Usuario:ManuelRomero/php/dwes/B2T1/arrays/concepto

De WikiEducator
< Usuario:ManuelRomero

cpifp Los Enlaces

TEMA 5: LOS ARRAYS EN PHP

LENGUAJE PHP: **TRABAJANDO CON ESTRUCTURAS DE INFORMACIÓN : ARRAYS**

¡El servidor te responde

PHP Los Arrays: estructuras de datos fundamentales en php

Contenido

Show presentation

Arrays

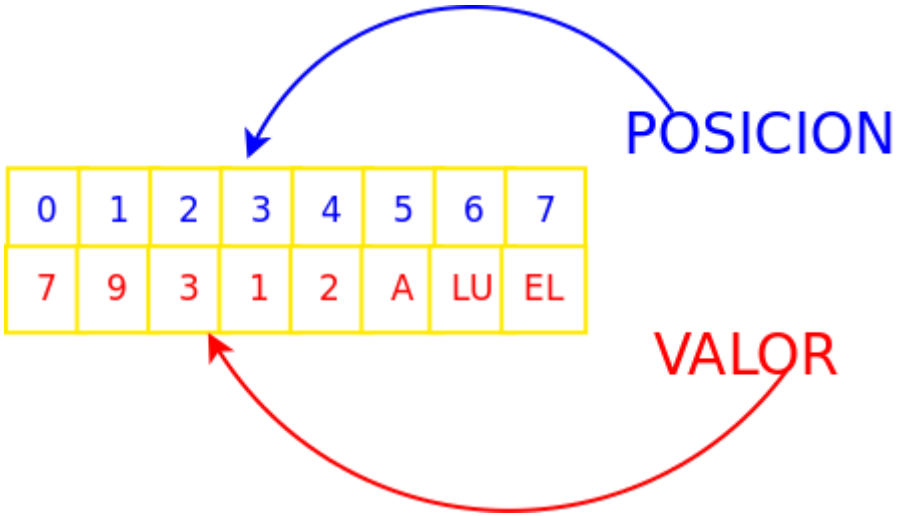
- Un tipo de datos compuesto es aquel que está formado por varios valores que se pueden tratar de manera independiente, pero a la vez se maneja de forma única.

- En PHP puedes utilizar dos tipos de datos compuestos: ***el array*** y ***el objeto***.
- Los objetos los veremos más adelante. Además de los ***objetos***, y muy relacionados están los ***recursos***, que también veremos más adelante.

Definición

Un array es un tipo de datos compuesto que nos permite almacenar varios valores en una única variable.

- Cada miembro del array se almacena en una posición a la que se hace referencia utilizando un **valor clave o índice**.



- Dependiendo del valor de la posición podemos clasificar los arrays de dos tipos:

1. ***Indexado***. Cada valor es un entero que indica su posición, empezando por cero.
2. ***Asociativo***. El valor de cada posición tiene un significado diferente a la posición que ocupa y puede ser de cualquier tipo.

```
// array numérico
$modulos1 = array(0 => "Programación",
                  1 => "Bases de datos", ...,
```

```
                  9 => "Desarrollo web en entorno servidor");

// array asociativo
$modulos2 = array("PR" => "Programación",
                  "BD" => "Bases de datos", ...,
                  "DWES" => "Desarrollo web en entorno servidor");
```

```
$modulos1 = [0 => "Programación",
                  1 => "Bases de datos", ...,
                  9 => "Desarrollo web en entorno servidor"];
// array asociativo
$modulos2 = ["PR" => "Programación",
                  "BD" => "Bases de datos", ...,
                  "DWES" => "Desarrollo web en entorno servidor"];
```

Arrays asociativos Vs indexados

Podemos pasar de array indexado a asociativo creando un nuevo índice en cualquier momento. En realidad los arrays indexados son un caso especial de array asociativo, en dónde el valor de la clave la genera de forma automátia el sistema. En breve expondremos esta idea que vemos en el ejemplo

