UNIDAD DIDÁCTICA 4:
ESTRUCTURAS DE DATOS EN PHP. LOS ARRAYS.

Módulo profesional:
Desarrollo Web en Entorno Servidor

Duración: 160 Horas.
Código: 0613

Ciclo Formativo de Grado Superior:
Desarrollo Aplicaciones Web

Contenido

Resumen Introductorio

En esta unidad conoceremos un instrumento básico en cualquier lenguaje de programación como son los arrays. Son estructuras de datos que pueden ser tan complejas como nuestras necesidades tengamos y que nos permiten de una forma rápida almacenar y acceder a información.

En PHP los arrays son muy flexibles, al igual que el resto de variables, por lo que nos será muy sencillo poder utilizarlos.

Introducción

En el lenguaje PHP las variables nos permite almacenar una única variable con un un nombre y un contenido. Pero qué ocurre si necesitamos almacenar grandes cantidades de información para después devolver la misma de una forma rápida y automática.

Para resolver esta problemática tenemos los arrays, que no es una creación del lenguaje php, sino que en realidad es una solución de otros lenguajes también.

En el caso práctico de PHP, los arrays son muy flexibles basándose en mapas ordenados, clave/valor, y de esta forma podremos generar tanto arrays lineales de información, como matrices o mapas más complejos.

Caso Introductorio

Queremos poder imprimir por pantalla de forma automática información almacenada o bien generarla de forma automática a partir de una información base.

Es el caso de generación de tablas html automáticas a partir de un array. Cuando finalices esta unidad serás capaz de realizar estos automatismos.

Desarrollo:

1. Arrays

Comencemos con una definición, la que nos proporciona php.net.

Arrays en PHP

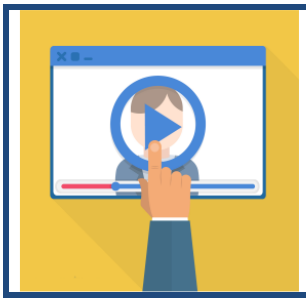
http://php.net/manual/es/language.types.array.php

Un array en PHP es en realidad un mapa ordenado. Un mapa es un tipo de datos que asocia valores con claves. Este tipo se optimiza para varios usos diferentes; se puede emplear como un array, lista (vector), tabla asociativa (tabla hash - una implementación de un mapa), diccionario, colección, pila, cola, y posiblemente más. Ya que los valores de un array pueden ser otros arrays, también son posibles árboles y arrays multidimensionales.

Los arrays los podemos entender como conjutos de variables, como contenedores de información, como almacenes de información que nos facilitan dos aspectos muy importantes: nos facilita cómo almacenar la información pero sobre todo como recuperarla.

1.1. ¿Por qué utilizamos arrays?

Cuando nos encontramos con un sistema en el que tenemos muchas variables, sobre una misma temática e incluso que se denominan casi parecido, nos encontramos en el justo instante y necesidad de utilizar arrays.



En el vídeo que encontrarás una introducción a los arrays, para qué sirven y ejemplo de creación

https://youtu.be/Q2OuTvntHU0

En el siguiente enlace tienes el código utilizado:

https://github.com/pacogomezarnal/codigoEjemploCursoPHP/tree/master/array

1.2. Definición de los arrays

Como muy bien nos indica la documentación de php.net, para definir un nuevo array podemos definirlo de dos formas diferentes:

- Mediante la palabra reservada *array()*
- Mediante el *doble corchete []*, a partir de la versión 5.4 y que por lo tanto se parece mucho más a las versiones modernas de otros lenguajes.

Lo mejor es comenzar viéndolo con ejemplos.

Ejemplo creación Array

Ejemplo creación array

```php
<?php
$personas=array(
1=>"Paco",
2=>"Manolo",
3=>"Lucia"
);
?>
```

Como se puede observar:

- Se crea una nueva variable denominada personas
- Esta variable es de tipo array
- Contiene 3 elementos
  - Las claves son integer
  - Los valores son String

Este primer ejemplo nos da pie a analizar la definición que nos proporciona php.net sobre array.

Arrays en PHP

http://php.net/manual/es/language.types.array.php

Un array puede ser creado con el constructor del lenguaje array(). Éste toma cualquier número de parejas ***clave*** =>***valor*** como argumentos.

```
array(
    clave  => valor,
    clave2 => valor2,
    clave3 => valor3,
    ...
    )
```

La coma después del último elemento del array es opcional, pudiéndose omitir. Esto normalmente se hace para arrays de una única línea, es decir, es preferible array(1, 2) que array(1, 2,). Por otra parte, para arraysmultilínea, la coma final se usa frecuentemente, ya que permite una adición más

	sencilla de nuevos elementos al final.

