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

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';
```

Arrays: Enumerativi

- E’ possibile anche “popolare” un array direttamente dalla definizione array():

```php
$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:

```php
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>";
}
```

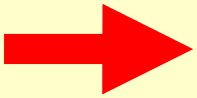
Esercizio 10:

- 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
<?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

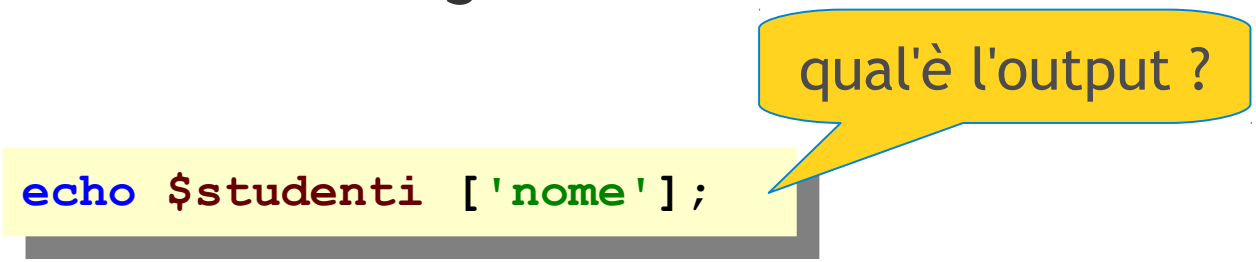
```php
<?php
$studenti = array ();
$studenti ['nome'] = 'Mario';
$studenti ['cognome'] = 'Rossi';
```

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:



Arrays: associativi

- qual'è l'output del seguente codice ?

```php
<?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
<?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'attivitità di debugging

```
echo $students;
```

Arrays: utilizzo e manipolazione

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

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

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

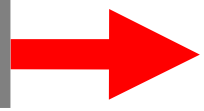


```
Array
(
    [nome] => Mario
    [cognome] => Rossi
)
```

Arrays: utilizzo e manipolazione

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

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

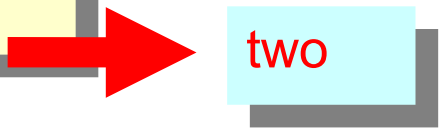


```
Array (
[0] => one
[1] => two
[2] => three
)
```

Arrays: utilizzo e manipolazione

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

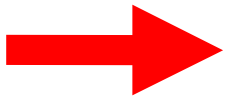
```php
<?php
$string = 'one:two:three';
$array = explode(':', $string);
list($one, $two, $three) = explode(':', $string);
echo $two;
```



Arrays: utilizzo e manipolazione

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

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

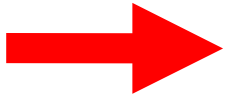


one:two:three

Arrays: utilizzo e manipolazione

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

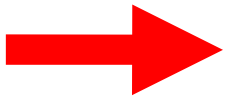
```php
<?php
$months = range(0, 12);
print_r($months);
```



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

- È possibile utilizzarla anche con valori alfabetici

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



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

Arrays: utilizzo e manipolazione

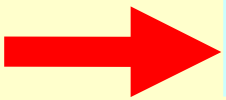
- **array_search()** va utilizzata per cercare un valore all'interno di una array

```php
<?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
<?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 ?

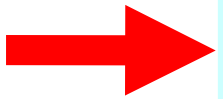


John non e' tra gli studenti

Arrays: utilizzo e manipolazione

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

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



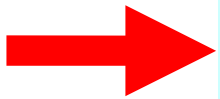
```
Array (
[0] => a
[1] => b
[2] => c
[3] => d
)
```

- *nota: le array enumerative vengono reindicizzate.*
  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 **sovrascivono** le prime*

Arrays: utilizzo e manipolazione

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

```php
<?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
<?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);
```

Sessioni

- Http e Stateless
- Cookies e stato
- Sessioni e php

Sessioni: Cookies e Stato

- l'HTTP è per definizione STATELESS
- ogni transazione viene definita atomic
- ogni richiesta, tra browser e server, genera un processo che gestisce un file e poi termina
- ogni richiesta è separata dalle altre, e non vengono passate informazioni , su quanti e quali pagine sono state richieste prima dell'attuale
- con PHP possiamo trasformare l'HTTP da STATELESS a STATEFULL.
  - usando cookie e/o le sessioni

Sessioni: Cookies e Stato

- Netscape sviluppò un meccanismo di gestione dello stato chiamato cookie

- un cookie è un stringa contenente dati che vengono generati sul server e memorizzati sul client
  nota: i dati sono una serie di coppie nome/valore
- il browser deve essere configurato per "accettare" i cookie"
- i cookie vengono automaticamente trasmessi dal client al server

Sessioni: Cookies e Stato

- Quando impostiamo un cookie, il PHP non fa altro che includere un cookie nell'header della risposta inviata al client.