Ejemplo de asignación

- Aunque estos conceptos los analizaremos mas tarde mira el siguiente código

```
<?php
//Creo e inicializo un array llamado modulos
```

```php
$modulos = array("PR" => "Programación",
                 "BD" => "Bases de datos",
                 "DWES" => "Desarrollo web en entorno servidor");

//Agrego un nuevo elemento sin especificar posición
$modulos[] = "otro módulo";

//Ahora añado un módulo en el índice con valor  10  que NO ES LA POSICIÓN 10
$modulos[10] = "Módulo indexado con el índice 10";


//incorporamos un nuevo elmento en el array sin especificar el valor del índice
//PHP asignará un valor siguiente el último valor numérico que fue el 10 (indice último numérico agragado)
$modulos[] = "Último módulo añadido";

//Visuzlizo con var_dump el contenido y estructura de la variable $modulo
var_dump($modulos);
```

- La salida que genera ver el contenido de la variable $modulos es un array de 6 elementos
- Y vemos el valor de los índices y los contenidos de cada elemento:

```
array (size=6)
  'PR' => string 'Programación' (length=13)
  'BD' => string 'Bases de datos' (length=14)
  'DWES' => string 'Desarrollo web en entorno servidor' (length=34)
  0 => string 'otro módulo' (length=12)
  10 => string 'Módulo indexado con el índice 10' (length=34)
  11 => string 'Último módulo añadido' (length=24)
```

Array numéricos o indexados

- Clasificación en función de la clave que llevan:

1. Clave numérica posicional : La clave especifica la posición de cada elemento.
2. Clave con valor: La clave tiene un significado por sí mismo y tiene **asociado** un valor.

Arrays indexados

En los arrays numéricos empezamos por 0.

- El contenido de un elemento de un array puede ser cualquier valor, incluido otro array, puediendo fácilmente construir estructuras mas complejas o arrays N-dimensionales, como veremos en ejemplos posteriores

Trabajar con un array

- A la hora de trabajar con arrays, hay que saber operar con ellos:

1. ***Crear*** o definir un array.
2. ***Asignar***, agregar valores al array.
3. ***Leer*** elementos del array.
4. ***Borrar*** elementos del array.

Crear un array

- Podemos usar el operador ***array();***
- A partir de la versión 5.3 se puede directamente usar el operador ***[]***
- En php no hay que especificar ni índice ni lógicamente tipo.

El ***array***, como hemos comentado, va a ser un conjunto de elementos, cada tipo de cada elemento dependerá del valor que contenga en cada momento , y por supuesto puede ser modificado (tanto el valor como el tipo).

Crear un array

```
/*
Creamos una variable '''array''' llamada ''miArray'' vacía.
Dos formas equivalentes:
*/
$miArray = array();
$miArray = [];
/*
Creamos un array indexado de ciudades
*/
$miArray = array("Burgos","Zaragoza","Huesca", "Teruel","Soria");
$miArray = ["Burgos","Zaragoza","Huesca", "Teruel","Soria"];
```

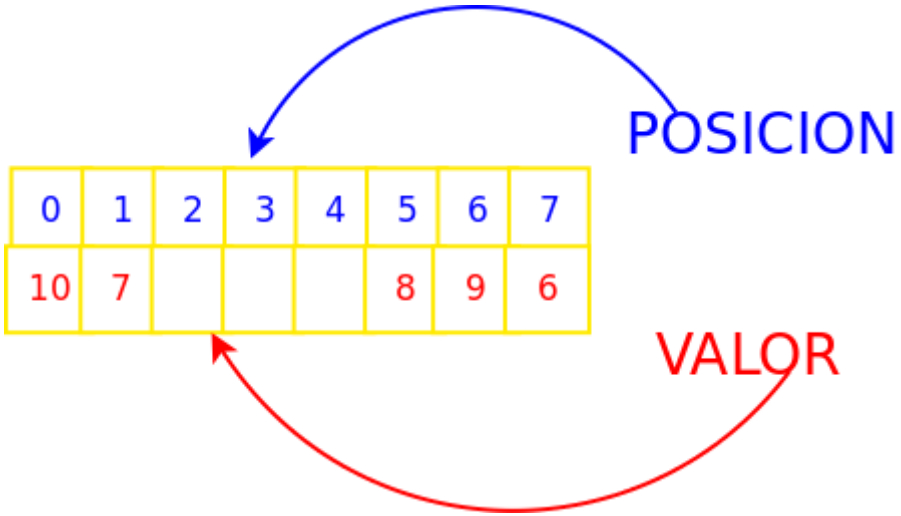
Escribir en un array

- Simplemente hay que asignar un valor a una posición del ***array***
- Si no especificamos valor al índice, PHP asignará un valor numérico superior al valor numérico más alto asignado
- Si pongo un valor superior al número de índices, el siguiente elemento estará en una posición mas

