

Tutorial Xdebug

Despliegue de aplicaciones web (DAW)

2º DESARROLLO DE APLICACIONES WEB
13/10/2020

NEREA ÁLVAREZ JUSTEL

Contenido

INTRODUCCIÓN	2
DESARROLLO DE LOS CONTENIDOS	3
1. Documenta la instalación de Xdebug en el servidor.	3
2. Configuración del Xdebug en el IDE	4
3. Xdebug.ini	9
CONCLUSIÓN.....	11
BIBLIOGRAFÍA	12

INTRODUCCIÓN

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. Utiliza **DBGp** que es protocolo de depuración simple que se usa para comunicar el motor de depuración propio de php con un cliente, normalmente un IDE.

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.

DESARROLLO DE LOS CONTENIDOS

1. Documenta la instalación de Xdebug en el servidor.

En el servidor creado instalaremos Xdebug, con el siguiente comando:

sudo apt install php-xdebug

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  php-xdebug
0 upgraded, 1 newly installed, 0 to remove and 20 not upgraded.
Need to get 473 kB of archives.
After this operation, 2174 kB of additional disk space will be used.
Get:1 http://es.archive.ubuntu.com/ubuntu focal/universe amd64 php-xdebug amd64 2.9.2+2.8.1+2.5.5-1build1 [473 kB]
Fetched 473 kB in 1s (544 kB/s)
Selecting previously unselected package php-xdebug.
(Reading database ... 109020 files and directories currently installed.)
Preparing to unpack .../php-xdebug_2.9.2+2.8.1+2.5.5-1build1_amd64.deb ...
Unpacking php-xdebug (2.9.2+2.8.1+2.5.5-1build1) ...
Setting up php-xdebug (2.9.2+2.8.1+2.5.5-1build1) ...
Processing triggers for libapache2-mod-php7.4 (7.4.3-4ubuntu2.2) ...
Processing triggers for php7.4-cli (7.4.3-4ubuntu2.2) ...
```

Tras la instalación realizaremos un listado de los módulos, el comando para esto será:

php -m | grep xdebug

```
miadmin@NAJUSED:~$ php -m | grep xdebug
xdebug
```

Ahora procederemos a modificar el fichero **xdebug.ini** el cual se encuentra en el directorio **/etc/php/7.4/mods-available** y añadiremos las siguientes líneas:

