Server-side
Web Development

Unit 03. Debugging PHP

Fidel Oltra, Ricardo Sánchez

IES Jaume II El Just
Tavernes de la Valldigna
Departament d'Informàtica
Curs 2023-24

Debugging is a very important part of developing. In this tutorial we will learn how to debug PHP applications.

1 Install Xdebug on Xubuntu

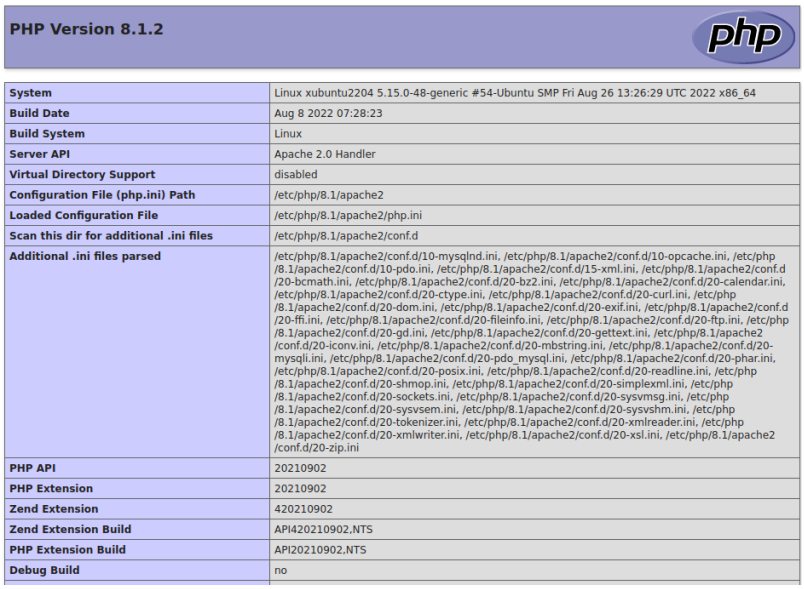
The first step is to install the **Xdebug** debugger in Xubuntu.

We have to test whether Xdebug for PHP is installed and the steps required to install and configure the most recent version of Xdebug.

Write a simple program to print the output of **phpinfo()** function as shown below:

```php
# index.php
<?php
echo phpinfo();
```

The output should be similar to:



PHP Version 8.1.2	
System	Linux xubuntu2204 5.15.0-48-generic #54-Ubuntu SMP Fri Aug 26 13:26:29 UTC 2022 x86_64
Build Date	Aug 8 2022 07:28:23
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.1/apache2
Loaded Configuration File	/etc/php/8.1/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.1/apache2/conf.d
Additional .ini files parsed	/etc/php/8.1/apache2/conf.d/10-mysqlnd.ini, /etc/php/8.1/apache2/conf.d/10-opcache.ini, /etc/php/8.1/apache2/conf.d/10-pdo.ini, /etc/php/8.1/apache2/conf.d/15-xml.ini, /etc/php/8.1/apache2/conf.d/20-bcmath.ini, /etc/php/8.1/apache2/conf.d/20-bz2.ini, /etc/php/8.1/apache2/conf.d/20-calendar.ini, /etc/php/8.1/apache2/conf.d/20-ctype.ini, /etc/php/8.1/apache2/conf.d/20-curl.ini, /etc/php/8.1/apache2/conf.d/20-dom.ini, /etc/php/8.1/apache2/conf.d/20-exif.ini, /etc/php/8.1/apache2/conf.d/20-ffi.ini, /etc/php/8.1/apache2/conf.d/20-fileinfo.ini, /etc/php/8.1/apache2/conf.d/20-ftp.ini, /etc/php/8.1/apache2/conf.d/20-gd.ini, /etc/php/8.1/apache2/conf.d/20-gettext.ini, /etc/php/8.1/apache2/conf.d/20-iconv.ini, /etc/php/8.1/apache2/conf.d/20-mbstring.ini, /etc/php/8.1/apache2/conf.d/20-mysqli.ini, /etc/php/8.1/apache2/conf.d/20-pdo_mysql.ini, /etc/php/8.1/apache2/conf.d/20-phar.ini, /etc/php/8.1/apache2/conf.d/20-posix.ini, /etc/php/8.1/apache2/conf.d/20-readline.ini, /etc/php/8.1/apache2/conf.d/20-shmop.ini, /etc/php/8.1/apache2/conf.d/20-simplexml.ini, /etc/php/8.1/apache2/conf.d/20-sockets.ini, /etc/php/8.1/apache2/conf.d/20-sysvmsg.ini, /etc/php/8.1/apache2/conf.d/20-sysvsem.ini, /etc/php/8.1/apache2/conf.d/20-sysvshm.ini, /etc/php/8.1/apache2/conf.d/20-tokenizer.ini, /etc/php/8.1/apache2/conf.d/20-xmlreader.ini, /etc/php/8.1/apache2/conf.d/20-xmlwriter.ini, /etc/php/8.1/apache2/conf.d/20-xsl.ini, /etc/php/8.1/apache2/conf.d/20-zip.ini
PHP API	20210902
PHP Extension	20210902
Zend Extension	420210902
Zend Extension Build	API420210902,NTS
PHP Extension Build	API20210902,NTS
Debug Build	no

