - mese testuale su 3 lettere in inglese)
date ("H : i : s")	Restituisce l'ora di sistema corrente (**H** - ora su due cifre , **G** – ora senza zero iniziale, **i** - minuti su due cifre, **s** - secondi su due cifre
date ("w")	Restituisce un numero intero che rappresenta il giorno della settimanam. (**w** - giorno della settimana, numerico (0=dom, 6=sab), **l** - giorno della settimana testuale in inglese, **D** - giorno della settimana su 3 lettere in inglese)
I separatori (: - / .)	Vanno utilizzati a piacimento, in quanto influenzano solo l'estetica. Provare ("H<j/Y-i#D").
mktime (*h, m, s, mm, gg, aa*)	Fornisce il *timestamp* di una data ben definita (inserire solo numeri interi). Per fare calcoli possiamo inserire numeri superiori a quelli usuali ad esempio se inseriamo 20 nei mesi verrà interpretato come 12+8, fornendo il *timestamp* dell'agosto dell'anno successivo. Restituisce un numero.
time()	Restituisce il *timestamp* attuale (il numero di secondi dal 01/01/1970)

Funzioni su data

- esempi

```
$a = mktime(15,56,20,4,24,2003); /* $a riceve il timestamp del 24/4/2003 alle
                                    15.56.20 */
$b = date('d M y - H:i', $a);    //$b sarà "24 Apr 03 - 15:56"
$a = mktime(14,0,0,4,24+60,2003); /* timestamp delle ore 14 di 60 giorni
                                    dopo il 24/4/2003 */
```


Argomenti

- Scrittura di una funzione
- Istanza di una funzione
- Valore di ritorno
- Uso di parametri
- Variabili locali
- Variabili globali
- Ritorno di più valori
- Funzioni su variabili
  - esempi
- Funzioni su stringhe
  - esempi
- Funzioni su array
  - esempi
- Funzioni su data
  - esempi