```
zend_extension=xdebug.so
xdebug.show_error_trace = 1
xdebug.remote_enable = on
xdebug.remote_handler = dbgp
xdebug.remote_host = localhost
xdebug.remote_port = 9000
xdebug.remote_connect_back=1
xdebug.idkey=netbeans-xdebug
```

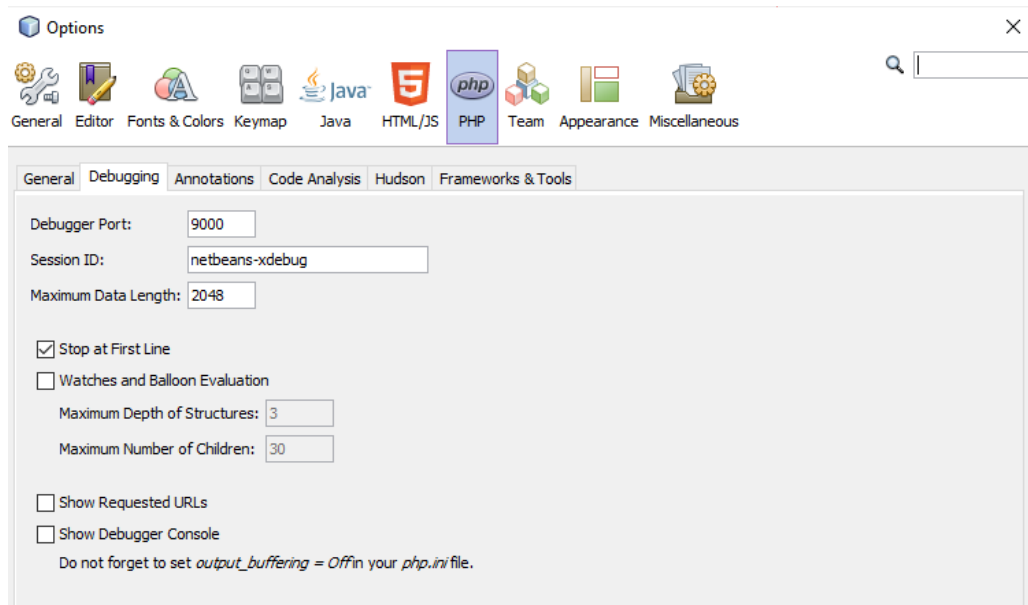
Para comprobar el funcionamiento de la información añadida, visualizaremos **phpinfo()**, buscaremos las variables Xdebug cambiadas anteriormente en el fichero **xdebug.ini**.

xdebug.remote_cookie_expire_time	3600	3600
xdebug.remote_enable	On	On
xdebug.remote_host	localhost	localhost
xdebug.remote_log	no value	no value
xdebug.remote_log_level	7	7
