Step Debugging Laravel With Xdebug

Overview

Xdebug (v3.2.1) is an extension for PHP comprising code analysis tools such as step debugging, profiling and test coverage analysis.

This page will only currently explore the installation and use of the step debugger, though it can be extended the other aforementioned use cases later on, if they are deemed to be important.

The installation process is not straight forward at all, and seemed to have confused PHP developers everywhere, which must be why (in addition to countless stackoverflow threads) this Youtube video exists, with Derick Rethans, the creator of Xdebug, doing a nice walkthrough. It can be watched for more details about the program, if interest arises.

If following a good documentation, this should not take you more than 5-10 minutes though.

Installation

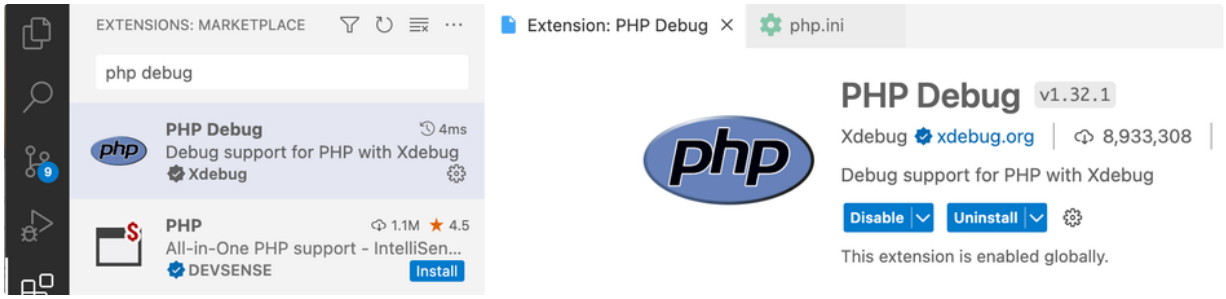
1. **Use the correct PHP version**. Each Xdebug installation is specific to only one PHP version on your local computer, so it is *important that you install it using the same the PHP version running the Paywho API system.* Therefore, run phpinfo() from public/index.php file to display the used PHP version (is there an easier to way to find out?). The Xdebug installation is probably made easier by "php -v" displaying the same version in the terminal. Additionally, Xdebug is installed with pecl, so it is also important to use the pecl distribution included with the very same PHP version as well.
    a. If this is ***not*** the case for you, it can be fixed this way
        i. Find the relevant PHP distribution. Not sure how to do this actually, but mine was at "/usr/local/Cellar/php@<version>/<version>/bin/" and Narin's at "/usr/local/homebrew/opt/php@<version>/bin/".
        ii. run "which php" and "which pecl". Hopefully the output is the same folder (e.g. "/usr/local/bin/")
        iii. Running "ls -la /usr/local/bin/" displays the symlinks of php and pecl. Reroute them:
            1. ln -s /usr/local/Cellar/php@<version>/<version>/bin/php /usr/local/bin/php
            2. ln -s /usr/local/Cellar/php@<version>/<version>/bin/pecl /usr/local/bin/pecl
            3. don't forget to double check the PHP version again ("php -v")
2. **Run "pecl install xdebug"**. Then "php -v" to see if xdebug is included in the output: "*with Xdebug v3.2.1, Copyright (c) 2002-2023, by Derick Rethans)*"

```
PHP 8.1.18 (cli) (built: Apr 12 2023 11:56:55) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.1.18, Copyright (c) Zend Technologies
    with Xdebug v3.2.1, Copyright (c) 2002–2023, by Derick Rethans
    with Zend OPcache v8.1.18, Copyright (c), by Zend Technologies
```

