PHP ESSENTIALS #6

By WI400 Team

: costrutti, funzioni, inclusioni

Agenda

- echo
- die
- exit
- funzioni

Costrutto: echo

- *echo* non è propriamente una funzione ma un costrutto del linguaggio. Per questo motivo è possibile utilizzare echo senza le parentesi tonde
- Stampa una o più stringhe

```
<html>
<head>
<title>echo</title>
</head>
    <body>
    <?php
        echo "<h1>Hello World!</h1>";
    ?>
    </body>
</html>
```

Costrutto: echo

- è possibile utilizzare la funzione echo con una sintassi abbreviata che funziona solo se nel file di configurazione **php.ini** la direttiva short_open_tag è impostata su on:

  nota:con lo ZendServer questo è già abilitato

```
<?php
$stringa = "ciao";
?>
<!-- così viene stampata la variabile $stringa -->
<?=$stringa?>
```

Costrutto: die

- *die* non è propriamente una funzione ma un costrutto del linguaggio
- Stampa un messaggio e termina l'esecuzione dello script corrente
- E' necessario utilizzare le parentesi tonde se si vuole stampare un messaggio prima di terminare l'esecuzione dello script

  Esempio: die("Fatal Error");
- E' possibile omettere le parentesi se si vuole solo terminare l'esecuzione dello script senza effettuare nessun output

  Esempio: die;
- *die* è equivalente al costrutto exit

Esempio: die

```
<html>
<head>
<title>die</title>
</head>
    <body>
    <?
    $n = 5;
    if ($n > 1)
        die("<h1>$n è maggiore di 1!!!</h1>");
    ?>
    <h1>$n è minore o uguale ad 1!</h1>
    </body>
</html>
```

Funzioni

- introduzione,
- funzioni native,
- creare e richiamare funz,
- visibilità,
- funz. di variabili

Funzioni: introduzione

- Una funzione è un blocco di codice che esegue uno specifico compito
- Il PHP mette a disposizione molte funzioni già definite all'interno del linguaggio
- Talvolta però le funzioni che il PHP mette a disposizione non sono sufficienti
- In questi casi la soluzione migliore è quella di definire una funzione personalizzata
- Una funzione può essere utile anche per organizzare il codice e per riutilizzare il codice senza doverlo necessariamente riscrivere

Funzioni: Creare funzioni

- E’ possibile definire una funzione attraverso la seguente sintassi:

```
<?php
function nomefunzione ( [ parametri [ , … ] ] )
{
/* Codice della funzione */
}
```

- Il nome deve iniziare con una lettera o con il carattere underscore e può contenere caratteri, numeri e underscore
- Una funzione può contenere sia codice PHP che codice HTML:

```
<?php
function tabella() {
?>
    <table></table>
    <?php
}
```

Funzioni: Creare funzioni

- Le funzioni possono accettare dei parametri

```php
<?php
function nomefunzione($param1, $param2) {
    /* E' possibile utilizzare $param1 e $param2 */
}
```

- E' possibile definire I valori di default per rendere uno o più parametri opzionali

```php
<?php
function nomefunzione($param1, $param2 = 'default') {
    /* E' possibile utilizzare $param1 e $param2 */
}
```

- E' possibile fare in modo che la funzione accetti I parametri per riferimento e non per valore

```php
<?php
function nomefunzione(&$param1, &$param2) {
    /* $param1 e $param2 saranno il riferimento ai valori
       passati quando la funzione viene richiamata */
}
```

Funzioni: Creare funzioni

- Le funzioni possono anche restituire un risultato attraverso la parola chiave **return**:

```
<?php
function nomefunzione() {
    return 'risultato';
}
```

- Le funzioni possono restituire un solo valore

- Per restituire più di un valore è necessario farlo attraverso un array

Funzioni: Richiamare funzioni

- Il codice di una funzione non verrà mai eseguito fino a quando la funzione non viene esplicitamente richiamata

- Sia le funzioni del PHP che le funzioni definite dall'utente vengono richiamate allo stesso modo

```
<?php
$risultato = nomefunzione( [ parametri [ , … ] ] );
```

- Al termine dell'esecuzione la variabile *$risultato* conterrà il valore restituito dalla funzione *nomefunzione()*

Funzioni: Richiamare funzioni

- Tutti i parametri definiti come obbligatori devono essere passati quando viene richiamata la funzione

```php
<?php
function concatena($stringa1, $stringa2) {
    return $stringa1 . $stringa2;
}
echo concatena ( 'Hello', 'World' );
```

- Se un parametro è definito come opzionale, può essere tralasciato al momento in cui viene richiamata la funzione e verrà utilizzato il suo valore di default

```php
<?php
function concatena($stringa1, $stringa2 = 'World') {
    return $stringa1 . $stringa2;
}
echo concatena ( 'Hello' );
```