xdebug.remote_mode	req	req
xdebug.remote_port	9000	9000
xdebug.remote_timeout	200	200
xdebug.scream	Off	Off

2. Configuración del Xdebug en el IDE

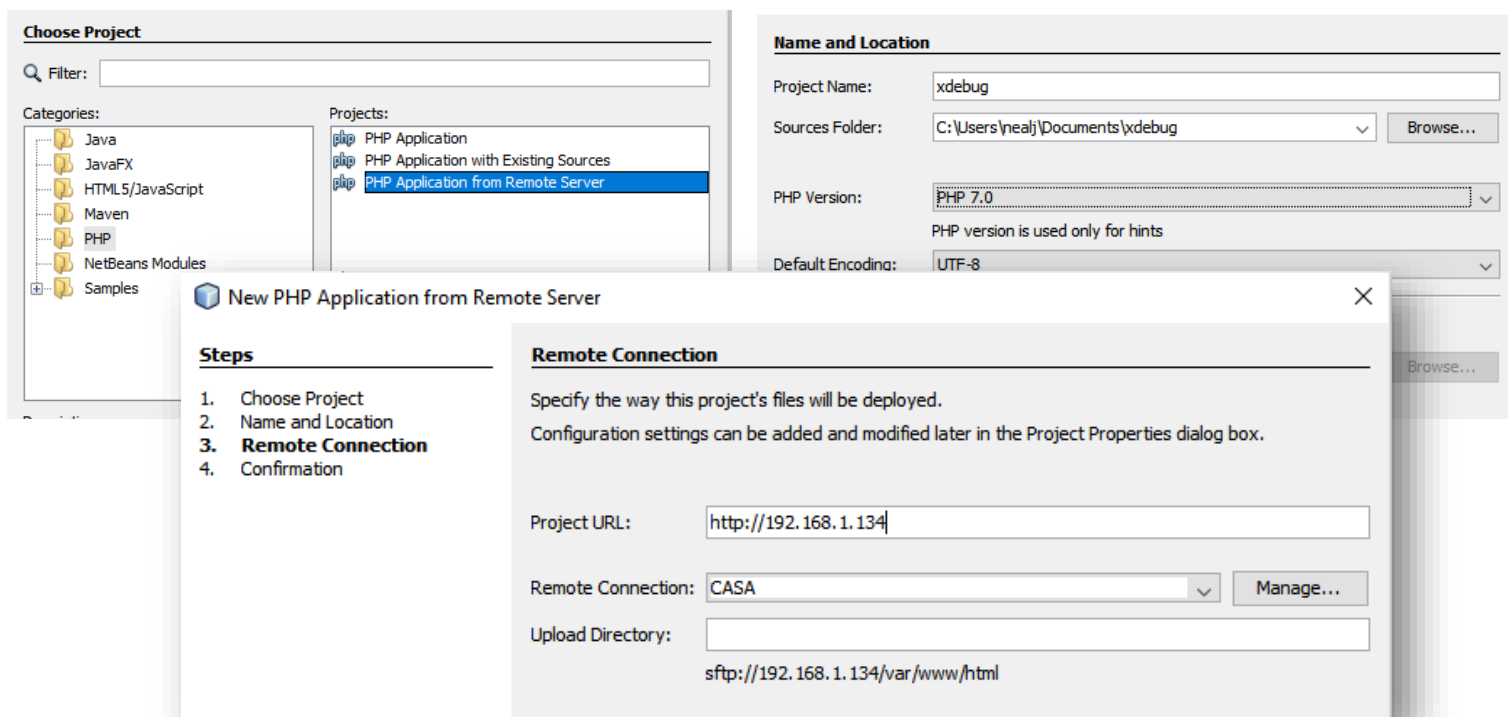
Abriremos el IDE, en nuestro caso Netbeans.

Para empezar iremos a Tools -> Options -> PHP -> Debugging

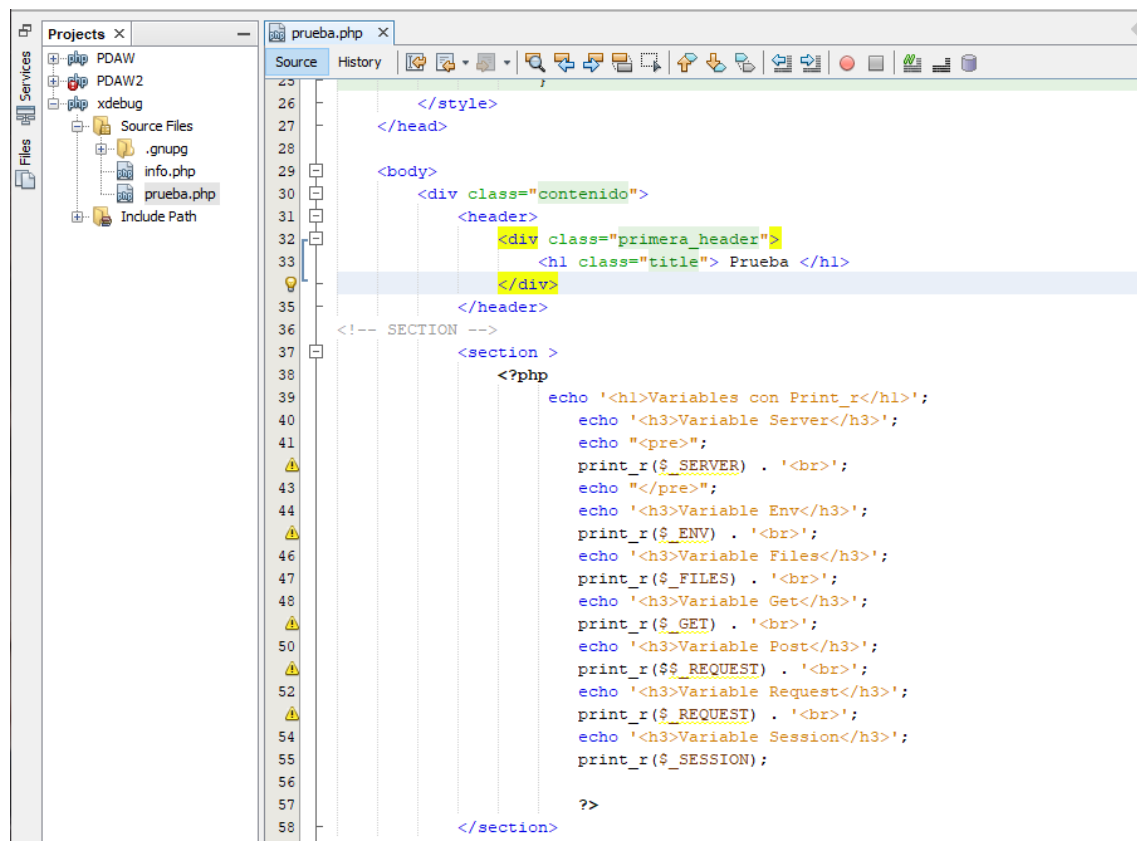


Comprobaremos que los valores que aparecen en el Netbeans son iguales a los introducidos en el fichero **xdebug.ini**.

Tras esta comprobación, crearemos un proyecto PHP, conectado a nuestro servidor.



Crearemos un fichero con extensión .php, el cual contendrá esto:



```

25
26     </style>
27 </head>
28
29 <body>
30     <div class="contenido">
31         <header>
32             <div class="primera_header">
33                 <h1 class="title"> Prueba </h1>
34             </div>
35         </header>
36         <!-- SECTION -->
37         <section>
38             <?php
39                 echo '<h1>Variables con Print_r</h1>';
40                 echo '<h3>Variable Server</h3>';
41                 echo "<pre>";
42                 print_r($_SERVER) . '<br>';
43                 echo "</pre>";
44                 echo '<h3>Variable Env</h3>';
45                 print_r($_ENV) . '<br>';
46                 echo '<h3>Variable Files</h3>';
47                 print_r($_FILES) . '<br>';
48                 echo '<h3>Variable Get</h3>';
49                 print_r($_GET) . '<br>';
50                 echo '<h3>Variable Post</h3>';
51                 print_r($_POST) . '<br>';
52                 echo '<h3>Variable Request</h3>';
53                 print_r($_REQUEST) . '<br>';
54                 echo '<h3>Variable Session</h3>';
55                 print_r($_SESSION);
56             <?>
57         </section>
58     </div>
59 </body>
60 </html>

```

Visualizaremos en fichero mediante un navegador, este es el resultado:



Prueba

Variables con Print_r

Variable Server

```

Array
(
    [HTTP_HOST] => 192.168.1.134
    [HTTP_CONNECTION] => keep-alive
    [HTTP_UPGRADE_INSECURE_REQUESTS] => 1
    [HTTP_USER_AGENT] => Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.75 Safari/537.36
    [HTTP_ACCEPT] => text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
    [HTTP_ACCEPT_ENCODING] => gzip, deflate
    [HTTP_ACCEPT_LANGUAGE] => es-ES;q=0.9
    [PATH] => /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
    [SERVER_SIGNATURE] =>
    Apache/2.4.29 (Ubuntu) Server at 192.168.1.134 Port 80

    [SERVER_SOFTWARE] => Apache/2.4.29 (Ubuntu)
    [SERVER_NAME] => 192.168.1.134
    [SERVER_ADDR] => 192.168.1.134
    [SERVER_PORT] => 80
    [REMOTE_ADDR] => 192.168.1.234
    [DOCUMENT_ROOT] => /var/www/html
    [REQUEST_SCHEME] => http
    [CONTEXT_PREFIX] =>
    [CONTEXT_DOCUMENT_ROOT] => /var/www/html
    [SERVER_ADMIN] => webmaster@localhost
    [SCRIPT_FILENAME] => /var/www/html/prueba.php
    [REMOTE_PORT] => 57934
    [GATEWAY_INTERFACE] => CGI/1.1
    [REQUEST_METHOD] => GET
    [QUERY_STRING] =>
    [REQUEST_URI] => /prueba.php
    [SCRIPT_NAME] => /prueba.php
    [PHP_SELF] => /prueba.php
    [REQUEST_TIME_FLOAT] => 1602606938.827
    [REQUEST_TIME] => 1602606938
)

```

Variable Request

