

# MANUAL XDEBUG



Miguel Ángel Aranda García

## ¿Qué es Xdebug?

Es una extensión de PHP para hacer debug con herramientas de depuración tradicionales, desde el editor, tal como se hace en lenguajes de programación clásicos. Podemos encontrar esta extensión en su sitio web: <http://xdebug.org/>.

## ¿Que nos permite Xdebug?

Xdebug te permitirá no solo analizar el contenido de las variables, sino también realizar el seguimiento del flujo de ejecución, para saber qué es lo que realmente está ocurriendo cuando algo no funciona como se esperaba.

Tienes disponible información sobre todas las variables creadas en un momento dado de la ejecución de tu aplicación y puedes ver su contenido de manera expandida. Te ofrece también la posibilidad de marcar puntos de ruptura de tu código, donde el flujo del programa se detendrá para que puedas ver el estado de tus variables, o la traza de ejecución. Si lo deseas puedes ser tan minucioso como acompañar la ejecución del programa línea a línea.

## Instalación

### Prerrequisitos para la instalación.

#### Actualizar servidor ubuntu

Como siempre lo primero es hacer un “sudo apt update” y a continuación un “sudo apt upgrade”.

**“sudo apt update” y “sudo apt upgrade”**

#### Instalar php-xdebug:

Una vez hecho esto usaremos el comando “sudo apt install php-xdebug” (Si tienen una versión anterior a la actual el comando sería “sudo apt install php.version-xdebug”).

**“sudo apt install php-xdebug” o “sudo apt install php.version-xdebug”**

Y reiniciamos apache “sudo systemctl restart apache2”.

## Comprobar que está instalado.

Una vez reiniciado el servidor apache creamos un archivo llamado “index.php” que contendrá la línea de código “<?php phpinfo() ?>”

“<?php phpinfo() ?>”

Dentro de phpinfo() saldrá en diferentes partes:

1º Saldrá en : Additional .ini files parsed

Additional .ini files parsed	/etc/php/7.4/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.4/apache2/conf.d/10-opcache.ini, /etc/php/7.4/apache2/conf.d/10-pdo.ini, /etc/php/7.4/apache2/conf.d/20-calendar.ini, /etc/php/7.4/apache2/conf.d/20-ctype.ini, /etc/php/7.4/apache2/conf.d/20-exif.ini, /etc/php/7.4/apache2/conf.d/20-ff.ini, /etc/php/7.4/apache2/conf.d/20-fileinfo.ini, /etc/php/7.4/apache2/conf.d/20-ftp.ini, /etc/php/7.4/apache2/conf.d/20-gettext.ini, /etc/php/7.4/apache2/conf.d/20-iconv.ini, /etc/php/7.4/apache2/conf.d/20-json.ini, /etc/php/7.4/apache2/conf.d/20-mysqli.ini, /etc/php/7.4/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.4/apache2/conf.d/20-phar.ini, /etc/php/7.4/apache2/conf.d/20-posix.ini, /etc/php/7.4/apache2/conf.d/20-readline.ini, /etc/php/7.4/apache2/conf.d/20-shmop.ini, /etc/php/7.4/apache2/conf.d/20-sockets.ini, /etc/php/7.4/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.4/apache2/conf.d/20-sysvsem.ini, /etc/php/7.4/apache2/conf.d/20-sysvshm.ini, /etc/php/7.4/apache2/conf.d/20-tokenizer.ini, /etc/php/7.4/apache2/conf.d/20-xdebug.ini
------------------------------	---

2º Saldrá como variable global.

### xdebug

xdebug support	enabled
Version	2.9.2
<a href="#">Support Xdebug on Patreon, GitHub, or as a business</a>	

Justo debajo saldrán todas las directivas de xdebug y sus valores.

Directive	Local Value	Master Value
xdebug.auto_trace	Off	Off
xdebug.cli_color	0	0
xdebug.collect_assignments	Off	Off
xdebug.collect_includes	On	On

## Configuración

### zend\_extensions:

Tenemos que configurar el “zend\_extensions” dentro del archivo “php.ini”.

Sirve para poder conectarse a las capas inferiores del lenguaje de php e interpretar las llamadas para la depuración en el caso de la extensión de “xdebug”, también se usan para otras extensiones como “OPcache”.

Entramos en modo edición dentro de php.ini (antes hacer una copia de seguridad) y en la sección de modules de php agregamos la línea “[xdebug]” y debajo “zend\_extension = path/xdebug.so”.

Si no cambiamos la ruta de la instalación de la extensión de xdebug no necesitaremos poner el path de la extensión ya que php(./configure) buscará la ruta predeterminada de la extensión.

**“zend\_extension = xdebug.so”**

Después de guardar el archivo php.ini reiniciamos el servidor. Para comprobarlo usamos el comando “php -m” que nos mostrará todos los módulos instalados.