- Questo significa che per verificare se un client abbia effettivamente "accettato" il cookie abbiamo bisogno di almeno **due** richieste

- *E' impossibile verificare in un'unica richiesta (dal PHP) se il cookie è stato correttamente impostato oppure no*

Sessioni: Cookies e Stato

- Lo scenario tipico include due transazioni HTTP:
  - ✓ Il client richiede una pagina PHP. Lo script PHP viene eseguito e la risposta include l'header Set-Cookie.
  - ✓ Il client richiede una pagina e include un header Cookie nella richiesta. Lo script PHP viene eseguito ed è a questo punto che possiamo accedere al contenuto del cookie inviato facendo riferimento all'array superglobale $_COOKIE.

Sessioni: Cookies e Stato

- setcookie() è la funzione PHP per scrivere, modificare o cancellare un cookie
- setcookie() deve essere chiamata prima che qualsiasi output venga inviato al browser
- Solo i primi due parametri sono obbligatori:
  - name
  - value
- I parametri opzionali di setcookie() sono:
  - expire: La data di scadenza del cookie (Unix timestamp)
  - path: Il percorso sul server dove il cookie sarà disponibile (con '/' il cookie sarà disponibile in tutto il dominio, con '/directory/' il cookie sarà disponibile solo nella directory 'directory')
  - domain: Il dominio dove il cookie deve essere disponibile
  - secure: Indica che il cookie deve essere inviato solo attraverso connessioni HTTPS
  - httponly: Rende il cookie accessibile solo attraverso il protocollo HTTP (no javascript)

Sessioni: Cookies e Stato

- Creare un cookie con nome “username” e valore “mario”:

```
setcookie ( 'username', 'mario' );
```

- Cambiare il valore del cookie con nome “username” in “nuovovalore”

```
setcookie ( 'username', 'nuovovalore' );
```

- E’ possibile cancellare il cookie con nome “username” in due modi diversi

```
setcookie ( 'username', NULL );
setcookie ( 'username', '', time () - 3600 );
```

- Una volta che il cookie è stato settato, è possibile accedervi attraverso l’array superglobale $_COOKIE.

```
$cookie = $_COOKIE ['username'];
```

Sessioni: Creazione/lettura cookie (Esempio)

- Creiamo una semplice pagina PHP che visualizza la data e l'ora dell'ultimo accesso del nostro visitatore
- Impostiamo il cookie "ultimavisita"
- Non impostiamo il tempo di validità del cookie in modo che il cookie venga eliminato alla chiusura del browser

```
<?php
if (isset ( $_COOKIE ['ultimavisita'] ))
    $messaggio = "Bentornato";
else {
    $messaggio = 'Benvenuto';
    setcookie ( 'ultimavisita', time () );
}
?>
<html><head><title>Esempio cookie</title></head><body>
<h1><?echo $messaggio?></h1>
<?php
if (isset ( $_COOKIE ['ultimavisita'] )) {
    echo "La tua ultima visita: " . date ( "d/m/Y H:i:s", $_COOKIE [ultimavisita] );
    echo "<p><a href=\"elimina.php\">Elimina cookie</a>";
} else {
    echo 'E' la tua prima visita?';
}
?>
</body></html>
```