```

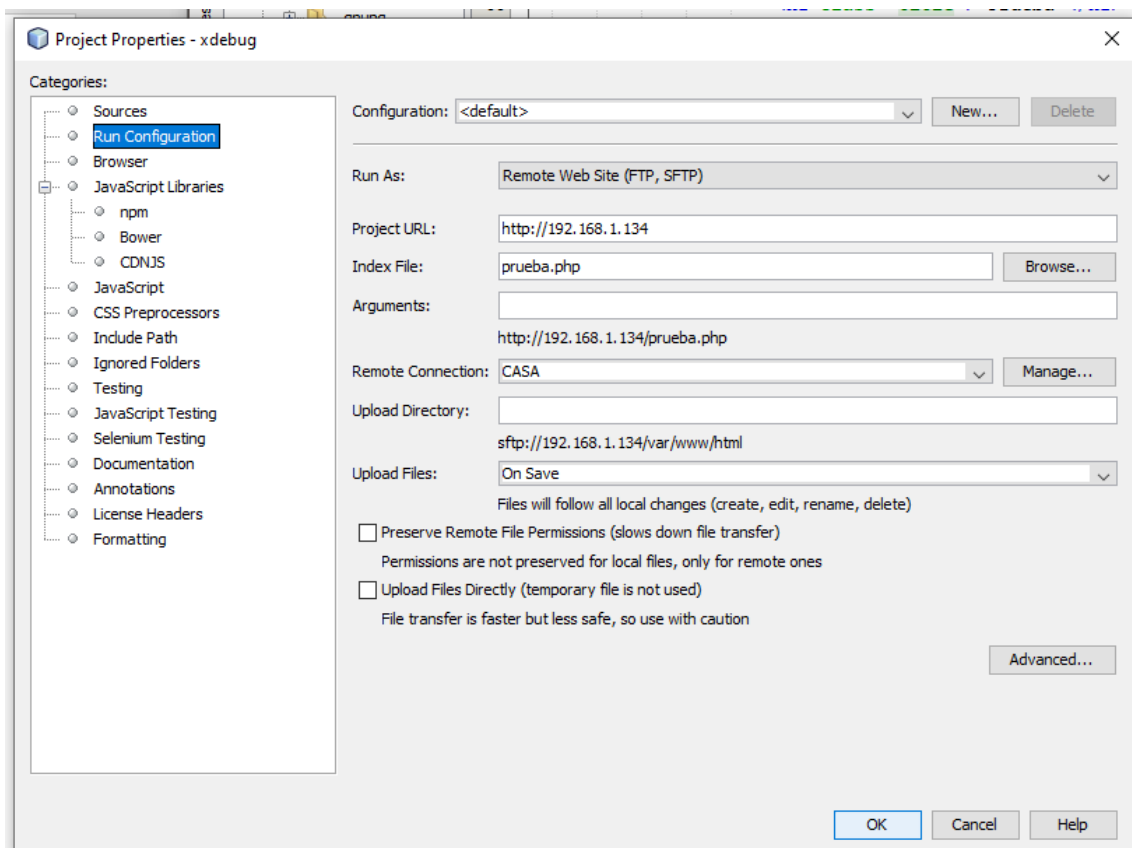
[REQUEST_METHOD] => GET
[QUERY_STRING] =>
[REQUEST_URI] => /prueba.php
[SCRIPT_NAME] => /prueba.php
[PHP_SELF] => /prueba.php
[REQUEST_TIME_FLOAT] => 1602606938.827
[REQUEST_TIME] => 1602606938

```

Para comenzar la depuración pulsaremos el botón aquí indicado:



Se abrirá la ventana de configuración del proyecto y tendremos que cambiar el **index file** por el nombre de nuestro archivo, este caso **prueba.php**, para que al pulsar en el botón de ejecución sea este archivo el que se ejecute.



Después de enlazar el archivo y pulsar el inicio de la ejecución.

Podremos controlar la depuración utilizando los siguientes botones.



Colocaremos un Breakpoint en la línea 50, pulsaremos la ejecución línea a línea para observar los valores.

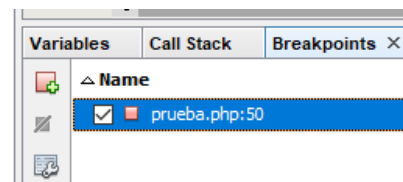
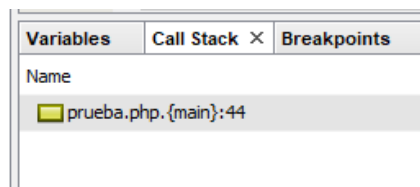
También podremos realizar la ejecución del ejercicio directamente pulsando el play.