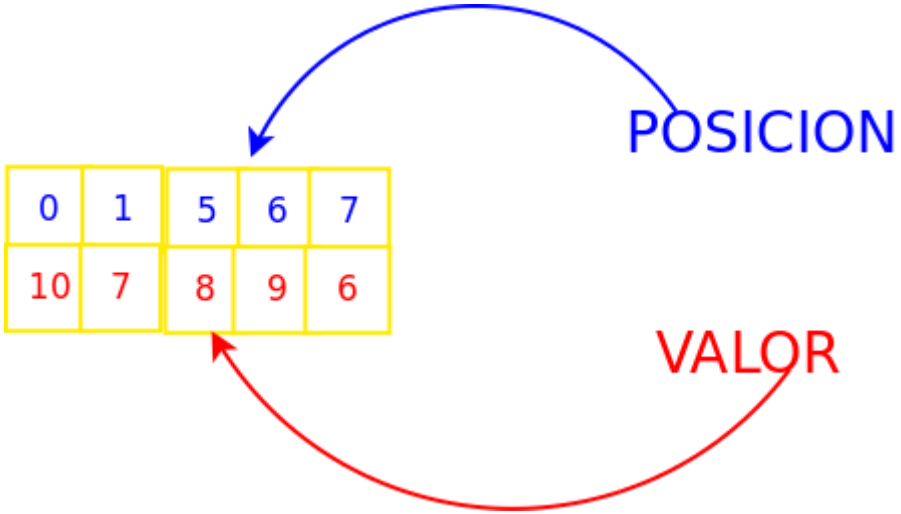
```php
$notas =[];
$notas[]=10;
$notas[]=7;
$notas[5]= 8;
$notas [] =9;
$notas [] =6;
```

Escribir en un array

- Podemos ver gráficamente como queda el **array**.



- Las posiciones no especificadas no existen.



- Vemos como la posición 2,3 y 4 no van a existir con valores en el array.
- Las puedo usar explícitamente.

```
$notas[2]= 8;

</div>
;Escribir en un array
Podemos ver cómo quedaría el array si realizamos las siguientes modificaciones:
<source lang=php>
$notas =[];
$notas[]=10;
$notas[20]=7;
$notas[5]= 8;
$notas [1] =9;
$notas [2] =6;
$notas [] =10;

var_dump($notas)
```

- Nos monstrará la siguiente información, donde vemos que la última posición añadida en el índice el sistema generó el valor 21.
- También podemos observar que el valor del índice no tiene nada que ver con la posición ordenada del elemento dentro de la variable (Tiene 6 posiciones, desde la 0 hasta la 5).

```
array (size=6)
  0 => int 10
  20 => int 7
  5 => int 8
```

```
  1 => int 9
  2 => int 6
  21 => int 10
```

</source>

- Un array se puede inicializar de los siguientes modos:

```
$capitales = array("España"=>"Madrid",
                   "Italia"=>"Roma",
                   "Alemania"=>"Berlín");
$capital ["España"=>"Madrid",
          "Italia"=>"Roma",
          "Alemania"=>"Berlín"];
```

- También se puede crear directamente con []

```
$capitales["España"]="Madrid";
$capitales["Italia"]="Roma"
$capitales["Alemania"]="Berlín";
```

Leer un array

- Hemos de diferenciar entre dos conceptos:
  - Leer un elemento de un **array**
  - Recorrer un array

Leer un elemento de un array

Simplemente accedemos a su posición por el índice

```
for ($n=0; $n<8; $n++)
    $notas[]= rand(1,10);

echo "la nota primera es $notas[0]";
echo "la nota de la posición 7 es $nota[7]";
echo "El valor de la variable $notas es -$nota-
```

- El código anterior saldría,

```
la nota primera es 6
la nota de la posición 7 es 8
El valor de la variable $notas es -Array-
```

Vemos como al escribir el nombre del array no veríamos su contenido si no el tipo del valor.