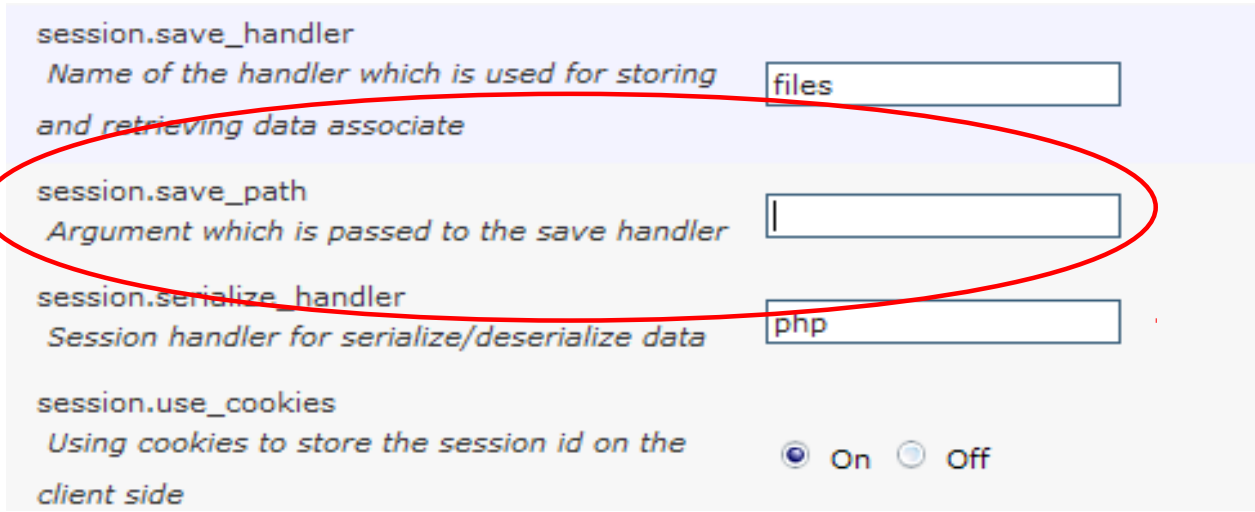
Sessioni: uso dal PHP

- Le sessioni sono state aggiunte in PHP 4.0.0
- Le sessioni forniscono lo strumento necessario per preservare i dati fra i vari accessi
- I dati della sessione sono salvati sul server e non sul client come nel caso dei cookie
- Ogni sessione è identificata da un **id** di sessione univoco
- L'id di sessione può essere memorizzato in un cookie nel client

La sessione permette di preservare un numero arbitrario di variabili sempre in accordo alle direttive definite nel file php.ini

Sessioni: uso dal PHP

- nella configurazione del PHP (il php.ini) va indicato nella direttiva session.save_path dove memorizzare sul server i dati di sessioni



Sessioni: uso dal PHP

- L'obiettivo è quello di mantenere i dati nel corso delle varie richieste che un client può effettuare

- Fortunatamente il PHP rende questo meccanismo molto semplice

```php
<?php
session_start();
$_SESSION['location'] = 'New York';
```

- Ogni script che vuole accedere ai dati in sessione deve richiamare la funzione session_start()
  nota: session_start() va eseguita su tutti gli script utilizzati

- L'array superglobale $_SESSION contiene tutti i dati di sessione che abbiamo deciso di salvare

Sessioni: uso dal PHP

- è possibile comunicare l'identificativo della sessione anche attraverso l'url
  session_start() va eseguita all'inizio dello script

- per accedere alle variabili di sessione si utilizza la variabile superglobal $_SESSION (è un array associativo)

```php
<?php
session_start();
$val=$_SESSION['count']++;
echo "salve, stai visitando questa pagina $val volte.";
```

Salve, stai visitando questa pagine 7 volte

Sessioni: uso dal PHP

- Le principali funzioni per utilizzare le sessioni sono

```
session_start();

session_destroy();
```

- session_start() deve essere chiamata all'inizio di ogni pagina PHP e permette la creazione di una nuova sessione o il recupero di una sessione creata in precedenza

- session_destroy() distrugge tutti i dati associati alla sessione corrente

nota: non viene cancellato il cookie nel browser dell'utente, viene cancellato all'uscita del browser

Sessioni: uso dal PHP

- La funzione **session_id()** restituisce l'id della sessione corrente

- E' possibile anche cambiare l'id della sessione richiamando la funzione **session_id()** e passandogli come parametro l'id che la sessione dovrà avere

Sessioni: uso dal PHP

- La funzione session_name() restituisce il nome della sessione corrente
- E’ possibile anche cambiare il nome della sessione richiamando la funzione session_name() e passandogli come parametro il nome che la sessione dovrà avere
- Il nome della sessione può contenere solo caratteri alfanumerici e deve contenere almeno un carattere alfabetico
- La funzione session_name() deve essere richiamata prima della funzione session_start()

Sessioni: uso dal PHP

- La funzione session_regenerate_id() cambia l'id della sessione corrente
- Viene spesso utilizzata per risolvere problemi di sicurezza riguardanti trasmissioni di dati attraverso il protocollo HTTP

```php
<?php
session_start (); // Apro la sessione
$old_sessionid = session_id (); // Prendo l'id della sessione
session_regenerate_id (); // Rigenero l'id della sessione
$new_sessionid = session_id (); // Prendo il nuovo id di sessione
echo "Vecchio ID di sessione: $old_sessionid<br />";
echo "Nuovo ID di sessione: $new_sessionid<br />";
print_r ( $_SESSION );// I dati della sessione non verranno modificati
```

Esercizio 13:

- crea due script PHP separati, start.php e continue.php
- in start.php va creata una variabile di sessione location che contiene la tua località.
  inoltre va creato un link a:
  `<a href="continue.php">Click Me!</a>`
- in continue.php, visualizzare l'output della località impostata in start.php

Esercizio 13: solution

- avvio della sessione e impostazione variabile
- Impostazione del link

```php
<?php
session_start();
$_SESSION['location'] = 'New York';
?>
<a href="session03_continue.php">Click Me!</a>
```

- visualizzazione del contenuto della variabile di sessione come array associativa

```php
<?php
session_start();
header('Content-Type: text/html; charset=utf-8');
echo "località ".$_SESSION['location'];
```

QUESTION TIME ?

Nome_____ Cognome_____ Data_____

ARRIVEDERCI

TITOLO