```

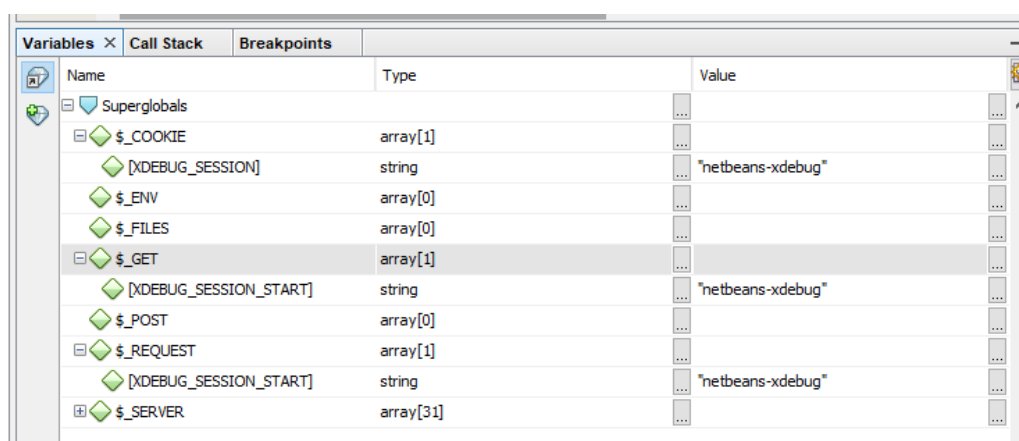
29 <body>
30 <div class="contenido">
31 <header>
32 <div class="primera_header">
33 <h1 class="title"> Prueba </h1>
34 </div>
35 </header>
36 <!-- SECTION -->
37 <section>
38 <?php
39     echo ' <h1>Variables con Print_r</h1>';
40     echo ' <h3>Variable Server</h3>';
41     echo "<pre>";
42     print_r($_SERVER) . ' <br>';
43     echo "</pre>";
44     echo ' <h3>Variable Env</h3>';
45     print_r($_ENV) . ' <br>';
46     echo ' <h3>Variable Files</h3>';
47     print_r($_FILES) . ' <br>';
48     echo ' <h3>Variable Get</h3>';
49     print_r($_GET) . ' <br>';
50     echo ' <h3>Variable Post</h3>';
51     print_r($_REQUEST) . ' <br>';
52     echo ' <h3>Variable Request</h3>';
53     print_r($_REQUEST) . ' <br>';
54     echo ' <h3>Variable Session</h3>';
55     print_r($_SESSION);
56
57     ?>
58 </section>

```

La línea verde es la que se está ejecutando, la información se puede ver en la pestaña Call Stack. Y la línea roja es la que contiene el Breakpoint, la información se puede ver en la pestaña Breakpoint.



La pestaña Variables, indica la información que contienen las variables del ejercicio.



Aquí visualizamos que se esta ejecutando el ejercicio con Xdebug.

Prueba

Variables con Print_r

Variable Server

```
Array
(
    [HTTP_HOST] => 192.168.1.134
    [HTTP_CONNECTION] => keep-alive
    [HTTP_UPGRADE_INSECURE_REQUESTS] => 1
    [HTTP_USER_AGENT] => Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.75 Safari/537.36
    [HTTP_ACCEPT] => text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
    [HTTP_ACCEPT_ENCODING] => gzip, deflate
    [HTTP_ACCEPT_LANGUAGE] => es-ES,es;q=0.9
    [PATH] => /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin
    [SERVER_SIGNATURE] =>
    Apache/2.4.29 (Ubuntu) Server at 192.168.1.134 Port 80

    [SERVER_SOFTWARE] => Apache/2.4.29 (Ubuntu)
    [SERVER_NAME] => 192.168.1.134
    [SERVER_ADDR] => 192.168.1.134
    [SERVER_PORT] => 80
    [REMOTE_ADDR] => 192.168.1.234
    [DOCUMENT_ROOT] => /var/www/html
    [REQUEST_SCHEME] => http
    [CONTEXT_PREFIX] =>
    [CONTEXT_DOCUMENT_ROOT] => /var/www/html
    [SERVER_ADMIN] => webmaster@localhost
    [SCRIPT_FILENAME] => /var/www/html/prueba.php
    [REMOTE_PORT] => 57977
    [GATEWAY_INTERFACE] => CGI/1.1