Recorrer un array

- Recorrer un array es pasar por todos sus valores, desde la primera posición hasta la última.
- En otros lenguajes la forma normal de hacer esto es obtener el número de elementos y visitarlo con un bucle for.
- En php tenemos la función ***count()*** o ***sizeof()*** que nos devuelve en número de elementos del array
- Vemos en el ejemplo cómo hacerlo

```php
<?php

//Inicializo un array de tamaño entre 20 y 30 elementos
$tamano = rand(10,15);
for ($n = 0; $n<$tamano ; $n++){
    $notas[]=rand(0,10);
}
//Ahora para recorrerlo obtengo el tamaño del array
$elementos = sizeof($notas);

echo "<h2>Vamos a recorrer un array de $elementos elementos</h2>";
//Reocorro cada elemento
for ($n=0; $n<$elementos; $n++){
    echo "Valor de la posición $n <b>$notas[$n]</b><br />";
}
?>
```

```
Vamos a recorrer un array de 12 elementos

Valor de la posición 0 8
Valor de la posición 1 6
Valor de la posición 2 6
Valor de la posición 3 10
Valor de la posición 4 9
Valor de la posición 5 3
Valor de la posición 6 6
Valor de la posición 7 7
Valor de la posición 8 2
Valor de la posición 9 6
Valor de la posición 10 4
Valor de la posición 11 10
```

```php
$ciudades = ["Burgos","Zaragoza","Huesca", "Teruel","Soria"];
$numeroCiudadades = count($ciudades);
echo "El array tienen $numeroCiudades ciudades<br/>";
//mostrará 5 ciudades
```

Recorrer un array

- Vamos a usarla con un ejemplo

Actividad

- Creamos 10 notas aleatorias, y posteriormente las visualizamos
- Sacamos la máxima la mínima y la media

```php
<?php
    //Creo la variable array de notas
    $notas=[];
    //Relleno el array con 10 notas
    //$notas = array_fill(0, 10,rand(1,10));
    for ($a=0;$a<10;$a++)
      $notas[$a]=rand(1,10);
    $min = $notas[0];
    $max = $notas[0];
    for ($a=0;$a<10;$a++){
        echo "Valor de la posición $a = ". $notas[$a]."<br />";
        $total+=$notas[$a];
        $min = $notas[$a]<$min ? $notas[$a] : $min;
        $max = $notas[$a]>$max ? $notas[$a] : $max;
    }
    echo "Valor de la nota media ".($total/10)."<br />";
    echo "Valor mínimo".$min."<br />";
    echo "Valor máximo ".$max."<br />";
```

Recorrer un array

Pregunta

¿Por qué no conviene usar la función ***count***

- Esta no es una forma conveniente de recorrer un array en php, pues no tiene porqué coincidir en el índice de cada elemento con su posición numérica.

- Supongamos el siguiente caso

```
$capitales = array();//Creo un array vacío. No hace falta hacerlo antes de dar valores
$capitales= ["España"=>"Madrid", "Italia"=>"Roma","Alemania"=>"Berlín"];
```

Recorrer un array

- La forma de recorrer un array en php es con la estructura de control ***foreach***

```
foreach ($array as $valor){
 ....
}
```

- La forma de leerla sería

Para cada elemento de $array, cuyo valor guardo en $valor
Esto lo hago hasta que llegue al último elemento

- O bien:

```
foreach ($array as $valor => $indice){
 ....
}
```

- Cuya lectura sería

Para cada elemento de $array, cuyo valor índice (valor del índice guardo en $indice y cuyo valor almaceno en $valor
Esto lo hago hasta que llegue al último elemento

- Veamos un ejemplo y la salida que produce

```php
<?php

$capitales =["España"=>"Madrid", "Italia"=>"Roma","Alemania"=>"Berlín"];

$n=0; //Para llevar un control del número de elementos

//Accediendo solo a los contendios
echo "<h2>Vamos a ver las capitales del array </h2>";
//$capital es el nombre de una variable que ponemos de forma libre (podemos elegir el nombre que queramos)
foreach ($capitales as $capital){
    echo "La capital de la posición <b>$n</b> es <b>$capital</b><br />";
    $n++;
}

//Si quiero ver también los índices
echo "<h2>Vamos a ver los paises con sus capitales del array </h2>";
//Tanto $pais como $capital son nombres de variables que ponemos de forma libre (podemos elegir los nombres que queramos)
$n=0;
foreach ($capitales as $pais =>$capital){
    echo "La capital de la posición <b>$n</b> del país <b>$pais </b> es <b>$capital</b><br />";
    $n++;
}

?>
```

