

## **UD 3. Bases de datos en PHP. Herramientas para depuración de código.**

### **Índice de contenidos**

Previo. 0. Herramientas para depuración de código.....	2
0.1. Instalación de herramientas de depuración.....	3
0.2. Utilizando herramientas de depuración.....	5

## Previo. 0. Herramientas para depuración de código

Seguramente en la aplicación de tienda online que has programado en la UD anterior te hayas encontrado con algunos problemas: algún código que no funcionaba, páginas que no mostraban lo que deberían, ...

Con lo que has visto hasta ahora ya puedes empezar a desarrollar algunas aplicaciones web sencillas en lenguaje PHP. Sin embargo, cuando empiezan a surgir problemas tienes que ingeniártelas para poder descubrir qué es lo que está fallando. Puedes poner trozos de código en las páginas que desarrollas para mostrar información interna de la aplicación, imprimir mensajes para comprobar las partes del código que se van ejecutando o imprimir el contenido de las variables que usas. Para ello puedes utilizar `echo`, `print`, `print_r` o `var_dump`.

*Para saber más ...*

En la [documentación de PHP](#) puedes consultar información sobre la función `var_dump`.

Por ejemplo, si quieres ver el contenido de la variable superglobal `$_SERVER`, puedes introducir una línea como:

```
print_r($_SERVER);
```

Este método de depuración es muy tedioso y requiere hacer cambios constantes en el código de la aplicación. También existe la posibilidad de que, una vez encontrado el fallo, te olvides de eliminar de tus páginas algunas de las líneas creadas a propósito para la depuración, con los consiguientes problemas.

*Para saber más ...*

Si quieres conocer más sobre los principios de la depuración, y la utilización de comandos como `echo` o `print_r` para depurar tu código, puedes consultar [este artículo de php-hispano](#) ([copia en archive.org](#)).

Si ya has programado anteriormente en otros lenguajes, seguramente conoces algunas herramientas de depuración específicas que se integran con el entorno de desarrollo, permitiéndote por ejemplo detener la ejecución cuando se llega a cierta línea y ver el contenido de las variables en ese momento. Al usar estas herramientas minimizas o eliminas por completo la necesidad de introducir líneas específicas de depuración en tus programas y agilizas el procedimiento de búsqueda de errores.

De las herramientas disponibles que posibilitan la depuración en lenguaje PHP, en esta unidad aprenderás a configurar y utilizar la extensión Xdebug. Es una extensión de código libre, bajo

licencia propia (The Xdebug License, basada en la licencia de PHP) que se integra perfectamente con los IDE.

Si la depuración está activada, Xdebug controla la ejecución de los scripts en PHP. Puede pausar, reanudar y detener los programas en cualquier momento. Cuando el programa está pausado, Xdebug puede obtener información del estado de ejecución, incluyendo el valor de las variables (puede incluso cambiar su valor).


Xdebug es un servidor que recibe instrucciones de un cliente (el IDE). De esta forma, no es necesario que el entorno de desarrollo esté en la misma máquina que el servidor PHP con Xdebug.

## 0.1. Instalación de herramientas de depuración

Para instalar Xdebug deberemos descargarnos la versión que sea compatible con nuestra instalación de PHP. Lo más fácil es utilizar el analizador de configuración de la propia web de Xdebug, que nos dice exactamente que versión tenemos que descargar.

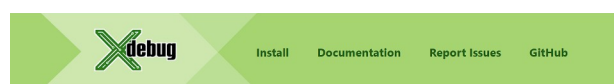
Entra a <https://xdebug.org/wizard> y pega todo el contenido de lo que devuelve la función phpinfo() (créate cualquier archivo .php que contenga esa función y ábrelo con el navegador).

PHP Version 8.0.28



System	Windows NT ALBERTO 10.0 build 22H2 (Windows 10) AMD64
Build Date	Feb 14 2023 12 10 00
Build System	Microsoft Windows Server 2019 Datacenter (10.0.17763)
Compiler	Visual C++ 19.29
Architecture	x64
Configure Command	ccrunt /nologo /e script configure.js --enable-snmpshim-build --enable-debug-pack --with-pdo-curl --with-pdo-firebird --enable-com-dotnet-shim --without-analyzer --with-pgsql
Server API	Apache 2.0 handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	no value
Loaded Configuration File	C:\xampp\php\php.ini
Scan this dir for additional ini files	(none)
Additional ini files parsed	(none)
PHP API	20200930
PHP Extension	20200930
Zend Extension	42020030
Zend Extension Build	AP04020030.TS.VS16
PHP Extension Build	AP02000300.TS.VS16
Debug Build	no
Thread Safety	enabled
Thread API	Windows Threads
Zend Symbol Handling	disabled

Puedes usar Ctrl+A para seleccionar toda la página y copiarla y pegarla luego en el analizador.



If you find Xdebug useful, please consider [supporting the project](#).