Figure 1: phpinfo() output

Now, copy the content and paste it within the input box of Xdebug Installation Wizard:

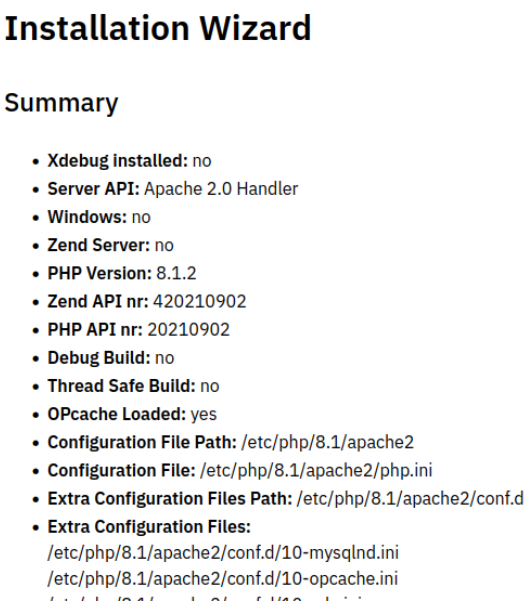


Figure 2: Xdebug installation wizard

Click on the **Analyse my phpinfo() output** button to start analyzing it. It will show whether Xdebug is installed and also shows the steps to install or update the most recent version of Xdebug for PHP:

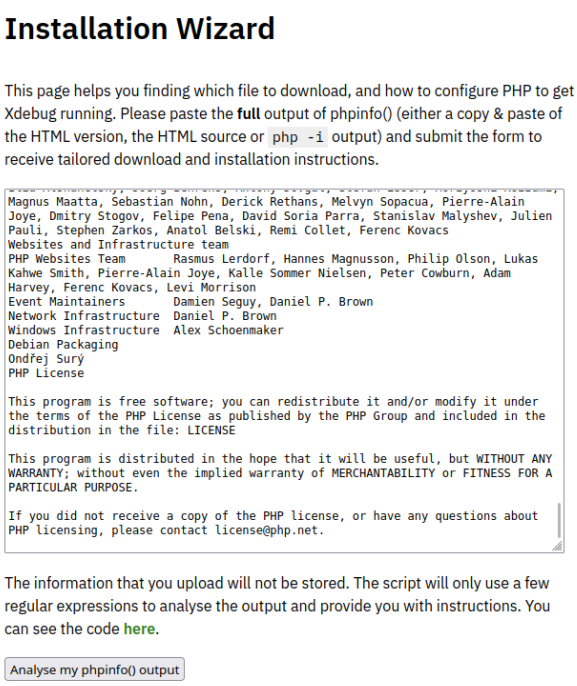


Figure 3: Output of the Xdebug installation wizard

Scroll down to the instructions section and follow the steps. In our case, the steps are:

1. Download xdebug-3.1.5.tgz
2. Install the pre-requisites for compiling PHP extensions. On your Ubuntu system, install them with: `apt-get install php-dev autoconf automake`
3. Unpack the downloaded file with `tar -xvzf xdebug-3.1.5.tgz`
4. Run: `cd xdebug-3.1.5`
5. Run: `phpize`.