    [REQUEST_METHOD] => GET
    [QUERY_STRING] => XDEBUG_SESSION_START=netbeans-xdebug
    [REQUEST_URI] => /prueba.php?XDEBUG_SESSION_START=netbeans-xdebug
    [SCRIPT_NAME] => /prueba.php
    [PHP_SELF] => /prueba.php
    [REQUEST_TIME_FLOAT] => 1602607148.427
    [REQUEST_TIME] => 1602607148
)
```

3. Xdebug.ini

Dentro del archivo xdebug.ini tendremos que añadir algunas líneas de código. A continuación, veremos dichas líneas, así como su función.

xdebug.show_error_trace = 1 : Introducido en Xdebug a partir de la versión 2.4.

Cuando esta configuración se establece en 1, Xdebug mostrará un seguimiento de la pila de errores siempre que se genere un error, incluso si este error se detecta realmente.

xdebug.remote_enable = on : Este conmutador controla si Xdebug debe intentar ponerse en contacto con un cliente de depuración que está escuchando en el host y el puerto según lo establecido con la configuración xdebug.remote_host y xdebug.remote_port . Si no se puede establecer una conexión, el script simplemente continuará como si esta configuración fuera 0.

xdebug.remote_handler = dbgp : Introducido en Xdebug a partir de la versión 2.9.

Solo puede ser 'dbgp' para representar el protocolo del depurador. El protocolo DBGp es el único protocolo compatible. Para más información: <https://xdebug.org/docs/dbgp>.

xdebug.remote_host = localhost : Selecciona el host donde se está ejecutando el cliente de depuración, puede usar un nombre de host, una dirección IP. Esta configuración se ignora si xdebug.remote_connect_back está habilitado.

xdebug.remote_port = 9000 : El puerto al que Xdebug intenta conectarse en el host remoto. El puerto 9000 es el predeterminado tanto para Xdebug como para el cliente de depuración de línea de comandos. Como muchos clientes utilizan este número de puerto, es mejor dejar esta configuración sin cambios.

xdebug.remote_connect_back = 1 : Introducido en Xdebug a partir de la versión 2.1.

Si está habilitado, la configuración `xdebug.remote_host` se ignora y Xdebug intentará conectarse con el cliente que realizó la solicitud HTTP.

Tenga en cuenta que no hay ningún filtro disponible y cualquiera que pueda conectarse al servidor web podrá iniciar una sesión de depuración, incluso si su dirección no coincide con `xdebug.remote_host`.

xdebug.idkey = netbeans-xdebug : Controla qué IDE Key Xdebug debe pasar al controlador del depurador DBGp. El valor predeterminado se basa en la configuración del entorno. Primero se consulta la configuración del entorno `DBGP_IDEKEY`, luego `USER` y como último `USERNAME`. El valor predeterminado se establece en la primera variable de entorno que se encuentra. Si no se pudo encontrar ninguno, la configuración tiene por defecto `"`. Si se establece esta configuración, siempre anula las variables de entorno.

CONCLUSIÓN

Con este estudio del Xdebug hemos aprendido a realizar la instalación y posterior configuración de este.

También hemos realizado varias pruebas de uso, utilizando distintos códigos con distinta complejidad.

La comprobación ha sido realizada ejecutando el código como un ejercicio normal y ejecutando el Xdebug desde el Netbeans.

Hemos hecho un estudio de las distintas opciones de configuración disponibles en Xdebug.

BIBLIOGRAFÍA

Xdebug

<https://xdebug.org/>

Videotutorial del uso de Xdebug

<https://www.youtube.com/watch?v=4Vx8TnNdQ6M>

Información útil

<https://desarrolloweb.com/articulos/que-es-instalar-configurar-xdebug.html>

<https://styde.net/como-hacer-debug-usando-xdebug-phpstorm-y-homestead/>

https://xdebug.org/docs/all_settings