

UNIT 3

1.1 Le variabili superglobali `$_GET`, `$_POST`, `$_SESSION`, `$COOKIE`

PHP è un linguaggio dove in generale non sono supportate le cosiddette variabili globali, cioè variabili utilizzabili in qualsiasi ambito e punto del codice.

Ad ogni modo però in PHP, e nello specifico dal 4.1.x in poi sono state introdotte variabili interne di sistema che hanno comportamenti analoghi e sono cosiddette globali, anzi, **superglobali**.

In particolare si tratta di array (di cui tratteremo nello specifico più avanti) chiamati superglobali e predefiniti in PHP per l'utilizzo da parte dello sviluppatore.

Vediamo di seguito questi array superglobali:

`$_GET[]`

E' un array che include tutte le variabili ricevute da PHP tramite browser con il metodo GET di HTTP.

`$_POST[]`

E' un array che include tutte le variabili ricevute da PHP tramite browser con il metodo POST di HTTP.

`$_SERVER[]`

E' un array che include tutte le variabili relative al server web come hostname della macchina, remote address IP del navigatore etc...

`$_ENV[]`

E' un array che include tutte le variabili d'ambiente

`$_COOKIE[]`

E' un array che include tutte le variabili relative ai cookie che ha ricevuto dal web browser dell'utente

Tali variabili sono accessibili da qualunque punto di uno script PHP, di una funzione o di un metodo.

Alcuni esempi pratici:

<http://it.php.net/manual/it/tutorial.forms.php>

<http://it.php.net/manual/it/tutorial.oldcode.php>

1.2 Introduzione agli array in PHP

Dal momento in cui molti abbiamo già parlato di array è giusto iniziare a vedere una panoramica di cosa sono gli array in generale e come funzionano in PHP.

In PHP, così come in molti altri linguaggi di programmazione, con il termine array si identifica una struttura in cui i dati sono mantenuti tramite una associazione chiave --> valore.

Questa particolare struttura permette di gestire liste, tabelle hash, stack, code. Inoltre, il fatto che un valore possa esso stesso essere un array permette di gestire molte altre implementazioni di strutture dati ordinate che vengono utilizzate come basi dei più conosciuti algoritmi.

La creazione di un array avviene tramite il costrutto array e la definizione delle coppie di valori, si noti come l'indicazione delle chiavi (cioè degli indici del vettore) non è obbligatoria, si vedrà di seguito come PHP gestisce l'indicizzazione automatica dei valori per i quali non si è specificato l'indice.

```
$mioarray = array("chiave1" =>"valore1", 2 => "valore2", "chiave3" => "valore3", "valore4")
```

Utilissima, al solito, le funzioni `print_r` e `var_dump` che ci permettono di ispezionare il contenuto delle nostre strutture dati. Per il nostro esempio otterremo:

Array

```
(
  [chiave1] => valore1
  [2] => valore2
  [chiave3] => valore3
  [3] => valore4
)
```

Come si può vedere, l'interprete PHP si è occupato di assegnare l'indice 3 al valore4, il criterio utilizzato è quello di cercare il massimo intero utilizzato come indice e sommargli 1, nel caso non ci sia nessun indice numerico allora il primo sarà 0. Si presti attenzione al fatto che una chiave viene considerata un intero anche quando essa sia una rappresentazione standard dello stesso, quindi stringhe del tipo "3" verranno considerate indici numerici mentre non lo saranno le stringhe come "003".

I contenuti (valori indicizzati) dell'array sono accessibili indicando la chiave (indice) associata all'interno di parentesi quadre, sempre dal nostro esempio:

```
print $mioarray[2]; che stamperà valore2
```

oppure

```
print $mioarray["chiave1"]; che avrà come output valore1
```

si noti come

```
print $mioarray; non mostra tutti i valori ma l'indicazione che si sta cercando di stampare un array: array.
```

L'inserimento di nuovi valori all'interno dell'array può avvenire in modo esplicito cioè assegnando un valore ad una particolare chiave:

```
$mioarray["nuovachiave"]="nuovovalore";
```

oppure implicitamente delegando all'interprete la generazione dell'indice in base alle regole presentate sopra:
`$mioarray[]="nuovovalore";`

La rimozione di elementi di un array avviene tramite la funzione `unset`, ad esempio:
`unset($mioarray["nuovachiave"])`
rimuoverà dall'array l'elemento con indice `nuovachiave`.

Il fatto che un valore può essere a sua volta definito come un array offre la possibilità di utilizzare array multidimensionali, vediamo un esempio:

```
$multiarray= array("itemA" => array("campo1" => "valore1A",  
                                     "campo2" => "valore2A",  
                                     "campo3" => "valore3A"),  
                  "itemB" => array("campo1" => "valore1B",  
                                     "campo2" => "valore2B",  
                                     "campo3" => "valore3B")  
);
```

l'accesso alla particolare componente dell'array avviene, al solito, utilizzando gli indici all'interno di parentesi quadre, in questo caso scendendo tra i livelli fino a quello desiderato:

```
print $multiarray["itemB"]["campo2"];  
risulterà in valore2B
```

NOTA PER LA CERTIFICAZIONE ZEND

Sebbene nelle seguenti lezioni andremo ad approfondire il concetto di array e la sua manipolazione è tuttavia una delle parti più corpose e complesse da sapere per la certificazione Zend. In particolare la manipolazione degli array. Al fine di aiutarvi vi segnalo già la parte di manuale da studiare per prepararsi alla certificazione.