- La salida generada será

```
Vamos a ver las capitales del array

La capital de la posición 0 es Madrid
La capital de la posición 1 es Roma
La capital de la posición 2 es Berlín
Vamos a ver los paises con sus capitales del array

La capital de la posición 0 del país España es Madrid
La capital de la posición 1 del país Italia es Roma
La capital de la posición 2 del país Alemania es Berlín
```

Modificar un array recorrido

- Supongamos que queremos modificar el contenido de un array que estamos recorriendo.
- Por ejemplo supongamos que tenemos la siguiente estructura de datos

Un array de productos de cada uno de los cuales vamos a tener nombre y precio

```php
<?php
 $productos = [
        'lechugas'=>  ['precio' => 100, 'unidades'=>50],
        'manzanas'=>  [ 'precio' => 200, 'unidades'=>100],
        'peras'=>  [ 'precio' => 300, 'unidades'=>150],
        'tomates'=>  [ 'precio' => 400, 'unidades'=>200],
        'cebollas'=>  ['precio' => 500, 'unidades'=>25],
    ];

echo "<h2>Visualizamos los productos</h2>";

//Para cada producto
foreach ($productos as $producto=>$datos){
    $precio = $datos['precio'];
    $unidades = $datos['unidades'];
    echo "<h3>producto $producto precio $precio unidades $unidades</h3>";
}
?>
```

- Si quisiéramos modificar el precio un 10% e incrementar 100 cada producto podríamos pensar en hacer en el bucle, pero vemos que el precio no se modificaría

```php
echo "<h2>Modificamos el precio  (10%) y las unidades en 100 unidades</h2>";
//Para cada producto
foreach ($productos as $producto=>$datos){
     $datos['precio'] *=1.10;
     $datos['unidades'] +=100;
}
echo "<hr />";

//Vovemos a visualizar y vemos que no ha cambiado
//Para cada producto

echo "<h2>Visualizamos los productos previamente modificados</h2>";
foreach ($productos as $producto=>$datos){
    $precio = $datos['precio'];
    $unidades = $datos['unidades'];
    echo "<h3>producto $producto precio $precio unidades $unidades</h3>";
}
```

- Esto es por que en realidad en cada interación se está cogiendo el valor del elemento del array ***$productos*** y se copia en la variable ***$prodcuto*** el índice y ***$datos*** el valor, pero $datos es otra posición de memoria diferente que el contenido correspondiente del array, por lo que al modificarlo el array se queda inalterado.

Si queremos modificarlo, tendríamos que hacer que el valor de ***$datos*** estuviera en la misma posoción de memoria que la posición correspondiente del array, y esto se consigue con el operador **&**, por lo que el bucle sería

```php
foreach ($productos as $producto=>&$datos){
```

en lugar de

```
foreach ($productos as $producto=>$datos){
```

El código completo que sí que modifica el array sería

```
<?php
 $productos = [
        'lechugas'=>  ['precio' => 100, 'unidades'=>50],
        'manzanas'=>  [ 'precio' => 200, 'unidades'=>100],
        'peras'=>  [ 'precio' => 300, 'unidades'=>150],
        'tomates'=>  [ 'precio' => 400, 'unidades'=>200],
        'cebollas'=>  ['precio' => 500, 'unidades'=>25],
    ];

echo "<h2>Visualizamos los productos</h2>";

//Para cada producto
foreach ($productos as $producto=>$datos){
    $precio = $datos['precio'];
    $unidades = $datos['unidades'];
    echo "<h3>producto $producto precio $precio unidades $unidades</h3>";
}
//Ahora modificamos el precio un 10% y e incrementamos 100 unidades cada producto

echo "<h2>Modificamos el precio  (10%) y las unidades en 100 unidades</h2>";
//Para cada producto
foreach ($productos as $producto=>$datos){
     $datos['precio'] *=1.10;
     $datos['unidades'] +=100;
}
echo "<hr />";

//Vovemos a visualizar y vemos que ho ha cambiado
//Para cada producto

echo "<h2>Visualizamos los productos previamente modificados</h2>";
foreach ($productos as $producto=>$datos){
    $precio = $datos['precio'];
    $unidades = $datos['unidades'];
    echo "<h3>producto $producto precio $precio unidades $unidades</h3>";
}
```

