Bibliografía:

Implantación de Aplicaciones Web, Jorge Sánchez Asenjo

PHP con acceso a bases de datos, Félix Mateos.

TEMA 5. ARRAYS EN PHP

Un array es una colección de datos a los que podemos referirnos con un mismo nombre de variable, y que se encuentran situados en posiciones correlativas de memoria.

Los datos no están obligados a ser todos del mismo tipo; por ejemplo, podemos mezclar datos numéricos con cadenas de caracteres.

Para indicar que una variable va a referirse a un array en lugar de a un dato único podemos hacerlo de dos maneras:

```
   $datos=array();
```

Ó

```
   $datos=[];
```

Para asignar valores a un array lo puedo hacer en la declaración, de dos maneras:

```
$datos=array(5,4,7);
Ó
$datos=[5,4,7]
```

En ambos casos se crearía un array de tres posiciones, estas posiciones empiezan en 0 e indican la posición de la celda dónde está el dato.

5	4	7
0	1	2

También puedo asignar valores posteriormente de forma individual mediante la utilización de la sintaxis de corchete. Las siguientes instrucciones me generarían el mismo array que en el caso anterior:

```
$datos=[];
$datos[0]=5;
$datos[1]=4;
$datos[2]=7;
```

En un array puedo introducir datos de distinto tipo:

```
$datos=['uno',5,'cuadro',2.3];
```

No podemos usar la instrucción `echo` para imprimir todos los elementos de un array, pero sí la instrucción `print_r`.

```
print_r($datos);
```

Sacaría por pantalla:

```
Array( 0=>uno 1=>5 2=>cuadro 3=>2,3 )
```

Podemos saber el número de elementos que tiene un array usando la función cont,

```
echo cont($datos); // mostraría 4, que es el número de elementos del array $datos
```

Una de las cualidades más interesantes de los arrays es que un elemento de un array puede ser a su vez otro array, de modo que podemos construir así matrices multidimensionales.

```
001   <?php
002      $juan=array('Juan Félix Mateos',185,90);
003      $ana=array('Ana Irene Palma',172,57);
004      $alumnos=array($juan,$ana);
005      print_r( $alumnos);
006   ?>
```

O alternativamente:

```
001   <?php
002      $alumnos=array(
003         array('Juan Félix Mateos',185,90),
004         array('Ana Irene Palma',172,57));
005      print_r( $alumnos);
006   ?>
```

En el caso de los arrays multidimensionales, para referirnos a un dato concreto tendremos que concatenar sus índices entre corchetes. Por ejemplo, para referirnos a la talla (segundo dato) del segundo alumno escribiríamos `$alumnos[1][1]`.

Otra cualidad interesante de los arrays de PHP es que los índices no están obligados a ser números; también pueden ser cadenas de caracteres. Por ejemplo:

```
<?php
   $pedro=array('edad'=>38,'peso'=>80,'pelo'=>'moreno');
   echo $pedro['edad']; // Mostraría 38
   echo '<br />';
   echo $pedro['pelo']; // Mostraría moreno
?>
```

En memoria almacenaría los datos así:

38	80	moreno
Edad	peso	pelo

Los arrays que utilizan como índices cadenas de caracteres se denominan asociativos.

Recorrer un array

For

El bucle más usado para recorrer un array es el for, aunque también se puede usar el while o el do while.

Ejemplo de código de un programa que calcula la nota media de un array de notas:

```
<?php
      $notas=array(9,3,5,6,4,9,4,2,1);
      $suma=0;
```

```php
        for($i=0;$i<count($notas);$i++){
            echo "nota ".($i+1).":".$notas[$i]."<br>";
        }

        for($i=0;$i<count($notas);$i++){
            $suma=$suma+$notas[$i];
        }

        $media=$suma/count($notas);
        echo 'La nota media es:'.$media;

        print_r($notas);
        ?>
```

foreach

Es un tipo específico de bucle que se utiliza para recorrer los elementos de un array (u objeto; sólo a partir de PHP 5), bien tanto sus índices como sus datos, como bien sólo sus datos. Se utiliza sobre todo en arrays asociativos.