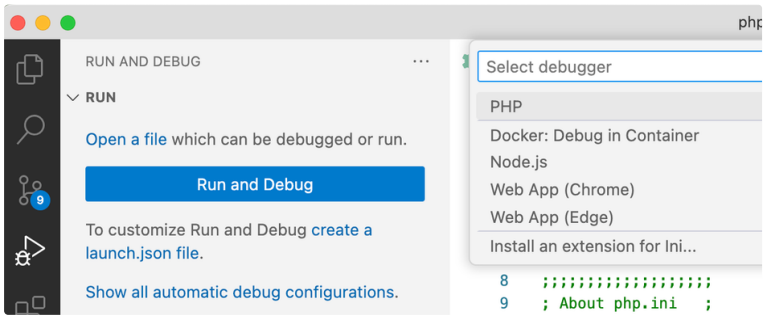
3. **Edit your php.ini file** (find it with "php --ini", output: Loaded Configuration File: "/usr/local/etc/php/8.1/php.ini")
    a. At the top, zend_extension="xdebug.so" should already have been added automatically
    b. Directly underneath, add following 3 settings
        i. xdebug.mode = debug
        ii. xdebug.start_with_request = yes
        iii. xdebug.client_port = 9003
        iv. (Here is a list of all settings, if you are interested).
        v. A restart of our Apache server ("brew services restart httpd") might be necessary to reload any changes to the PHP configuration file (*is there a less nuclear option?*)

```
php.ini
usr > local > etc > php > 8.1 > php.ini
1   zend_extension="xdebug.so"
2   xdebug.mode = debug
3   xdebug.start_with_request = yes
4   xdebug.client_port = 9003
5
6   [PHP]
7
8   ...................
```

4. **Install the Xdebug VS Code extension**, “PHP Debug”.



5. **Last step: Create a launch.json file**. In the “Run and Debug” pane of VS Code, click the link “create a launch.json file“, then choose “PHP“.

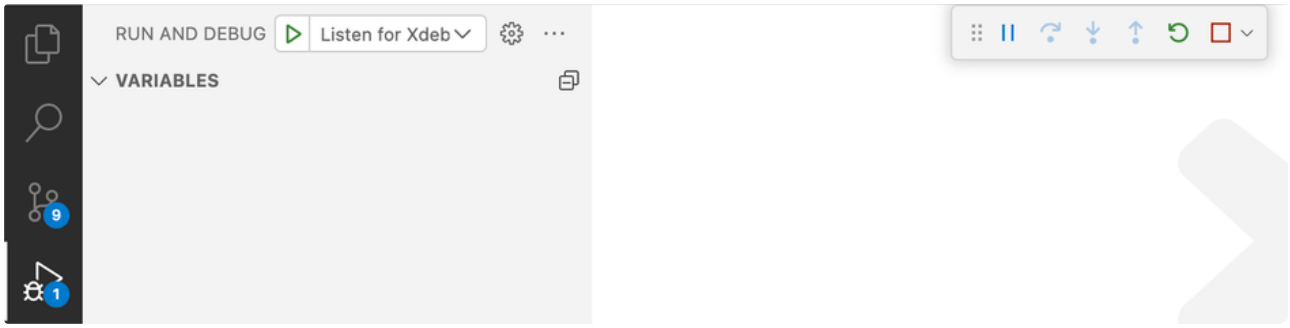


The launch file should simply be automatically created like in the screenshot below (Note that the port number (9003) matches the client port property in the php.ini file, which is important).