   As part of its output it should show:

   ```
   Configuring for:
   ...
   Zend Module Api No:      20210902
   Zend Extension Api No:   420210902
   ```

   If it does not, you are using the wrong phpize. Please follow this FAQ entry and skip the next step.
6. Run: `./configure`
7. Run: `make`

8. Run: `cp modules/xdebug.so /usr/lib/php/20210902`
9. Create `/etc/php/8.1/apache2/conf.d/99-xdebug.ini` and add the line: `zend_extension = xdebug`
10. Please also update php`.ini` files in adjacent directories, as your system seems to be configured with a separate php`.ini` file for the web server and command line. (Nothing to do in this step)
11. Restart the Apache Webserver: `sudo service apache2 restart`

Do the steps of your wizard output, but at the step 9 add the following lines to the file:

```
[xdebug]
xdebug.mode=develop,debug
xdebug.discover_client_host=1
xdebug.client_port = 9003
xdebug.start_with_request=yes
```

Reload the test page and scroll down until you find the Xdebug configuration:

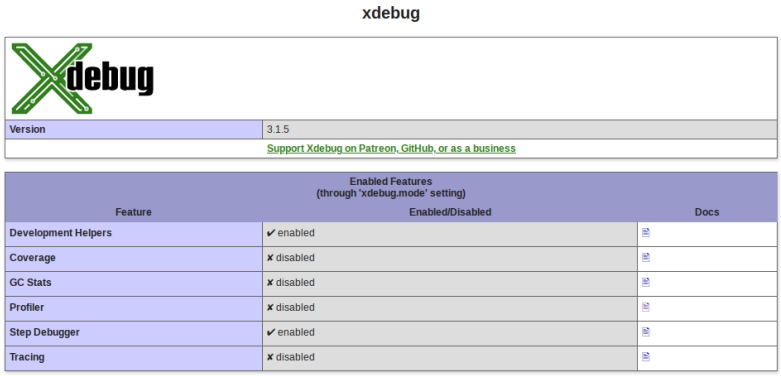


Figure 4: Xdebug configuration in phpinfo()

> An alternative way to install xdebug in Ubuntu OS is via the repositories:
>
> `sudo apt install php-xdebug`
>
> It's easier but it generally installs an older version. If you choose this way, you have to do step 9 searching for the `xx-xdebug.ini` file in the `/etc/php/8.1/apache2/conf.d` folder.

2 Install Xdebug on Windows

The steps are the same as in the Linux installation, but adjusting the paths to your system. Use the wizard to help you with the process.

3 Debugging in VS Code

If you have already installed the **PHP Tools** extension, it comes with a debugging tool. Otherwise, you can install the **PHP Debug** extension of **Xdebug**.

Create a script to check the debugging process, something like:

```php
<?php
$array = [0,5,10,15,20];

foreach ($array as $value) {
    $v = $value;
    echo "$v <br>";
}
```

Save it as `testdebug.php`.

We need to configure the Visual Studio Code to debug our code. Click the Run Icon on the Activity Bar on the left side to start configuring for xdebug. Now, click the create a **launch.json** file to configure the project for debugging. It will create the next file:

.vscode > {} launch.json > Launch Targets > {} Listen for Xdebug

```jsonc
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Launch built-in server and debug",
            "type": "php",
            "request": "launch",
            "runtimeArgs": [
                "-S",
                "localhost:9003",
                "-t",
                "."
            ],
            "port": 9003,
            "serverReadyAction": {
                "action": "openExternally"
            }
        },
        {
            "name": "Debug current script in console",
            "type": "php",
            "request": "launch",
            "program": "${file}",
            "cwd": "${fileDirname}",
            "externalConsole": false,
            "port": 9003
        },
```

Figure 5: launch.json for PHP Debug

Go to the 3rd element with the name “Listen for Xdebug” and add the lines:

```
"pathMappings": {
    "/var/www/html": "${workspaceRoot}"
}
```

> If you are using Windows or XAMMP, adjust the *pathMappings* value to your server root.