La segunda forma de inicializar y crear un array sería utilizando el doble corchete:

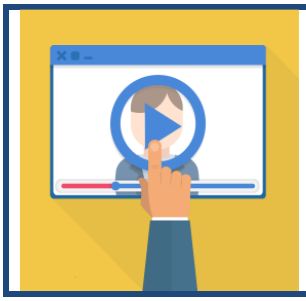
Ejemplo creación array

```php
<?php
$personas=[
1=>"Paco",
2=>"Manolo",
3=>"Lucia"
];
?>
```

Esta forma, aquellos que hayáis ya programado en otros lenguajes de programación quizá sea más sencilla de leer, pero en definitiva el resultado es el mismo.

1.3. Clave=>Valor

Es importante detenernos en cómo se manejan y como se definen los arrays. Puede ser interesante volver a ver el vídeo explicativo inicial.



En el vídeo que encontrarás una explicación del uso de los arrays

https://youtu.be/TwqBdoLPb0c

En el siguiente enlace tienes el código utilizado:

https://github.com/pacogomezarnal/codigoEjemploCursoPHP/tree/master/array

Los arrays en php únicamente pueden ser definidos como un *mapa* asociativo clave valor. Esto quiere decir que todo valor definido dentro del

array siempre va a tener una clave con la cual acceder a dicho valor se haya o no se haya asignado.

Ejemplo creación array sin claves

```php
<?php
$personas=["Paco","Manolo","Lucia"];
?>
```

En el anterior ejemplo vemos que no se ha definido *claves* para ningún valor. En este caso php, automáticamente les asigna claves incrementales de tipo integer.

(array claves mezcladas)

Ejemplo creación array claves mezcladas

```php
<?php
$personas=[
1=>"Paco",
"jefes"=>"Manolo",
3=>"Lucia"
];
?>
```

En el anterior ejemplo por el contrario podemos ver como hemos definido un array donde las claves pueden ser de los dos tipos permitidos en php, tanto integer como string.

A partir de estos ejemplos nos podemos hacer una idea que cualquier combinación es válida:

- Array no indexados (sin claves), automáticamente se le asigna integer
- Array con índices integer y string
- Array con índices y sin índices

1.4. Acceso a la información de un array

Una vez que tenemos claro cómo crear un array, queremos poder utilizar el array generado y por lo tanto utilizar el contenido y/o la información contenida en él.

Para ello seguiremos el siguiente vídeo.

Lo que hemos visto en el video por lo tanto es que si tenemos el siguiente código ejemplo:

Ejemplo creación array claves mezcladas

```php
<?php
$personas=[
1=>"Paco",
```

```
"jefes"=>"Manolo",
3=>"Lucia"
];
?>
```

(creaciónarray sin claves)

Para acceder al contenido del primer elemento, deberemos utilizar la sintaxis *$personas[1],* para acceder al del segundo elemento deberemos utilizar la sintaxis *$personas["jefes"],* y el de la tercera sería mediante *$personas[1].*

En el caso de un array no indexado:

Ejemplo creación array sin claves

```
<?php
$personas=["Paco","Manolo","Lucia"];
?>
```

(Ejemplo creación array sin claves e impresión por pantalla)

Para acceder al contenido del primer elemento, deberemos utilizar la sintaxis *$personas[0],* para acceder al del segundo elemento deberemos utilizar la sintaxis *$personas[1],* y el de la tercera sería mediante *$personas[2].*

Tal y como hemos visto en el vídeo y en el código ejemplo, vamos a utilizar continuamente el acceso a los arrays combinándolo con las funciones de impresión por pantalla como es echo. Veámoslo con un ejemplo:

Ejemplo creación array sin claves e impresión por pantalla

```
<?php
$personas=["Paco","Manolo","Lucia"];
echo "La primera posicion es ".$personas[0];
?>
```

(adecuación de formatos)

Lo cual nos mostrará por pantalla *La primera posición es Paco*. PHP es un lenguaje poco tipificado, lo hemos comentado en situaciones anteriores, y es el propio servidor/intérprete quien se encarga de realizar las transformaciones pertinentes para adecuar los formatos.

