PHP

UT5: ARRAYS

1. Introducción
2. Arrays numéricos o escalares
3. Arrays asociativos
4. Ordenar un array
5. Arrays multidimensionales
6. Funciones para manejar arrays

www.php.net/manual/es/

1. Introducción

Un **array** es una colección de datos a los que podemos referirnos con un mismo nombre de variable.

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 debemos usar la función `array`.

Sintaxis:

array ([clave o índice=>] valor, ...)

Un ***array*** se caracteriza por tres cosas:

- Está formado por un conjunto de elementos.
- Tiene un **índice** (*key* o *index*) para referirse a cada uno de sus elementos. Este índice, también llamado subíndice, puede ser de tipo numérico entero o de tipo cadena.
- Cada elemento tiene un **contenido** (*value*), que puede ser de diferente tipo que el resto de los elementos.

Según el tipo de índice los arrays en PHP pueden ser:

- Arrays numéricos o escalares: los índices son números
- Arrays asociativos: los índices son cadenas.

2. Arrays numéricos o escalares

Sintaxis:

array ([clave o índice=>] valor, ...)

El índice del primer elemento tiene el valor 0.

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

Pueden definirse:

1. Con la función `array` sin definir posiciones o índices:

```
$ciudad=array("Madrid","Paris","Londres");
print_r($ciudad);
```

2. Con referencia:

```
$ciudad=array();
$ciudad[0]="Madrid";
$ciudad[1]="Paris";
$ciudad[2]="Londres";
print_r($ciudad);

/* O también   */
$ciudad=array();
$ciudad[]="Madrid";
$ciudad[]="Paris";
$ciudad[]="Londres";
print_r($ciudad);
```

3. Con la función array definiendo posiciones o índices:

```
$ciudad=array(0=> "Madrid", 1=> "Paris", 2=> "Londres");
print_r($ciudad);
```

Mostrar información del array:

1. Mostrar el contenido del array con el bucle for:

```
echo 'Mostrar el contenido del array con el bucle for <br/>';
$ciudad=array("Madrid","Paris","Londres");
$numero_elementos=count($ciudad);
for($i=0;$i<$numero_elementos;$i++):
        echo "    $ciudad[$i]";
endfor;
```

2. Mostrar la posición o índice y el contenido del array con el bucle for:

```
echo '<br><br/>Mostrar la posición o índice y el contenido del array con el bucle for <br/>';
$ciudad=array("Madrid","Paris","Londres");
$numero_elementos=count($ciudad);
for($i=0;$i<$numero_elementos;$i++):
        echo   "    Posición o índice = $i    
                Contenido: $ciudad[$i]<br/> ";
endfor;
```

3. Mostrar el contenido del array con el bucle foreach:

```
echo '<br><br/>Mostrar el contenido del array con el bucle foreach <br/>';
$ciudad=array("Madrid","Paris","Londres");
foreach ($ciudad as $nombre_cada_ciudad):
        echo "    $nombre_cada_ciudad";
endforeach;
```

4. Motrar la posición o índice y el contenido del array con el bucle foreach:

```
echo '<br><br/>Mostrar la posición o índice y el contenido del array con el bucle foreach <br/>';
        $ciudad=array("Madrid","Paris","Londres");
        foreach ($ciudad as $i=>$nombre_cada_ciudad):
                echo   "    Posición o índice = $i    
                        Contenido: $ciudad[$i]<br/> ";
        endforeach;
```

3. Arrays asociativos

La clave o índice es un String.

Pueden definirse:

1. Mediante la función array():

```
$persona = array("nombre" => "David",  "apellidos" => "Casas Ramos", "edad" => 24);
print_r($persona);
```

2. Con referencia:

```
$persona["nombre"] = "David";
$persona["apellidos"] = "Casas Ramos";
$persona["edad"] = 24;
print_r($persona);
```

Mostrar información del array:

1. Mostrar el contenido del array con el bucle foreach:

```
foreach ($precios as $indice=>$valor):
    echo "<br/> Producto = $indice     Precio = $valor";
endforeach;
```

Los arrays que utilizan como índices cadenas de caracteres suelen denominarse asociativos, pero en realidad no son una entidad diferente de los arrays que utilizan números como índices (que suelen denominarse arrays numéricos); lo que ocurre es que cuando asignamos un dato a un array sin especificar un índice, PHP por defecto le asigna un índice entero (el siguiente disponible dentro del array, empezando por 0). Tanto es así que no son una entidad diferente, que no hay ninguna limitación para combinar en un mismo array índices numéricos y de cadena. Por ejemplo:

```
001    <?php
002            $cronos_vueltas_fernando_alonso=array('circuito'=>'Montmel&oacute',
003            2.20,
004            5=>2.21,
005            3=>2.22,
006            2.23,
007            'fecha'=>'1-julio-2011');
008            print_r($cronos_vueltas_fernando_alonso);
009    ?>
```

Obsérvese en este ejemplo que el tiempo 2.23 se asigna al índice 6 porque es el entero inmediatamente posterior al mayor de los utilizados ya en el array (en otras palabras: pueden quedar índices numéricos vacíos dentro de un array).