## Installation Wizard

This page helps you finding which file to download, and how to configure PHP to get Xdebug running. Please paste the **full** output of `phpinfo()` (either a copy & paste of the HTML version, the HTML source or `php -i` output) and submit the form to receive tailored download and installation instructions.

```
PMP logo
PMP Version 8.0.28
System Windows NT ALBERTO 10 x64 Build 22621 (Windows 10) AMD64
Build Date Feb 14 2023 12:10:00
Compiler Visual C++ v19.29
Build System Microsoft Windows Server 2019 Datacenter [8.0.17763]
Architecture x64
Configure Command script /nologo ;if exist config { :*-enable-anal-shared' *-enable-debug-pack' *-with-pdo-ocid%...%\..\..\\instanet\vsdk_shared' *-with-oci8-19s_...\..\..\instanet\vsdk_shared' *-enable-object-out-put; } else { echo can-not detect shared ; } -without-analyzer; *-with-gso-Server API Apache 2.0 Handler
Virtual Directory Support enabled
Configuration File (path) Pmp.ini no value
Loaded Configuration File C:\xampp\php\php.ini
Scan this dir for additional .ini files (none)
Additional .ini files present: (none)
PMP API 20200930
PMP Extension 20200930
Zend Extension 42020930
Zend Extension Build API20200930,TS,VS16
PMP Extension Build API20200930,TS,VS16
Debug Mode no
```

The information that you upload will not be stored. The script will only use a few regular expressions to analyse the output and provide you with instructions. You can see the code [here](#).

Pulsa el botón Analyse my phpinfo() output.

## Installation Wizard

### Summary

- Xdebug installed: no
- Server API: Apache 2.0 Handler
- Windows: yes
- Compiler: MS VS16
- Architecture: x64
- Zend Server: no
- PHP Version: 8.0.28
- Zend API nr: 420200930
- PHP API nr: 20200930
- Debug Build: no
- Thread Safe Build: yes
- OPcache Loaded: no
- Configuration File Path: no value
- Configuration File: C:\xampp\php\php.ini
- Extensions directory: C:\xampp\php\ext

### Instructions

1. Download [php\\_xdebug-3.2.2-8.0-vs16-x86\\_64.dll](#)
2. Move the downloaded file to C:\xampp\php\ext, and rename it to `php_xdebug.dll`
3. Update C:\xampp\php\php.ini and add the line:  
`zend_extension = xdebug`
4. Restart the Apache Webserver

Descarga el archivo con extensión .dll.

Posteriormente moveremos nuestro archivo descargado a la carpeta de extensiones de la instalación de PHP. En nuestro caso, dado que estamos usando XAMPP, la ruta donde deberemos mover el archivo .dll es: C:\xampp\php\ext

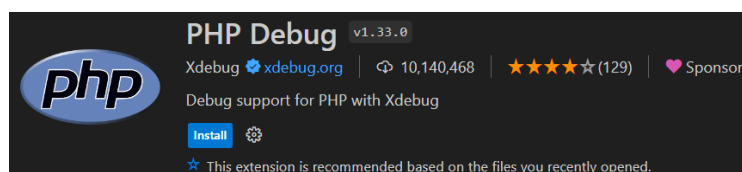
Con este paso ya tendremos instalado Xdebug. Ahora solo nos hará falta habilitarlo en el archivo de configuración de PHP (php.ini) el cuál está ubicado en C:\xampp\php.

Ahora debes modificar el fichero de configuración de PHP (php.ini) para activar la extensión. En XAMPP, modifica el fichero C:\xampp\php\php.ini y añade las siguientes líneas al final:

```
[xdebug]
zend_extension="C:\xampp\php\ext\php_xdebug-3.2.2-8.0-vs16-x86_64.dll"
; Opciones de configuración
xdebug.mode = debug
xdebug.start_with_request = yes
```

Recuerda reiniciar el servidor web siempre que modifiques el fichero php.ini para aplicar los cambios (Stop y Start en XAMPP).

Por último, instala la extensión PHP Debug en Visual Studio Code:

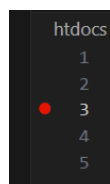


## 0.2. Utilizando herramientas de depuración

En este ejemplo, si lo ejecutamos se produce un bucle infinito:

```
<!-- ejemplo_debug.php -->
<?php
$persona = array("nombre" => "Lucía", "edad" => 1);
?>
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>La edad de <?php echo $persona['nombre']; ?></title>
</head>
<body>
<h1>Edad de <?php echo $persona['nombre']; ?></h1>
<?php
for ($i = 0; $persona['edad'] < 14; $i++) {
    $persona['edad'] *= $i;
}
$resultado = "<p>Lucía tiene " . $persona['edad'] . "
años.</p>";
echo $resultado;
?>
</body>
</html>
```

Para ver qué pasa en el bucle vamos a colocar un punto de parada dentro. Para ello hacemos clic a la izquierda del número de línea y se dibujará un punto rojo (para quitarlo realizamos la misma operación).



A continuación pulsamos en la pestaña con el icono del insecto (el bug) y en la parte superior izquierda le damos al símbolo play en verde para la opción **Listen for Xdebug**.



Ahora cuando intentemos ver la página con el navegador, la ejecución se parará y podremos ir ejecutándola paso a paso. En variables podemos monitorizar los valores de `$i` y `$persona` (incluso en WATCH podemos incluir `$persona['edad']` o alguna expresión más compleja).

Al final nos daremos cuenta de que la edad siempre es 0. Si hacemos que el bucle empiece en 1 corregiremos el programa.

## Actividad 3.0

Depura el archivo `debugme.php`:

```
<!-- debugme.php -->
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Page Title</title>
  </head>
  <body>
    <h1>Lista de números</h1>
    <?php
      $j=10; // Limite del bucle
      for ($i = 0; $i<$j; $j++) {
        echo "<li>" . $i;
        if ($i=7) {
          echo " (el número de la suerte)";
        }
        echo "</li>";
      }
    ?>
  </body>
</html>
```