



PHP ESSENTIALS #7

By WI400 Team

: array, sessioni



▪ arrays

- ✓ enumerativi
- ✓ associativi
- ✓ uso di arrays

Array

- Gli array sono il modo più corretto per memorizzare una serie di dati correlati tra loro
- Soluzione migliore di utilizzare variabili diverse
- E' possibile utilizzare una sola variabile per memorizzare diverse informazioni
- Ogni elemento è composto da una coppia di chiave e valore
- Esistono due tipi di array:
 - ✓ Array enumerativi
 - ✓ Array associativi

| 3

| 12/05/10



Arrays: Enumerativi

- Gli array enumerativi hanno chiavi di tipo numerico
- Sono utili quando non abbiamo bisogno di conoscere il significato delle chiavi (vogliamo solo una collezione di valori)
- Creare array enumerativi è molto semplice:

```
<?php  
$studenti = array ();  
$studenti [] = 'Marco';  
$studenti [] = 'Andrea';
```

| 0

| 00/00/00



Arrays: Enumerativi

- E' possibile anche "popolare" un array direttamente dalla definizione `array()`:

```
$studenti = array('Marco', 'Andrea');
```

- E' possibile utilizzare un array come una normale variabile
- E' necessario però indicare l'indice dell'elemento al quale accedere secondo la seguente sintassi:

```
echo $studenti[1];
```

qual'è l'output ?

Andrea

Arrays: Enumerativi

- E' possibile scorrere tutti gli elementi di un array attraverso l'utilizzo di un ciclo `for`:

```
for($i = 0; $i < count( $studenti ); $i++) {  
    echo "<p>$studenti[$i]</p>";  
}
```

- Il modo migliore di scorrere tutti gli elementi di un array è quello di utilizzare il ciclo `foreach`:

```
foreach ( $studenti as $studente ) {  
    echo "<p>$studente</p>";  
}
```

Esercizio 10:

- crea una array() enumerativa \$favorites, nella quale ogni elemento è un nome.
- scorri l'intera array() concatenando la stringa “ nome” ad ogni elemento.
- scorri nuovamente l'intera array() per stamparne gli elementi

Esercizio 10: solution

- creazione e inizializzazione dell'array
- ciclo for per aggiunta stringa
- ciclo foreach per stampa

```
<?php
$favorites = array ( );
$favorites [] = 'Pippo';
$favorites [] = 'Pluto';
for($i = 0; $i < count ( $favorites ); $i ++) {
    $favorites [$i] .= ' nome';
}
foreach ( $favorites as $name ) {
    echo "<p>$name</p>";
}
```

Pippo nome
Pluto nome

Arrays: associativi

- Gli array associativi sono gli array che hanno come chiavi delle stringhe
- Sono utili quando le chiavi ci servono a descrivere i dati contenuti
- La creazione di un array associativo è molto simile alla creazione di un array enumerativo
- E' necessario indicare la chiave per ogni elemento dell'array

```
<?php  
$studenti = array ();  
$studenti ['nome'] = 'Mario';  
$studenti ['cognome'] = 'Rossi';
```

Arrays: associativi

- Anche gli array associativi possono essere creati attraverso la funzione array():

```
$studenti = array (  
    'nome' => 'Mario',  
    'cognome' => 'Rossi',  
);
```

- E' possibile utilizzare un array come una normale variabile
- E' necessario però indicare l'indice dell'elemento al quale accedere secondo la seguente sintassi:

qual'è l'output ?

```
echo $studenti ['nome'];
```

Mario

Arrays: associativi

- qual'è l'output del seguente codice ?

```
<?php  
$students = array (  
    'name' => 'John',  
    'location' => 'London',  
    'address' => 'Piccadilly Street'  
);  
$elem='location';  
echo $students['$elem'];
```

Nota: ricordi gli apici singoli ?
`echo $students[$elem]`
è corretto

....a notice...

Arrays: associativi

- Una differente sintassi del ciclo **foreach** può essere utilizzata per estrarre anche le **chiavi** degli elementi oltre ai loro **valori**:

```
foreach ( $studenti as $chiave => $valore ) {  
    echo "<p>Chiave: $chiave; Valore: $valore</p>";  
}
```

Esercizio 11:

- Creare un array associativo chiamato \$io che contiene il vostro nome e il vostro cognome.
- Scorrere l'intero array attraverso l'utilizzo del ciclo **foreach** e visualizzare
 - ‘Il mio nome è ...’ e
 - ‘Il mio cognome è...’utilizzando la struttura di controllo **if** per controllare se l'elemento corrente è nome o cognome.

Esercizio 11: solution

```
<?php
$io = array (
    'nome' => 'Mario',
    'cognome' => 'Rossi'
);

foreach ( $io as $chiave => $valore ) {
    if ($chiave == 'nome') {
        echo "<p>Il mio nome è $valore.</p>";
    } else {
        echo "<p>Il mio cognome è $valore.</p>";
    }
}
```



Il mio nome è Mario.

