



PHP – Depurar aplicaciones



Ciclo: DAW
Módulo: DWES
Curso: 2020-2021
Autor: César Guijarro Rosaleny



Introducción.....	3
Mostrar los errores en el servidor.....	5
Mostrar el valor de las variables.....	6
echo y print.....	6
print_r y var_dump.....	7
Xdebug y PHP Debug.....	9

Introducción

Para poder encontrar errores en nuestros archivos PHP tenemos que tener algo de información sobre el error, si no sería casi imposible descubrirlo en miles de líneas de código.

Por lo tanto, mientras estamos desarrollando una web tenemos que tener la posibilidad de mostrar los errores en tiempo de ejecución.

Sin embargo, cuando nuestra web está en producción puede que no nos interese mostrar esos errores. Si el intérprete PHP encuentra algún fallo, lo mostrará por el navegador y lo podrá ver cualquier usuario que entre en nuestra web.

Notice: Undefined variable: a in /var/www/html/DWS/index.php on line 4

Además, los mensajes de error pueden dar información sobre cómo están codificadas algunas partes de la web, incluso en temas de seguridad, lo cual es mucho más grave que un simple motivo estético.

Afortunadamente, podemos habilitar o deshabilitar la función de mostrar errores en función de si el sitio está en desarrollo o en producción. De hecho, muchos *hostings* tienen deshabilitada la opción de mostrar errores por defecto por los motivos anteriores.

Otro problema es cómo descubrir el error con la información que se nos muestra por pantalla. Si estás acostumbrado a programar en Java o C con algún IDE como *Netbeans* o *Eclipse*, puede que hayas utilizado algunas de sus útiles opciones de depuración, como *breakpoints*, *watches*...

Programando PHP con VSCode no tenemos, en principio, ninguna herramienta de depuración para poder usar todas esas opciones. La forma tradicional de depurar en PHP es mostrar por el navegador el valor de las variables mediante las funciones que nos ofrece el lenguaje.

La buena noticia es que podemos utilizar herramientas desarrolladas por terceros para instalar extensiones que nos faciliten la mayoría de esas opciones. Una de las más populares es [Xdebug](#).

Mostrar los errores en el servidor

Para poder mostrar los errores de tu web podemos hacerlo de varias formas. Si queremos que sea permanente, podemos editar nuestro archivo **php.ini** y añadir la línea:

```
display_errors = on;
```

De la misma manera, si queremos que PHP no nos muestre los errores cambiaremos la línea anterior por:

```
display_errors = off;
```

Puede ser que tu proveedor de hosting no permita modificar el archivo **php.ini**. O puede que quieras mostrar los errores sólo en uno de tus archivos PHP. Para eso, debes añadir al principio del archivo en cuestión (de todos los que quieras que se muestren errores):

```
<?php
    ini_set('display_errors', 1);
    ini_set('display_startup_errors', 1);
    error_reporting(E_ALL);
```

De esta manera, el navegador nos informará del error aunque la directiva *display_errors* de *php.ini* esté en *off*.

Mostrar el valor de las variables

echo y print

La primera opción que se nos viene a la cabeza para mostrar una variable por pantalla es utilizar **echo** o **print**. Podemos usar cualquiera de las dos para que nos muestre el valor de una variable en un momento dado.

```
<?php
    $a = 5;
    echo "La variable a vale: $a<br>";
```



La variable a vale: 5

Si el error es fácil de localizar, nos pueden valer estas dos funciones. El problema es la forma de mostrar los datos. Vamos a cambiar el tipo de datos de `$a` a un array numérico.

```
<?php
    $a = array(9,12,34,56,78);
    echo "La variable a vale: $a<br>";
```

Si observamos la salida del navegador, vemos que la única información que nos da sobre la variable es que es un *array*, pero no nos muestra sus valores por ningún lado.



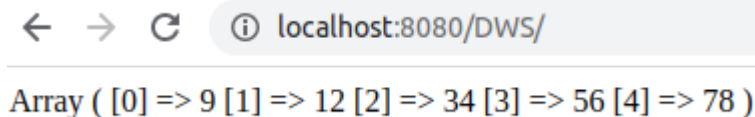
La variable a vale: Array

print_r y var_dump

Para mostrar los valores de los tipos de datos complejos, tenemos otras dos funciones que nos pueden ser útiles: **print_r** y **var_dump**.