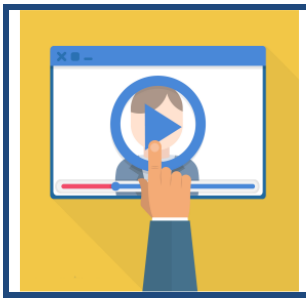
Ejemplo de adecuación de formatos

```
<?php
$personas=["Paco",42,1.34];
echo"La edad de ".$personas[0]." es ".$personas[1];
?>
```

Como podéis observar en el anterior ejemplo:

- *$persona[0]*, devolverá un String, cuyo valor es Paco
- *$persona[1]*, devolverá un número cuyo valor es 42
- La función echo imprimirá el resultado "La edad de Paco es 42"

Esta es una de las grandes ventajas de PHP cuando nos referíamos a este lenguaje como poco tipificado. Como podemos observar, no sólo el array $personas almacena diferente tipos de variables sin ningún tipo de problema, sino que funciones externas hacen uso de dichos contenidos y las transforman de un tipo a otro sin necesidad de realizar castings.



En el vídeo que encontrarás una explicación y ejemplo de uso de arrays

https://youtu.be/3_Z4yb7SjAY

En el siguiente enlace tienes el código utilizado:

https://github.com/pacogomezarnal/codigoEjemploCursoPHP/tree/master/array

1.5. Modificación del contenido

Una vez que sabemos cómo crear arrays y acceder al contenido del mismo, necesitamos saber cómo modificar el contenido por un lado del array y por otro cómo incluso agrandar y/o acortar borrando elementos del array. Para ello nos basaremos en las explicaciones que mostramos en el siguiente vídeo y código:

Analicemos la parte de modificación de contenido a través del ejemplo:

Ejemplo de modificación de un elemento

```php
<?php
//Arrays
$alturasPrimer=array(164,178,169,182);
$alturasSegundo=[164,178,169,182];
//Mostrar una posicion del Array
echo"La posicion 0 del array es: ".$alturasSegundo[0];
echo"<br>";
echo"La posicion 2 del array es: ".$alturasSegundo[2];
echo"<br>";
//Modificar el contenido de una posicion
$alturasSegundo[2]=178;
echo"La nueva posicion 2 del array es: ".$alturasSegundo[2];
```

```
?>
```

(añadir un elemento)

En este caso observamos:

- En primer lugar se genera el array*$alturasSegundo*
- Modificamos la tercera posición del array mediante *$alturasSegundo[2]=178;*

Ejemplo de añadir un elemento

```
<?php
//Arrays
$alturasPrimer=array(164,178,169,182);
$alturasSegundo=[164,178,169,182];
//Anyadir un nuevo elemento
$alturasSegundo[]=189;
?>
```

(eliminar un elemento)

En este ejemplo:

- En primer lugar se genera el array*$alturasSegundo*
- Añadiremos un nuevo elemento *$alturasSegundo[]=189;*

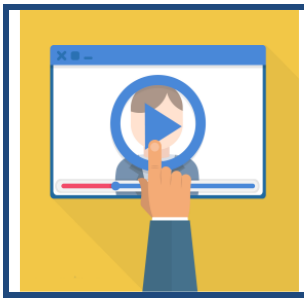
Por último para eliminar un elemento de un array tendremos que:

Ejemplo de eleminar un elemento

```
<?php
//Arrays
$alturasPrimer=array(164,178,169,182);
$alturasSegundo=[164,178,169,182];
//Anyadir un nuevo elemento
$alturasSegundo[]=189;
//Eleminar un elemento
unset($alturasSegundo[2]);
?>
```

En este ejemplo:

- En primer lugar se genera el array*$alturasSegundo*
- Eliminaremos un elemento *unset($alturasSegundo[2]);*



En el vídeo que encontrarás una explicación y ejemplo de uso de arrays
https://youtu.be/3_Z4yb7SjAY

En el siguiente enlace tienes el código utilizado:

https://github.com/pacogomezarnal/codigoEjemploCursoPHP/tree/master/array

1.6. Arrays multidimensionales

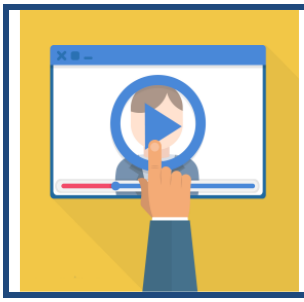
La tipología de uso de los arrays en php hace que sea muy flexible y cómodo poder utilizarlos y crear estructuras más complejas, en este caso estructuras multidimensionales como la que vemos en el ejemplo:

Ejemplo de array multidimensional

```php
<?php
//Arrays multidimensional
$clasesUniversidad=[
0=>[
0=>"Paco",
1=>"Lucia",
2=>"Marcos"
],
1=>[
0=>"Paco",
1=>"Pedro",
2=>"Juan"
]
];
?>
```

En el anterior ejemplo vemos como hemos desplegado un nivel más dentro de nuestro array*$clasesUniversidad*. Donde el primer nivel podría indicar el número de clase y el segundo nivellos nombres de los alumnos.