- También podemos usar funciones de tipo cursor para recorrer el array.
- Estas funciones me permiten moverme entre los elementos y extraer el valor o índice de cada uno de ellos
- Puedo ir al primero, último, anterior o siguiente
- Para ello tendríamos las siguintes funciones

```
current() - Devuelve el elemento actual en un array
end() - Establece el puntero interno de un array a su último elemento
prev() - Rebobina el puntero interno del array
reset() - Establece el puntero interno de un array a su primer elemento
each() - Devolver el par clave/valor actual de un array y avanzar el cursor del array
key() - Obtiene una clave de un array
list() - Asignar variables como si fueran un array
next() - Avanza el puntero interno de un array
```

- Vemos el ejemplo anterior

```php
<?php

$capitales = ["España" => "Madrid", "Italia" => "Roma", "Alemania" => "Berlín"];


//Accediendo solo a los contendios
echo "<h2>Vamos a ver las capitales del array  recorriéndolo como un cursor</h2>";

reset($capitales);//Voy al primer elemento
$n = 0;
do {
    $pais = key($capitales); //Obtener el indice actual
    $capital = current($capitales); //Obtener el valor del elemento actual del array
    echo "La capital de la posición <b>$n</b> del país <b> $pais </b > es <b > $capital</b ><br />";
    $n++;
}while (next($capitales)); //Next avanza al siguiente elemento del array, cuando llegue al último dará false


?>
```

- Y la salida que genera

```
Vamos a ver las capitales del array recorriéndolo como un cursor

La capital de la posición 0 del país España es Madrid
La capital de la posición 1 del país Italia es Roma
La capital de la posición 2 del país Alemania es Berlín
```