Don't forget the comma before `pathMappings`!:

```
{
    "name": "Listen for Xdebug",
    "type": "php",
    "request": "launch",
    "port": 9003,
    "pathMappings": {
        "/var/www/html": "${workspaceRoot}"
    }
}
```

Figure 6: launch.json for PHP Debug, changes

Now, add a breakpoint in the line 5 (`$v = $value`) of your `testdebug.php` file (by clicking on the left of the number) and start the debugger with the configuration "Listen for Xdebug":

Figure 7: Start debugging in VSCode

Go to your browser and type `localhost/testdebug.php`. The debugger bar on VSCode should now show the debugger options (continue, step over, etc). You could also see the variables values on the left panel:

You can also start the debugger with the "Debug current script in console" configuration. It will show the output step by step in the VSCode console.

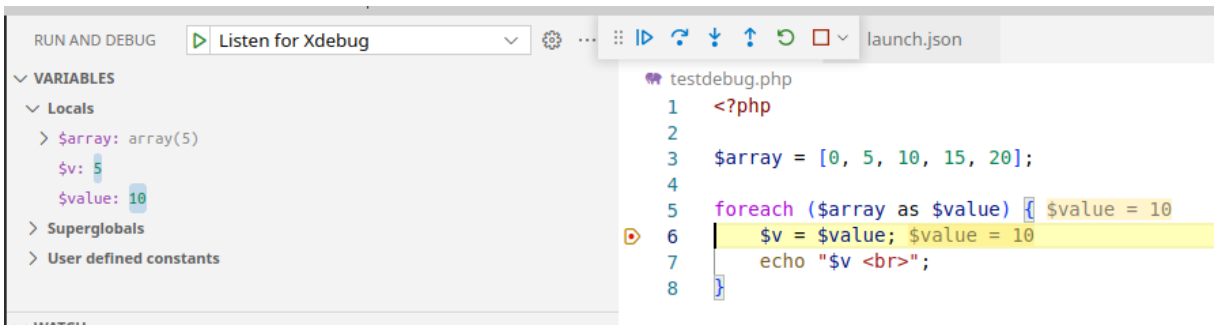


Figure 8: Debugging in VSCode

4 Debugging in PhpStorm

If we have installed Xdebug already, make a new project and copy or write the content of the previous `testdebug.php` script.

Create a new breakpoint in the line 5.

To start debugging, click on the Debug button. It will open the debug console where you can run the program step by step from the breakpoint you've created.