Para comprender de una forma más detallada los arrays multidimensionales, veamos el siguiente vídeo:



En el vídeo que encontrarás una explicación y ejemplo de uso de arrays multidimensionales
https://youtu.be/JWjehwRsnTk

En el siguiente enlace tienes el código utilizado:

https://github.com/pacogomezarnal/codigoEjemploCursoPHP/tree/master/array

Otro ejemplo de array multidimensional

```php
<?php
//Arrays multidimensional
<?php
$pais=[
"CValenciana"=>array(
"prov1"=>"Castellon",
"prov2"=>"Valencia",
"prov3"=>"Alicante"
),
"Murcia"=>array(
"prov1"=>"Murcia"),
"Cataluña"=>array(
1=>"Girona",
2=>"Lerida"
)
];
?>
```

2. Funciones sobre arrays

Los arrays en cualquier lenguaje de programación dan pie a que existan muchas operaciones sobre los mismos para poder realizar acciones tales como ordenar, mostrar, copiar, extraer, dividir, combinar, ...

Y es por ese motivo que al igual que en otros lenguajes de programación, php le dedica un gran capítulo a las funciones y operaciones sobre arrays.

2.1. Estructura de control for

La estructura *for* es una de las estructuras más utilizadas dentro de cualquier lenguaje de programación, para la generación de bucles de programación como para recorrer arrays o estructuras de datos. Según la documentación de php la sintaxis de un bucle for es:

```
for (expr1; expr2; expr3)
    sentencia
```

Imagen 1: Estructura for

Fuente de la imagen:http://php.net/manual/es/control-structures.for.php

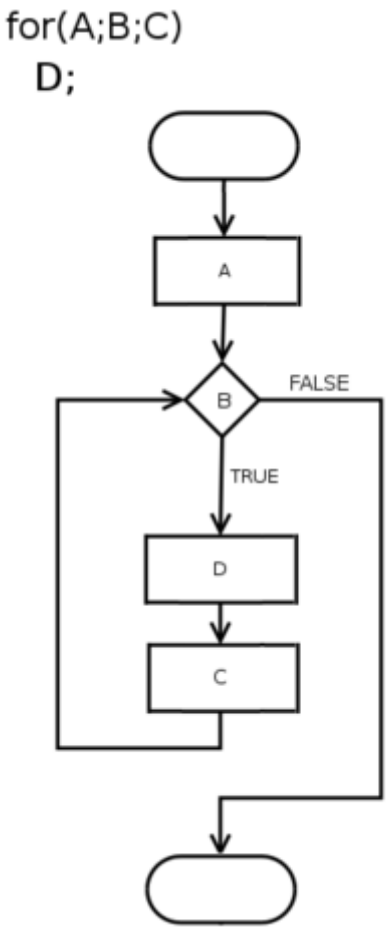


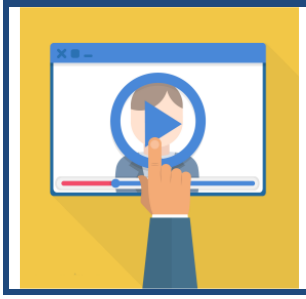
Imagen 2: Estructura for

Fuente de la imagen:https://upload.wikimedia.org/wikipedia/commons/thumb/0/06/For-loop-diagram.png/220px-For-loop-diagram.png

La primera expresión (expr1) es evaluada (ejecutada) una vez incondicionalmente al comienzo del bucle. En el comienzo de cada iteración, se evalúa expr2. Si se evalúa como TRUE, el bucle continúa y se ejecutan la/sy sentencia/s anidada/s. Si se evalúa como FALSE, finaliza la ejecución del bucle. Al final de cada iteración, se evalúa (ejecuta) expr3.