Asimismo, no existe ningún tipo de vinculación entre los datos asignados a índices de cadena y la posición que ocupan en el array. Por ejemplo, en el código siguiente si intentamos referirnos a `$mes[1]` no obtendremos febrero, sino un error por hacer uso de un índice no definido en el array.

```
001    <?php
002            $mes=array(0=>'enero','uno'=>'febrero',2=>'marzo');
003            echo $mes[1];
004    ?>
```

4. Ordenar un array

asort($array);

Ordena un array por sus valores en orden ascendente (A-Z) pero manteniendo las asociaciones índice-valor.

arsort($array);

Idem anterior pero en orden descendente.

ksort($array);

Ordena un array por sus "índices" manteniendo las asociaciones índice-valor. De A-Z.

krsort($array);

Idem anterior pero en orden descendiente. De Z-A.

sort($array);

Ordena un array por sus valores en orden ascendente. De A-Z.

rsort($array);

Igual que sort pero en orden descendente. De Z-A.

```
<?php
        echo "<pre>";
        $pila=array("cinco" =>5,"uno" =>1, "cuatro" =>4,"dos" =>2 , "tres"=>3);
        echo "<b>Array sin ordenar</b><br />" ;
        print_r($pila);
        echo "<b>Ordenación asort()</b><br />" ;
        asort($pila);
        print_r($pila);
        echo "<b>Ordenación arsort{)</b><br />" ;
        arsort($pila);
        print_r($pila);
        echo "<b>Ordenación ksort()</b><br />";
        ksort($pila);
        print_r($pila);
        echo "<b>Ordenación krsort()</b><br />";
        krsort ($pila);
        print_r($pila);
        echo "<b>Ordenación sort()</b><br />";
        sort($pila);
        print_r($pila);
        echo "<b>Ordenación rsort()</b><br />";
        rsort($pila);
        print_r($pila);
        echo "</pre>";
?>
```

5. Arrays multidimensionales

Son arrays en los que al menos uno de sus valores es, a su vez, otro array.

Pueden ser escalares o asociativos.

Pueden definirse:

1. Mediante la función array():

```
$notas = array(
                array("nombre"=>"David", "iaweb"=>7,"asgbd"=>8),
                array("nombre"=>"Rosa", "iaweb"=>5,"asgbd"=>4),
                array("nombre"=>"Pablo", "iaweb"=>3,"asgbd"=>6)          );

/* O también   */
$notas = array();
$notas[]=array("nombre"=>"David", "iaweb"=>7,"asgbd"=>8);
$notas[]=array("nombre"=>"Rosa", "iaweb"=>5,"asgbd"=>4);
$notas[]=array("nombre"=>"Pablo", "iaweb"=>3,"asgbd"=>6);
```

2. Con referencia:

```
$notas[0]["nombre"]="David";
$notas[0]["iaweb"]=7;
$notas[0]["asgbd"]=8;
$notas[1]["nombre"]="Rosa";
$notas[1]["iaweb"]=5;
$notas[1]["asgbd"]=4;
$notas[2]["nombre"]="Pablo";
$notas[2]["iaweb"]=3;
$notas[2]["asgbd"]=6;
```

Mostrar información del array:

1. Mostrar el contenido del array con el bucle foreach:

```
echo "<u>Nombre     IAWEB     ASGBD <br/></u>";
foreach ($notas as $i=>$alumnos):
        foreach($alumnos as $j=>$alumno):
                        echo "$alumno        ";
        endforeach;
                echo "<br/>";
         endforeach;
```

6. Funciones para manejar arrays

count($array);
Devuelve el número de elementos del array.

min($array);
Devuelve el elemento menor del array.

max($array);
Devuelve el elemento mayor del array.

array_reverse($array);
Devuelve otro array que contiene los mismos elementos ordenados al revés. Hace lo mismo que la función rsort();

in_array(valor buscado, $array);
Devuelve *True* si valor buscado está contenido en alguno de los elementos del array y False si no lo está,