**“php -m”**

### Activar la conexión remota para debugging con xdebug:

xdebug tiene activado por defecto xdebug.remote\_host como localhost y desactivado.

xdebug.remote_enable	Off	Off
xdebug.remote_host	localhost	localhost

Para activarlo tenemos volver al fichero php.ini donde debajo del comando “zend\_extension = xdebug.so” escribiremos las siguientes líneas ( Para que esté ordenado y sea fácil de encontrar),

“xdebug.remote\_enable=1” y “xdebug.remote\_port=( El puerto de Xdebug por donde escucha) ( El puerto por defecto es 9000)”.

**“xdebug.remote\_enable=1”**

**“xdebug.remote\_port = 9000”**

Para comprobar que ha funcionado después de volver a guardar el archivo php.ini, reiniciamos el servidor y volvemos a cargar phpinfo(). Que no se nos olvide abrir el puerto 9000 si tenemos el firewall activado con “sudo ufw allow 9000”.

Nos dirigimos a la línea de xdebug\_remote\_enable y debe poner “On”.

xdebug.remote_enable	On	On
----------------------	----	----

### Asignar una ip para la conexión remota para debugging con xdebug:

Seguimos usando el archivo php.ini y escribimos justo debajo de la línea de xdebug anterior

“xdebug.remote\_host = dirección ip”, si queremos poner múltiples ip la separaremos mediante el signo “,”.

Ejemplo:

xdebug.remote\_host = ip1, ip2, ip3...

**“xdebug.remote\_host = dirección ip cliente”**

### Activar el conector de retorno de xdebug:

Si activamos el comando “xdebug.remote\_connect\_back = 1”, entonces el comando “xdebug.remote\_host” cambia su funcionalidad y en este caso deja de filtrar las direcciones ip de los clientes, para asignarle la dirección ip del servidor para que todos los clientes puedan entrar.

**“xdebug.remote\_connect\_back = 1”**

**“xdebug.remote\_host = dirección ip servidor”**

### Asignar la ide key de xdebug:

Si, otra vez volvemos al archivo php.ini y escribimos justo debajo de la línea de xdebug anterior

“xdebug.idekey = “nombre de ide que vamos a usar (preferible usar el id por defecto asignado en el ide)”, si queremos poner múltiples ip la separaremos mediante el signo “;”.

**“xdebug.idekey= “netbeans-xdebug””**

El resto de parámetros son opcionales, para más info:

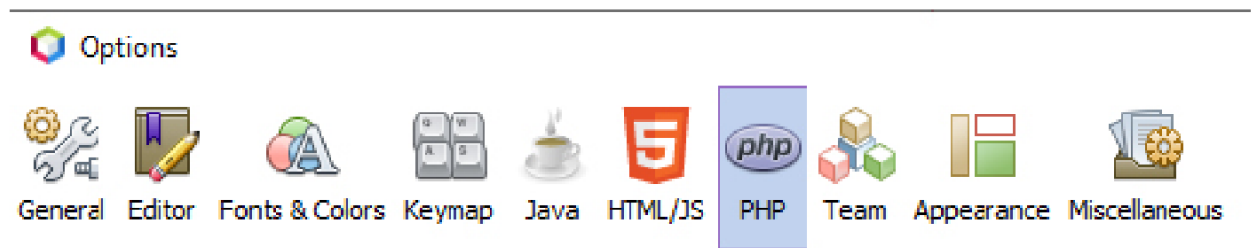
<https://xdebug.org/docs/remote> (No es muy buena guía, pero es oficial).

## Configuración en NetBeans

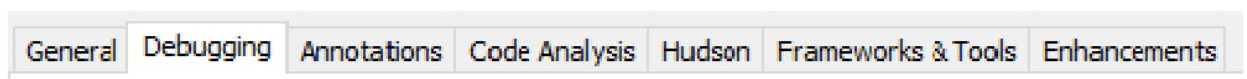
Es bastante sencillo de configurar.

Vamos a la pestaña de tools en el nav de Netbeans y en el desplegable a la opción.

Dentro de option vamos a PHP.



Por último dentro de php a la pestaña de “Debugging”.



Una vez en esta pestaña veremos los valores por defecto que asigna Netbeans, si hemos seguido las instrucciones anteriores todo estará bien y no tendremos que tocar nada, solo comprobamos que son correctos.

Debugger Port:

Session ID:

Maximum Data Length:

☒ Stop at First Line

☐ Watches and Balloon Evaluation

Maximum Depth of Structures:

Maximum Number of Children:

☐ Show Requested URLs

☐ Show Debugger Console

Do not forget to set `output_buffering = Off` in your `php.ini` file.