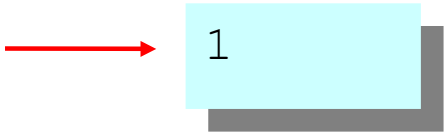
→ `HelloWorld`

Funzioni: variable scope

- Se non si utilizzano le funzioni, qualsiasi variabile creata può essere utilizzata in qualsiasi punto dello script
- Le funzioni hanno invece un loro set di variabili
- Le variabili definite all'interno delle funzioni sono diverse rispetto alle variabili definite al di fuori anche se hanno lo stesso nome

qual'è l'output ?

```php
<?php
$a = 1;
function somma() {
    $a = $a + 1;
}
somma ();
echo $a;
```



Funzioni: variable scope

- Se dall'interno di una funzione si vuole accedere alle variabili globali si deve utilizzare la parola chiave ***global***

qual'è l'output ?

```php
<?php
$a = 1;
function somma() {
    global $a;
    // Accediamo alla variabile globale $a
$a = $a + 1;
}
somma ();
echo $a;
```

→ 2

Funzioni: funzioni di variabili

- Così come per le variabili di variabili, è possibile richiamare una funzione PHP sulla base del contenuto di una variabile:

qual'è l'output ?

```
<?php
function verde() {
    echo 'E' stato selezionato il colore verde';
}

$colore = verde;
$colore ();
```

→

```
E' stato selezionato il colore verde
```

Esercizio 7

- Creare una funzione *ripetiSaluto()* che accetta due argomenti: una stringa (obbligatoria) ed un intero opzionale.
- Stampare la stringa passata come primo argomento tante volte in base al numero passato come secondo argomento.
- Il secondo argomento deve avere come valore di default 1.
- Provare la funzione scritta richiamandola in questi due modi:

```
ripetiSaluto('Benvenuto nel nostro portale', 3);
ripetiSaluto('Benvenuto nel nostro portale');
```

Esercizio 7: solution

```php
<?php
function ripetiSaluto($stringa, $numero = 1) {
    for($i = 0; $i < $numero; $i ++) {
        echo "<p>$stringa</p>";
    }
}

ripetiSaluto('Benvenuto nel Portale',3);
ripetiSaluto('Benvenuto nel Portale');
```

→

```
Benvenuto nel Portale

Benvenuto nel Portale

Benvenuto nel Portale

Benvenuto nel Portale
```

Esercizio 8:

- Creare una funzione chiamata inverti() che accetta due argomenti obbligatori.
- La funzione dovrebbe invertire i valori nelle variabili passate come argomento:

```php
<?php
$tre = 3;
$cinque = 5;
inverti ( $tre, $cinque );

echo $tre;// dovrebbe risultare 5

echo $cinque; // dovrebbe risultare 3
```

Esercizio 8: solution

- Considerando che $tmp è una variabile locale (per la funzione inverti), non dobbiamo preoccuparci del fatto che altre funzioni utilizzino o meno una variabile che si chiama $tmp.
- Questo spiega l'importanza delle diversità delle variabili locali dalle variabili globali.

```php
<?php
function inverti(&$var1, &$var2) {
    $tmp = $var1;
    $var1 = $var2; /* $var1 è stata passata per riferimento */
    $var2 = $tmp; /* $var2 è stata passata per riferimento */
}
```

Includes

- I costrutti include e require sono molto utili per includere codice memorizzato in file separati:

```
include '/percorso/directory/file.php';
```

- require genera un fatal error se il file non viene trovato, mentre include genera un warning.

```
require '/percorso/directory/file.php';
```

- Sono molto utili per la modularizzazione e per una migliore organizzazione del codice.

- Generalmente contenuti HTML e librerie di funzioni vengono memorizzate in file separati ed incluse dove utilizzate.

- *Nota: esistono rispettivamente le funzioni **require_once()** e **include_once()** per "inglobare" uno script una e una sola volta all'interno della stessa sessione*

Includes

- Un classico esempio di utilizzo di include e require è quello di avere un header ed un footer che siano identici per ogni file:

```
<?php include '/percorso/directory/header.html';?>

<p>Hello, World!</p>

<?php include '/percorso/directory/footer.html';?>
```

Esercizio 9:

- Creare due file: header.php e body.php

- Entrambi i file devono utilizzare la funzione echo
  - header.php - echo “Benvenuto”;
  - body.php - echo “Grazie per aver visitato il nostro portale”;

- Il file body.php deve includere il file header.php

- Ulteriori passi:
  - Dichiarare una variabile in un file ed utilizzarla (sempre attraverso il costrutto echo) nell’altro file. E’ importante l’ordine?

  - Cosa succede se si dichiara una funzione in un file e si richiama nell’altro? E’ importante l’ordine?

Esercizio 9: solution

- header.php:

```php
<?php
    echo 'Benvenuto';
```

- body.php:

```php
<?php
    include 'header.php';
    echo 'Grazie per aver visitato il nostro portale';
```

QUESTION TIME ?

Nome_____ Cognome_____ Data_____

ARRIVEDERCI

TITOLO