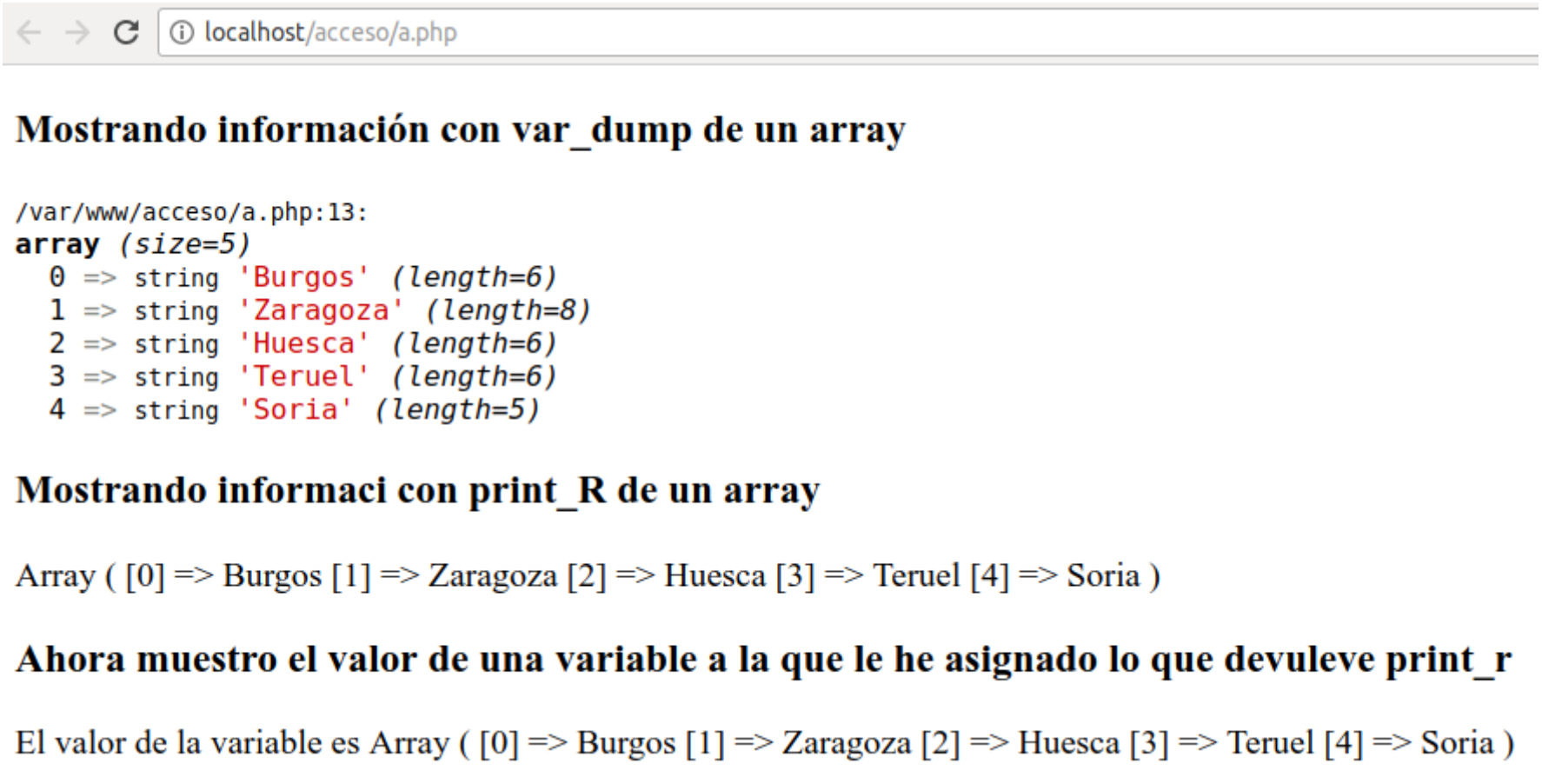
Ver el contenido de un array

- Podemos ver el contenido de un array de forma completa, usando las funciones de ***var_dump()*** y ***print_r()***.
- Podemos usar la función ya conocida ***var_dump()***.
- También podemos usar la función ***print_r***.

- La función print_r tiene un segundo parámetro booleano que por defecto es false, que sirve para hacer que la salida en lugar de sacarla por el estándar de salida la devuelva como un string.

```php
$miArray = [ "Burgos", "Zaragoza", "Huesca", "Teruel", "Soria" ];
        echo "<h3>Mostrando información con var_dump de un array</h3>";
        var_dump( $miArray ); //Muestra en tipo y contenido de la expresión $array en este caso un array
        echo "<h3>Mostrando información con print_R de un array</h3>";
        print_r( $miArray ); //Igual que el caso anterior pero con menos información en este caso solo la estructura
        $valor = "El valor de la variable es ";
        $valor .= print_r( $miArray, true );
        echo "<h3>Ahora muestro el valor de una variable a la que le he asignado lo que devuleve print_r</h3>";
        echo $valor;
```

- La salida que produce es



Funciones para manejar matrices

- Tamaño: count(), sizeof()
- Operador +: concatena dos matrices
- Recorrer una matriz next(), prev(), reset(), current(),key(), reset()
- busqueda preg_grep(), array_search(), in_array()
- Ordenar sort()
- Aquí tenemos un listado de todas las funciones, hay muchas muchas http://es1.php.net/manual/en/ref.array.php

- Para hacer referencia a los elementos almacenados en un ***array***, tienes que utilizar el valor clave entre corchetes:

```
$modulos1 [9]
$modulos2 ["DWES"]
```

- Es interesante recordar que en PHP puedes crear también arrays de varias dimensiones almacenando otro array en cada uno de los elementos de un array.

```
// array bidimensional
$ciclos = array(
     "DAW" => array ("PR" => "Programación",
                     "BD" => "Bases de datos", ...,
                     "DWES" => "Desarrollo web en entorno servidor"),
     "DAM" => array ("PR" => "Programación",
                     "BD" => "Bases de datos", ... ,
                     "PMDM" => "Programación multimedia y de dispositivos móviles")
);
```

- Para hacer referencia a los elementos almacenados en un array multidimensional, debes indicar las claves para cada una de las dimensiones:

```
$ciclos ["DAW"] ["DWES"]
```

- En PHP no es necesario que indiques el tamaño del ***array*** antes de crearlo.
- Ni siquiera es necesario indicar que una variable concreta es de tipo ***array***.
- Como ya hemos visto, simplemente puedes comenzar a asignarle valores:

```php
// array numérico
$modulos1 [0] = "Programación";
$modulos1 [1] = "Bases de datos";
...
$modulos1 [9] = "Desarrollo web en entorno servidor";
// array asociativo
$modulos2 ["PR"] = "Programación";
$modulos2 ["BD"] = "Bases de datos";
...
$modulos2 ["DWES"] = "Desarrollo web en entorno servidor";
```

- En PHP tampoco es necesario que especifiques el valor de la clave.
- Al omitirla el array se irá llenando a partir de la última clave numérica existente, o de la posición 0 si no existe ninguna:

```php
$modulos1 [ ] = "Programación";
$modulos1 [ ] = "Bases de datos";
...
$modulos1 [ ] = "Desarrollo web en entorno servidor";
```

Plantilla:MRM Tarea

Actividad

- Implementar una función que busca si un determinado valor aparece en una matriz.
- La función recibe 2 parámetros:

1. la matriz
2. el elemento a buscar,

- Retorna si ha encontrado el valor (TRUE) o no (FALSE).