Su sintaxis general para recorrer índices y datos es la siguiente:

```
foreach (array as variable_para_indice => variable_para_dato){
      instrucciones_de_la_iteración
}
```

Su sintaxis general para recorrer sólo los datos es la siguiente:

```
foreach (array as variable_para_dato){
      instrucciones_de_la_iteración
}
```

Ejemplo de programa que muestra una lista ordenada de producto-precio por pantalla a partir de un array asociativo usando foreach.

```php
<?php

$productos=array("lapiz"=>1.2,"cuaderno"=>2,"boligrafo"=>1.5,"sacapuntas"
=>0.85);
        echo "<ol>";
        foreach($productos as $nom=>$p){
            echo "<li>".$nom.":".$p."</li>";
        }
        echo "</ol>";
        ?>
```

Variables superglobales

Son variables predefinidas (es decir, su valor está definido por el propio intérprete PHP y no podemos crearlas nosotros mismos) que se reconocen automáticamente en cualquier posición del código (dentro y fuera de funciones). Todas las variables superglobales son arrays asociativos. Las variables superglobales son (no hay más que) :

- $_GET: Contiene los datos recibidos por el script a través de parámetros URL. Por ejemplo, si solicitamos el script letra_dni.php?numero=50453277, el script letra_dni dispondrá del dato 50453277 en la variable $_GET['numero'].

  Ejemplo:

  ```
  <?php
  $resultado=$_GET['numero'];
  $resultado=2*$resultado;
  echo 'El doble del numero recibido:'.$resultado;
  ?>
  ```

  Si se envían varios valores estos se separan por &: parametro2.php?numero=27&otro=3

- $_POST: Contiene los datos recibidos por el script a través del método HTTP POST. Por ejemplo, si un formulario cuyo action es el script letra_dni.php contiene un campo cuyo name es numero, el script letra_dni.php dispondrá del dato escrito en el campo numero del formulario en la variable $_POST['numero']. (Lo vemos más adelante)
- $_FILES: Contiene los archivos enviados al script a través de elementos `input type='file'` de un formulario.
- $_SESSION: Contiene las variables de sesión. A diferencia de las cookies, las variables de sesión no se envían al cliente.
- $_COOKIE: Contiene los datos recibidos del cliente a través de cookies. Esos datos deben haber sido previamente almacenados en la cookie del cliente por el mismo servidor (las cookies sólo pueden ser leídas por el propio servidor que las creó).
- $_REQUEST: Contiene todos los datos recibidos mediante los métodos Get, Post y Cookies. Si el script recibe un dato con el mismo nombre a través de varios de estos métodos, prevalecerá el del que ocupe la última posición (más a la derecha) en la directiva `request_order` del archivo php.ini. Por ejemplo, en WAMP se utiliza la directiva `request_order = "GP"`, que da prioridad a los datos recibidos a través del método POST y

no incluye en $_REQUEST los datos recibidos a través de cookies (por motivos de seguridad).

- $_SERVER: Este array contiene datos que rellena el propio servidor web en base a la RFC 3875 (http://www.faqs.org/rfcs/rfc3875.html). No podemos tener la certeza de que el servidor defina todos los datos que recoge esta norma, pues puede estar configurado para definir sólo algunos de ellos. Algunos de los datos interesantes que podemos recoger a través de esta variable son HTTP_REFERER (indica desde qué página ha solicitado el cliente el script actual), HTTP_USER_AGENT (indica qué navegador ha usado el cliente para solicitar el script), REMOTE_ADDR (indica la dirección IP desde la que se ha recibido la petición), PHP_AUTH_USER (contiene el nombre usado por el usuario como acreditación para acceder al script si está protegido por el sistema de autentificación HTTP de Apache).

Nota: Tenga en cuenta que HTTP_REFERER, HTTP_USER_AGENT, PHP_AUTH_USER, … son constantes de PHP que representan a cadenas de texto, de modo que aunque $_SERVER sea un array asociativo no debe escribir estas constantes entre comillas. Por ejemplo $_SERVER[' PHP_AUTH_USER'] no es válido; la sintaxis correcta es simplemente $_SERVER[PHP_AUTH_USER] (sin comillas).

- $_ENV: Contiene datos establecidos por el sistema operativo. Al igual que ocurre con $_SERVER no todos los sistemas establecen los mismos datos. Por defecto esta variable está vacía en WAMP porque en la directiva `variables_order` del archivo php.ini no se incluye la letra E de Environment.
- $GLOBALS (sin guión bajo). Contiene las variables globales, de modo que podemos referirnos a ellas desde el interior de cualquier función. Por ejemplo, si tenemos una variable global llamada $usuario y queremos referirnos a ella desde el interior de una función del mismo script podremos hacerlo con $GLOBALS['usuario']. Observe que se ha escrito el nombre de la variable sin el $.