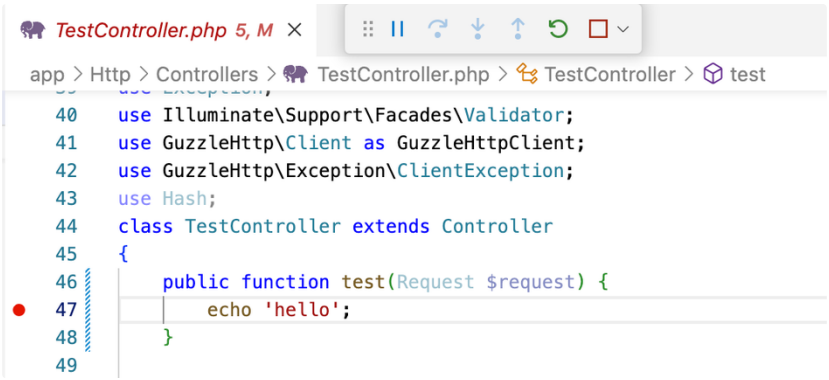
```
launch.json
php.ini
.vscode > launch.json > Launch Targets > {} Launch Built-in w
1   {
2       // Use IntelliSense to learn about possible
3       // Hover to view descriptions of existing at
4       // For more information, visit: https://go.m
5       "version": "0.2.0",
6       "configurations": [
7
8           {
9               "name": "Listen for Xdebug",
10              "type": "php",
11              "request": "launch",
12              "port": 9003
13          },
```

Debugging

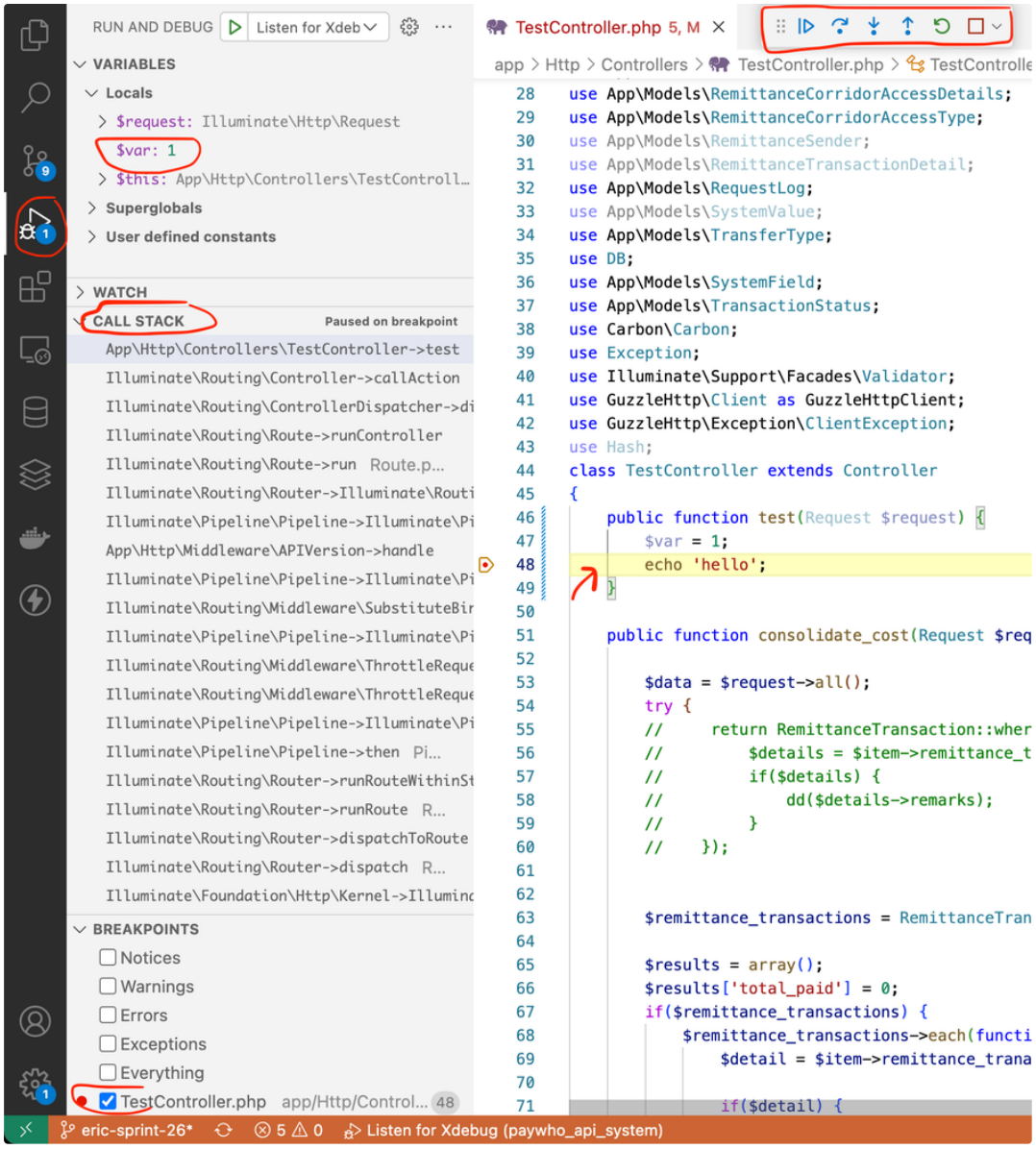
1. **Click “Play”** in the “Run and Debug“ pane of VS Code, and note the step debugger control window appearing to the right .