En el siguiente enlace tienes el código utilizado:

https://github.com/pacogomezarnal/codigoEjemploCursoPHP/tree/master/estructuraFor



En el vídeo que encontrarás una explicación y ejemplo de uso de for

https://youtu.be/ExUL4FP97H4

Ejemplo mostrado en el vídeo

```
<!DOCTYPE html>
<html>
<head>
<metacharset="utf-8">
<title>Ejemplo 1 de FOR</title>
</head>
<body>
<?php
//Contar del 1 al 10
for($i=1;$i<11;$i++){
echo$i."<br>";
}
echo"Fuera del bucle for<br>";
echo$i;
?>
</body>
</html>
```

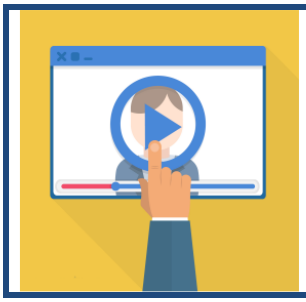
En el ejemplo podemos ver diferentes cuestiones:

- Como se embebe código php dentro de html. Práctica común a partir de ahora en nuestros ejemplos.

- Como se ejecuta una sentencia for:
  - Se inicializa $i=1
  - Se ejecuta echo $i."
"
  - Se incrementa $i
  - Se comprueba si $i < 11

2.2. Estructura de control for y html

En PHP un uso muy habitual y muy utilizado con la estructura for es la de combinarla con html para "mostrar" partes de código de forma automática. El caso más habitual es utilizarla con la conexión a la base de datos y por lo tanto de esta forma tener el contenido representado de una forma automática. En nuestro caso realizaremos un ejemplo con la etiqueta ul:



En el vídeo que encontrarás una explicación y ejemplo de uso de for

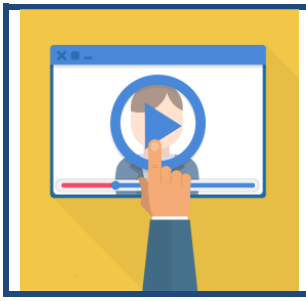
https://youtu.be/PFYbGMsVF4E

En el siguiente enlace tienes el código utilizado:

https://github.com/pacogomezarnal/codigoEjemploCursoPHP/tree/master/estructuraFor

2.3. Estructura de control for y arrays

Y por supuesto el uso principal de for es combinarlo con los arrays para recorrerlos de forma automática. Utilizando el mismo ejemplo que el realizado anteriormente, recorreremos un array para pintar un menu



En el vídeo que encontrarás una explicación y ejemplo de uso de for con arrays

https://youtu.be/XJZtOWTc4jQ

En el siguiente enlace tienes el código utilizado:

https://github.com/pacogomezarnal/codigoEjemploCursoPHP/tree/master/estructuraFor

Ejemplo mostrado en el vídeo

```
<!DOCTYPE html>
<html>
<head>
<metacharset="utf-8">
<title>Estructura For con HTML</title>
<linkrel="stylesheet"href="estilos.css">
</head>
<body>
<ul>
<?php
$menu=[
"Home",
"Nosotros",
"Productos",
"Contactar"
];
for($i=0;$i<count($menu);$i++){
echo"<li class=\"menu\">".$menu[$i]."</li>";
}
?>
</ul>
</body>
</html>
```

Qué está ocurriendo en el anterior ejemplo:

- En primer lugar generamos un array denominado $menu.
- Sus claves van desde el 0 hasta longitud – 1
- El bucle for inicializa la varible $i a 0, por lo que coincide justamente con el elemento primero del array
- Mediante el echo, mostramos uno a uno los elementos del array

RECUERDA…

En un array no indexado, las claves son generadas automáticamente y comienzan desde el índice 0

2.4. Foreach

Qué ocurre cuando queremos recorrer un array con claves que no sean numéricas, si probamos con la estructura for veremos que se produce un error ya que no existe esas claves numéricas. Para solucionar dicho problema tenemos la estructura ***foreach*** que según la documentación de PHP:

Foreach

http://php.net/manual/es/control-structures.foreach.php

El constructor foreach proporciona un modo sencillo de iterar sobre arrays. foreach funciona sólo sobre arrays y objetos, y emitirá un error al intentar usarlo con una variable de un tipo diferente de datos o una variable no inicializada.