La función **print_r** imprime información legible para los humanos sobre una variable.

```
<?php
$a = array(9,12,34,56,78);
print_r ($a);
```

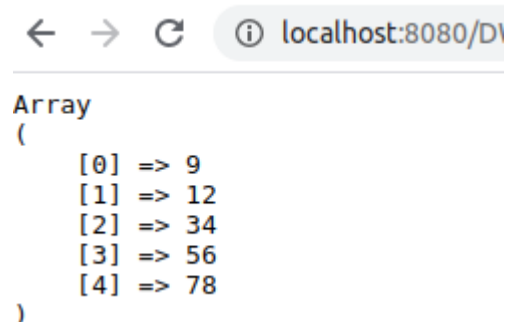


```
← → ↻ ⓘ localhost:8080/DWS/
Array ( [0] => 9 [1] => 12 [2] => 34 [3] => 56 [4] => 78 )
```

Esta función ya nos muestra los valores del *array*, aunque sigue sin darnos mucha más información y el formato no es muy cómodo.

Podemos anidarla con la etiqueta `<pre>` de HTML para mejorar la legibilidad.

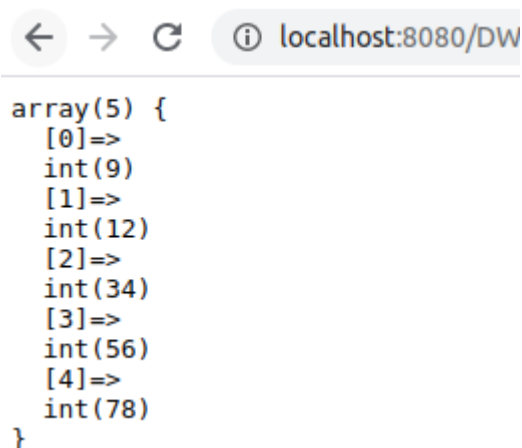
```
<?php
$a = array(9,12,34,56,78);
echo "<pre>";
print_r ($a);
echo "</pre>";
```



```
← → ↻ ⓘ localhost:8080/D...
Array
(
    [0] => 9
    [1] => 12
    [2] => 34
    [3] => 56
    [4] => 78
)
```

Si queremos mostrar más información, podemos usar **var_dump**. Además de mostrarnos el valor de la variable, nos muestra el número de elementos del *array* y el tipo de datos de los valores.

```
<?php
$a = array(9,12,34,56,78);
echo "<pre>";
var_dump($a);
echo "</pre>";
```



```
array(5) {
  [0]=>
  int(9)
  [1]=>
  int(12)
  [2]=>
  int(34)
  [3]=>
  int(56)
  [4]=>
  int(78)
}
```

En cualquier caso, tener que escribir tres líneas de código para mostrar el valor de una variable puede ser un tanto engorroso.

Aunque hay ocasiones en que mostrar los errores por pantalla se queda cortos a la hora de depurar, sigue siendo uno de los métodos más utilizados para encontrar errores en los ficheros PHP.

Xdebug y PHP Debug

Otra de las opciones que tenemos para aprovechar la potencia de los depuradores es instalar alguna herramienta implementada por terceros. Una de las más populares a la hora de depurar PHP es [Xdebug](#) en combinación con algún plugin de VSCode como [PHP Debug](#).

Para instalar *Xdebug* basta con ir al apartado [instalación](#) de su web y seguir las instrucciones según el sistema operativo que estemos usando.

Por su parte, *PHP Debug* se instala como cualquier otra extensión de VSCode. Pulsamos CTRL+p para abrir la barra de búsqueda y añadimos el código para instalar la extensión (también podemos buscarla directamente en el apartado extensiones):

```
ext install felixfbecker.php-debug
```

Después tendréis que configurar algunos parámetros dependiendo de que servidor HTTP estéis utilizando, sistema operativo, ubicación de vuestras webs...

Una vez instalada la extensión, ya podéis utilizar las herramientas propias de un depurador, como *breakpoints*, *watches*...

Además, si tenemos instalado *Xdebug*, se modificará el formato en que nos muestra los errores y la forma de visualizar las variables mediante **var_dump**, con lo que no hará falta usar `<pre>`.

```
<?php
    $a = array(9,12,34,56,78);
    var_dump ($a);
```

← → ↻ ⓘ localhost:8080/D

```
/var/www/html/DWS/index.php:3:
array (size=5)
  0 => int 9
  1 => int 12
  2 => int 34
  3 => int 56
  4 => int 78
```

← → ↻ ⓘ localhost:8080/DWS/

(!) Notice: Undefined variable: a in /var/www/html/DWS/index.php on line 3				
Call Stack				
#	Time	Memory	Function	Location
1	0.1542	397288	{main}()	.../index.php:0

/var/www/html/DWS/index.php:3:null