Il mio cognome è Rossi.

Arrays: utilizzo e manipolazione

- Durante lo sviluppo di applicazioni php può tornare comodo “stampare” il contenuto delle array per l'attività di debugging

```
echo $students;
```

→ Array

Arrays: utilizzo e manipolazione

- La funzione **print_r()** è utile per eseguire il debug:

```
<?php
$me = array ();
$me ['nome'] = 'Mario';
$me ['cognome'] = 'Rossi';

print "<pre>";
print_r ( $me );
print "</pre>";
```

→ Array
(
 [**nome**] => Mario
 [**cognome**] => Rossi
)

Arrays: utilizzo e manipolazione

- La funzione **explode()** consente di creare un array da una lista di valori separati dallo stesso carattere

```
<?php  
$string = 'one:two:three';  
$array = explode(':', $string);  
print_r($array);
```

Array (
[0] => one
[1] => two
[2] => three
)

Arrays: utilizzo e manipolazione

- È possibile utilizzare la funzione **list()** in abbinamento a **explode()**, per creare variabili separate invece di un array

```
<?php  
$string = 'one:two:three';  
$array = explode(':', $string);  
list($one, $two, $three) = explode(':', $string);  
echo $two;
```

two

Arrays: utilizzo e manipolazione

- con la funzione **implode()** è possibile creare una stringa delimitata da un separatore partendo da una array

```
<?php  
$array = array();  
$array[] = 'one';  
$array[] = 'two';  
$array[] = 'three';  
$string = implode(':', $array);  
echo $string;
```



one:two:three

Arrays: utilizzo e manipolazione

- la funzione **range()** consente di creare una array dato da un range di elementi

```
<?php  
$months = range(0, 12);  
print_r($months);
```



```
Array (  
[0] => 0  
[1] => 1  
.....  
[12] => 12  
)
```

- È possibile utilizzarla anche con valori alfabetici

```
<?php  
$alphabet = range('a', 'z');  
print_r($alphabet);
```



```
Array (  
[0] => a  
[1] => b  
.....  
[25] => z  
)
```

Arrays: utilizzo e manipolazione

- **array_search()** va utilizzata per cercare un valore all'interno di una array

```
<?php  
$students=array('John', 'Mary', 'Janice');  
$key = array_search('John', $students);
```

- ✓ nota: *array_search()* ritorna la chiave dell'elemento trovato, ritorna false quando il valore non viene trovato

```
<?php  
$students=array('John', 'Mary', 'Janice');  
if(!array_search('John', $students)) {  
    echo "John non e' tra gli studenti";  
} else {  
    echo "John e' presente !";  
}
```

qual'è l'output ?

John non e' tra gli studenti

Arrays: utilizzo e manipolazione

- È possibile unire delle array con **array_merge()**

```
<?php  
$one = array('a', 'b');  
$two = array('c', 'd');  
$three = array_merge($one, $two);  
print_r($three);
```

Array (
[0] => a
[1] => b
[2] => c
[3] => d
)

- ✓ nota: le array enumerative vengono reindividuate. in caso di “collisioni” di valori gli elementi vengono duplicati
- ✓ nota: in caso di “collisioni” tra array associative (identiche chiavi) le chiavi della seconda array sovrascrivono le prime

Arrays: utilizzo e manipolazione

- La funzione **array_sum()** calcola la somma degli elementi di un array:

```
<?php  
$mesi = range ( 1, 12 );  
$somma = array_sum ( $mesi );  
echo $somma;
```



78

Arrays: utilizzo e manipolazione

- Esistono diverse funzioni relative all'ordinamento e ognuna di esse è di difficile comprensione
-fino a che non le provate...
- **array_reverse()** ordine in senso inverso gli elementi
- **sort()** ordina una array per valore ascendente, **rsort()** discendente
- **ksort()** ordina una array per chiave ascendente, **krsort()** discendente
- **asort()** ordina una array per valore (ascendente) mantenendo l'associazione con la chiave, **arsort()** discendente

Esercizio 12:

- crea una array \$array dalla stringa seguente:

\$string = 'pinocchio=collodi&codice da vinci=brown';

- Dovresti crearne una array associativa di due elementi, l'output di print_r(\$array) dovrebbe produrre:

```
Array (
    [pinocchio] => collodi
    [codice da vinci] => brown
)
```

Esercizio 12: solution

- creazione di una prima array dalla stringa
- ciclo foreach per iterare con chiave
- Ulteriore scomposizione della stringa per creazione array finale

```
<?php
$string = 'pinocchio=collodi&codice da vinci=brown';
$array = array ( );
$books = explode ( '&', $string );

foreach ( $books as $book ) {
    list ( $title, $author ) = explode ( '=', $book );
    $array [$title] = $author;
}
print_r($array);
```