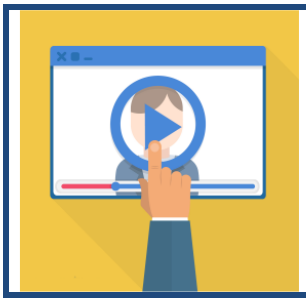
Existen dos sintaxis tal y como nos indica la documentación:

Dos sintaxis

```php
<?php
foreach(expresión_arrayas$clave=>$valor)
    sentencias

foreach(expresión_arrayas$valor)
    sentencias
?>
```

La primera sintaxis es la más completa, ya que en cada iteración tendremos disponibles tanto la clave como el valor del array. En la segunda acepción, sólo tendremos el contenido o valor del array. Desarrollemos más a través de un video:



En el vídeo que encontrarás una explicación y ejemplo de uso de foreach

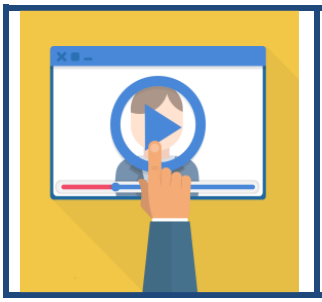
https://youtu.be/qURvE1amLtI

En el siguiente enlace tienes el código utilizado:

https://github.com/pacogomezarnal/codigoEjemploCursoPHP/tree/master/estructuraFor

(Dos formas de uso)

Tal y como explicábamos al principio, la estructura foreach puede ser utilizada de dos formas, la rápida donde sólo usamos el valor del contenido del array, y la más compleja donde queremos tener un control absoluto del recorrido de nuestro array teniendo tanto la clave como el valor del mismo.



En el vídeo que encontrarás una explicación y ejemplo de uso de foreach con su sintaxis completa
https://youtu.be/TFz-33S9Iv0

2.5. Funciones sobre arrays

Muchas son las funciones que tenemos en php para poder manejar arrays. Veamos un listado de las más importantes:

Nombre de la funcion	Explicación y ejemplo

count	Cuenta todos los elementos de un array o algo de un objeto
	`<?php`
	`$a[0]=1;`
	`$a[1]=3;`
	`$a[2]=5;`
	`$resultado=count($a);`
	`// $resultado == 3`
	`?>`

asort	Ordena un array y mantiene la asociación de índices
	`<?php`
	`$fruits=array("d"=>"lemon","a"=>"orange","b"=>"banana","c"=>"apple");`
	`asort($fruits);`
	`foreach($fruitsas$key=>$val){`
	`echo"$key = $val\n";`
	`}`
	`/* EL RESULTADO ES`
	`c = apple`
	`b = banana`
	`d = lemon`
	`a = orange`
	`*/`
	`?>`

shuffle	Mezcla un array
	`<?php`
	`$números=range(1,20);`

	shuffle($números);
	foreach($números as $número){
	echo "$número ";
	}
	?>

array_merge	Combina dos o más arrays
	ray1=array("color"=>"red",2,4);
	$array2=array("a","b","color"=>"green","shape"=>"trapezoid",4);
	$resultado=array_merge($array1,$array2);
	print_r($resultado);
	/* EL RESULTADO ES
	Array
	(
	[color] => green
	[0] => 2
	[1] => 4
	[2] => a
	[3] => b
	[shape] => trapezoid
	[4] => 4
	)
	*/
	?>

Este es una pequeña muestra de las funciones que php pone a nuestra disposición. Pero como hemos indicado anteriormente hay muchas más funciones y ejemplos.

Funciones sobre arrays

http://php.net/manual/es/ref.array.php

En esta dirección podemos observar todas las funciones que disponemos sobre los arrays.

2.6. Funciones para visualizar arrays

Ya en ejercicios anteriores hemos utilizado funciones especializadas o estructuras para poder visualizar un array. Por lo que podemos hacer un resumen de las opciones que tenemos para poder visualizar por pantalla un array:

- OPCIÓN 1: Utilizando estructuras de control tanto *for*para arrays indexados con integer o indexados automáticamente, como utilizando *foreach*, que nos sirve para cualquier tipo de arrays. Esta opción nos permite presentar arrays tanto en modo *debug* como en producción.

- OPCIÓN 2: Utilizando las funciones *print_r* o *var_dump*, ambas opciones nos muestran por pantalla de forma rápida un array de cualquier tipo, pero sólo lo podemos utilizar en modo debug.

Ejemplo de var_dump

```php
<?php
$a=array(1,2,array("a","b","c"));
var_dump($a);
/* EL RESULTADO ES
array(3) {
  [0]=>
int(1)
  [1]=>
  int(2)
  [2]=>
  array(3) {
    [0]=>
    string(1) "a"
    [1]=>
    string(1) "b"
[2]=>
string(1) "c"
  }
}
*/
?>
```