<http://it.php.net/manual/it/ref.array.php>

1.3 Approfondimento, variazione variabili di sistema da PHP > 4.1.0

Nelle versioni di php dopo la 4.1.0 sono state ridefinite le variabili d'ambiente che andranno a sostituire i vecchi array. Dato il loro largo e importante utilizzo le elenchiamo:

`$_SERVER` al posto di `$HTTP_SERVER_VARS`
`$_GET` al posto di `$HTTP_GET_VARS`
`$_POST` al posto di `$HTTP_POST_VARS`
`$_COOKIE` al posto di `$HTTP_COOKIE_VARS`
`$_FILES` al posto di `$HTTP_POST_FILES`
`$_ENV` al posto di `$HTTP_ENV_VARS`
`$_REQUEST` (non ha equivalenti nelle vecchie versioni)
`$_SESSION[]` per `$HTTP_SERVER_VARS[]`

Il funzionamento è molto simile ai vecchi array, quindi se in uno script fate riferimento alle variabili in input con gli appositi array non si deve far altro che cambiare i nomi. Decisamente diverso invece se per riferirsi alle variabili in input si usava il nome della variabile e non il suo array (es. `$var` invece di `$HTTP_GET_VARS[var]`).

Questo perchè nelle versioni dalla 4.2.0 in poi per default il `php.ini` contiene questa riga

```
register_globals = 'Off'
```

mentre in quelle precedenti la configurazione di default era:

```
register_globals = 'On'
```

Cambiando questa configurazione non vengono create in modo automatico le variabili di GET e POST ma sono presenti solo nelle rispettive variabili d'ambiente.

E' consigliabile mantenere la riga su Off, ma è sempre possibile impostarla su On (anche se ciò è deprecato per motivi di sicurezza).

Un'altra novità che può tornare utile è il fatto che queste variabili sono automaticamente globali, se in una funzione si inseriva:

```
global $HTTP_GET_VARS;
```

per rendere le variabili get visibili, ora tutto ciò non sarà necessario.

Un ultimo consiglio...

in questo momento di transizione è consigliabile scrivere gli script riferendosi alle variabili con i nuovi array ed inserire in cima allo script righe di questo tipo:

```
if(!isset($_GET)) $_GET = $HTTP_GET_VARS;
```

oppure

```
if(!isset($_POST)) $_GET = $HTTP_POST_VARS;
```

(ovviamente una per ogni array usato)

Usare questa soluzione può comportare alcuni problemi di sicurezza:

qualche utente malizioso potrebbe richiamare la pagina con `pagina.php?_GET['var']=53` e `$_GET` sarebbe settato.

Una soluzione proposta è questa:

```
if(!isset($_SERVER) OR !$_SERVER OR !is_array($_SERVER) OR  
count(array_diff($_SERVER, $HTTP_SERVER_VARS)))
```

```
$_GET = &$HTTP_GET_VARS;
```

```
$_POST = &$HTTP_POST_VARS;
```

```
$_SERVER = &$HTTP_SERVER_VARS;
```

```
$_ENV = &$HTTP_ENV_VARS;  
$_COOKIE = &$HTTP_COOKIE_VARS;  
$_FILES = &$HTTP_POST_FILES;  
$_SESSION = &$HTTP_SESSION_VARS;  
}
```

1.4 Hosting su piattaforma LAMP

Attualmente nonostante siano passati la bellezza di almeno 3 anni dall'introduzione di PHP 4.1.0 e dalla naturale sostituzione di `register_globals = Off` rispetto a `On` del passato quasi tutti gli hosting provider forniscono la loro piattaforma con configurazione di `register_globals` impostata ancora ad `On`.

Tale soluzione da un lato garantisce la funzionalità pressoché totale di tutte le applicazioni anche scritte in passato ma, prima di tutto, contravviene il monito di PHP secondo cui un domani tale configurazione sarà a `Off` di default, inoltre comporta il rischio che qualche sviluppatore "distratto" o "pigro" trovandosi tale configurazione programmi ancora in "vecchio stile".

Soluzioni possibili

Per ovviare al problema della retrocompatibilità con vecchie applicazioni senza però dimenticarsi di guardare avanti a mio parere vi è la seguente soluzione:

Impostare nella configurazione globale di `php.ini` `register_globals` a `Off`.

Per tutte le vecchie applicazioni che non funzionano correttamente impostare a livello di `virtualhost` o di `.htaccess` la conf più corretta.

In questo modo da un lato si è incentivati ad usare la nuova configurazione senza però aver problemi con le vecchie.

Maggiori informazioni sulla configurazione di PHP:

Dal manuale di PHP.NET

<http://it.php.net/manual/it/ini.php>

Link a infobox sulla conf di `php.ini` scritto da me tempo fa:

<http://openskills.info/infobox.php?ID=229>