1. Implementar la función, con los parámetros (el array, y el valor a buscar).
2. Para probar la función implementada, generar un array de 100 posiciones de valores
3. valores enteros entre 1 y 100.
4. Generar, también, el número que hay que buscar en el array.
5. Llamar a la función con el array y el valor como parámetros de la función.
6. Mostrar los resultados por pantalla.

Actividad

Random Images

- Escribir un programa que:
  - Inicialice un vector con 10 imágenes

podéis utilizar éste: código.php (http://www.tecn.upf.es/~ocelma/cpom/practicas/php/random_images/crea_array.txt) Crea el vector $imagenes.

- - La página debe mostrar, aleatoriamente, 3 imágenes

```
(puedes usar como alternativa  la función shuffle ($imagenes)
desordena el vector),
o usar un rand para obtener indices aleatorios.
```

- - Cada 5 segundos ha de refrescarse la página para ir mostrando imágenes distintas

(podéis usar, por ejemplo, este trozo de código HTML y añadirlo en el <HEAD> de la página

```
<head>
        <meta charset="UTF-8"/>
        <meta http-equiv="refresh" content="5" url="index.php"/>
```

```
        <title></title>
    </head>
</div>
```

Variables globales Vs superglobales

- Ya hemos visto como en php una variable tiene el ámbito en el cual es accesible y visible
- Las variables son locales a la función en la cual aparecen, si queremos acceder dentro de una función a una variable del script y actuar sobre su valor, debemos hacerla ***global***.
- PHP Dispone un un importante conjunto de variables superglobales.
- El desarrollador tiene acceso a dichas variables en cualquier momento del script.
- El sistema se encarga de tenerlas actualizadas, con el valor correspondiente

Superglobales

- PHP incluye unas Son variables internas predefinidas que pueden usarse desde cualquier ámbito, por lo que reciben el nombre de variables superglobales.
- No es necesario que uses global para acceder a ellas.

- Cada una de estas variables es un ***array*** que contiene un conjunto de valores
- Posteriormente veremos cómo se utilizan los arrays).
- Aquí puedes acceder a las variables superglobales (http://es.php.net/manual/es/language.variables.superglobals.php) disponibles en PHP se pueden ver son las siguientes:

Superglobales (Algunas principales)

1. $GLOBALS Hace referencia a todas las variables disponibles en el ámbito global
2. $_SERVER Información del entorno del servidor y de ejecución
3. $_GET Variables HTTP GET
4. $_POST Variables HTTP POST
5. $_FILES Variables de Carga de Archivos HTTP
6. $_COOKIE Cookies HTTP
7. $_SESSION Variables de sesión

8. $_REQUEST Variables HTTP REQUEST. Un array asociativo que por defecto contiene el contenido de $_GET, $_POST y $_COOKIE.
9. $_ENV

- Analizaremos una de ellas

$_SERVER.

Contiene información sobre el entorno del servidor web y de ejecución. Entre la información que nos ofrece esta variable, tenemos:
Principales valores de la variable $_SERVER

1. $_SERVER['PHP_SELF']: script que se está ejecutando actualmente.
2. $_SERVER['SERVER_ADDR']: dirección IP del servidor web.
3. $_SERVER['SERVER_NAME']: nombre del servidor web.
4. $_SERVER['DOCUMENT_ROOT']: directorio raíz bajo el que se ejecuta el guión actual.
5. $_SERVER['REMOTE_ADDR']:dirección IP desde la que el usuario está viendo la página.
6. $_SERVER['REQUEST_METHOD']:método utilizado para acceder a la página ('GET', 'HEAD', 'POST' o 'PUT')

Actividad

Haz un script que nos de la información de las variables vistas anteriormente

---