## Uno en NetBeans

### Uso de debugger en Netbeans 12.

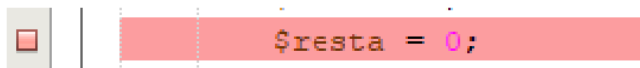
#### Iniciar debug sobre un proyecto

#### Netbeans:

En Netbeans seleccionamos el proyecto en el que deseamos usar el debugger.

En mi caso voy a usar un ejemplo de formulario de php.

En la línea que deseemos hacemos un breakpoint, yo usare justo la línea donde empieza `<?php`.

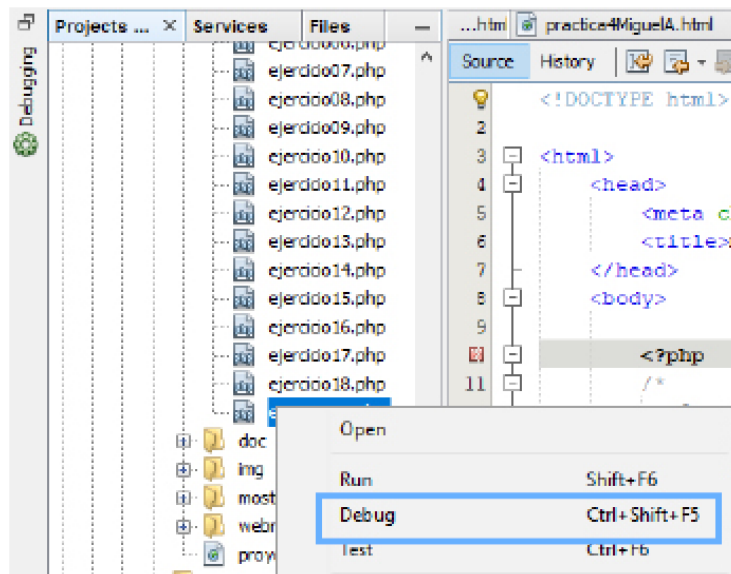


El cuadrito rojo y el fondo en rojo indican cual es el breakpoint, si ponemos el puntero encima nos lo dice con un comentario.

Para iniciar la ejecución con debugger tenemos diferentes formas:

1º Pulsando las teclas ("control + shift + F5").

2º Desde la pestaña de Proyectos, pulsando botón derecho sobre nuestro proyecto y la opción "Debug".

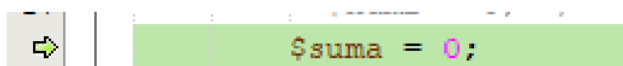


## Trabajar con debug sobre un proyecto Netbeans:

Una vez iniciado el debug pasan tres cosas:

- 1º Se abrirá una ventana en el navegador que elegimos por defecto.
- 2º En Netbeans se colorea de verde la línea por donde está el cursor de la ejecución.
- 3º Se nos abrirá una pestaña dentro de Netbeans donde podremos ver las variables, callstack y los breakpoints.

### 2º Posición del cursor.



### 3º Pestaña con variables, callstack y breakpoints.

Variables	Call Stack	Breakpoints	
<div> <div>Name</div> <div>Type</div> </div>			
Superglobals			
\$COOKIE			array[1]
\$ENV			array[0]
\$FILES			array[0]
\$GET			array[1]
\$POST			array[0]
\$REQUEST			array[1]
\$SERVER			array[31]


Dentro de esta pestaña habrá 3 elementos principales:

#### 1º Variables:

Dentro de la pestaña variables podremos ver paso por paso de la ejecución que variables se van asignando a ellas o vaciándose.






## 2º Call stack:

En la pestaña de call stack podremos ver cual es la posición del puntero dentro del documento que esté ejecutándose, en caso de llamada a archivos externos también seguirá el recorrido con el path absoluto.

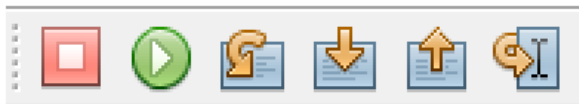
Variables	Call Stack X	Breakpoints
Name		
 proyectoDWES/prcyectoTema3/codgoPHP/ejercicio19.php.{main}: 19		

## 3º BreakPoints:

En esta pestaña podremos ver el listado de los breakpoint asignados en todo el recorrido del proyecto.

Variables	Call Stack	Breakpoints X
Name		
  	<input checked="" type="checkbox"/>  ejercicio19.php:19	
	<input checked="" type="checkbox"/>  ejercicio19.php:22	

## Nav de botones para debug.



Estos son los botones de navegación dentro del recorrido del debug.



Detiene la ejecución del debug.



Continúa la ejecución hasta el final de la ejecución.



Salta a la siguiente etapa del proyecto.





Salta a la siguiente instrucción.



Salta a la instrucción anterior.



Salta al siguiente breakpoint.