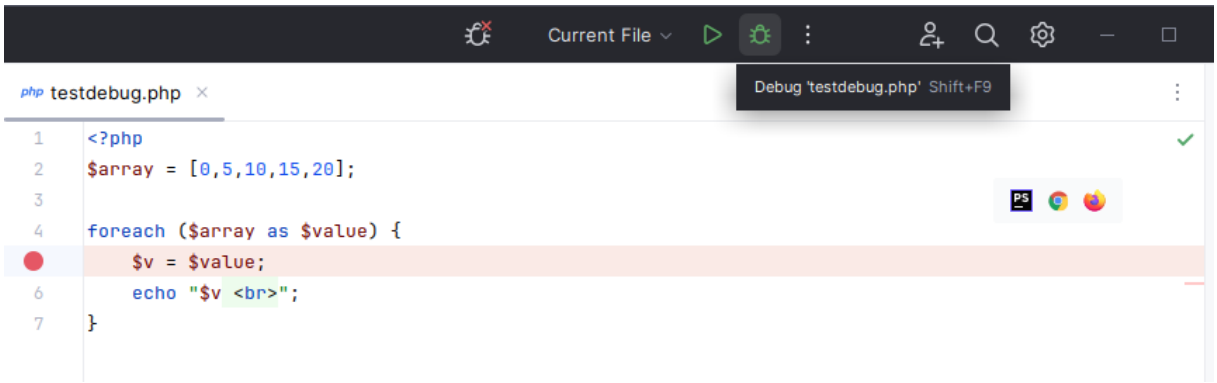


Figure 9: Debugging in PhpStorm

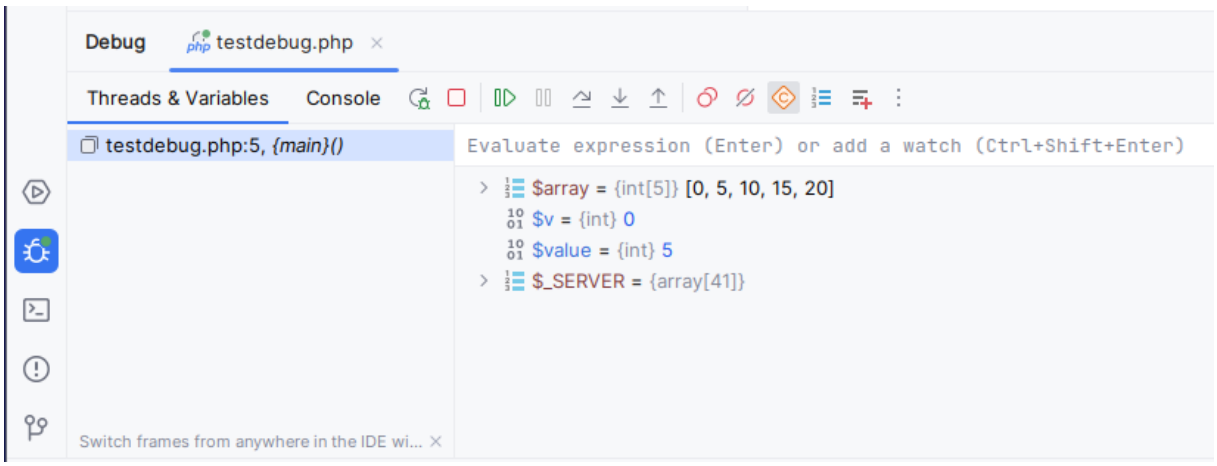


Figure 10: Debug console

> PhpStorm automates the process of getting Xdebug up and running. On your first attempt to run a debugging session without a debugger installed, the IDE will prompt you to download and install the relevant version of Xdebug.

If you want to debug interactively a web app, toggle the `Start Listening for PHP Debug Connections` button or choose the same option in the Run menu. You need to have the browser's add-on installed on your browser.

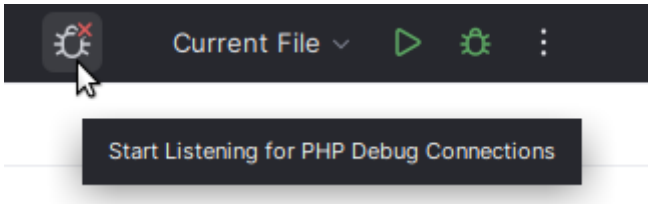


Figure 11: Start Listening for PHP Debug Connections

Then, open your app in the browser. It will appear the incoming connection from Xdebug dialog in PhpStorm. Then you can debug your app and see the result at the same time in the web browser.

5 Install the browser's add-on

We can install an add-on to our browser in order to enhance the debugging process. In Firefox or Google Chrome search for the Xdebug Helper add-on and install it:

To activate it, click on the bug in the address bar and change it to green:

Figure 12: Xdebug Helper for Firefox

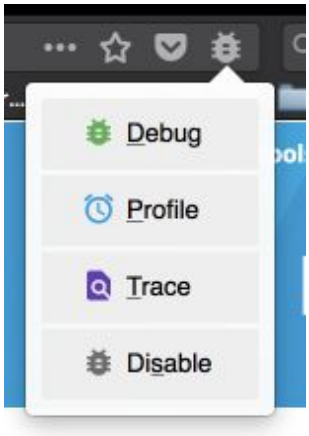


Figure 13: Xdebug Helper