2. Set a debug point somewhere.



3. Using the browser or Postman or whichever tool you use, call an API. VS Code should automatically pop into focus and the code should light up at the debug point if the running thread is hitting it. Use the step debugger controls to your heart's content. Notice that you can see the value of any variable in scope, all your debug points and the call stack, in which you can click through the stack and follow the thread.


RUN AND DEBUG
Listen for Xdeb
VARIABLES
Locals
$request: Illuminate\Http\Request
$var: 1
$this: App\Http\Controllers\TestControll…
Superglobals
User defined constants
WATCH
CALL STACK
Paused on breakpoint
App\Http\Controllers\TestController->test
Illuminate\Routing\Controller->callAction
Illuminate\Routing\ControllerDispatcher->di
Illuminate\Routing\Route->runController
Illuminate\Routing\Route->run Route.p…
Illuminate\Routing\Router->Illuminate\Routi
Illuminate\Pipeline\Pipeline->Illuminate\Pi
App\Http\Middleware\APIVersion->handle
Illuminate\Pipeline\Pipeline->Illuminate\Pi
Illuminate\Routing\Middleware\SubstituteBir
Illuminate\Pipeline\Pipeline->Illuminate\Pi
Illuminate\Routing\Middleware\ThrottleReque
Illuminate\Routing\Middleware\ThrottleReque
Illuminate\Pipeline\Pipeline->Illuminate\Pi
Illuminate\Pipeline\Pipeline->then Pi…
Illuminate\Routing\Router->runRouteWithinSt
Illuminate\Routing\Router->runRoute R…
Illuminate\Routing\Router->dispatchToRoute
Illuminate\Routing\Router->dispatch R…
Illuminate\Foundation\Http\Kernel->Illumina
BREAKPOINTS
Notices
Warnings
Errors
Exceptions
Everything
TestController.php app/Http/Control… 48
TestController.php 5, M
app > Http > Controllers > TestController.php > TestControlle
28 use App\Models\RemittanceCorridorAccessDetails;
29 use App\Models\RemittanceCorridorAccessType;
30 use App\Models\RemittanceSender;
31 use App\Models\RemittanceTransactionDetail;
32 use App\Models\RequestLog;
33 use App\Models\SystemValue;
34 use App\Models\TransferType;
35 use DB;
36 use App\Models\SystemField;
37 use App\Models\TransactionStatus;
38 use Carbon\Carbon;
39 use Exception;
40 use Illuminate\Support\Facades\Validator;
41 use GuzzleHttp\Client as GuzzleHttpClient;
42 use GuzzleHttp\Exception\ClientException;
43 use Hash;
44 class TestController extends Controller
45 {
46 public function test(Request $request) {
47 $var = 1;
48 echo 'hello';
49 }
50
51 public function consolidate_cost(Request $req
52
53 $data = $request->all();
54 try {
55 // return RemittanceTransaction::wher
56 // $details = $item->remittance_t
57 // if($details) {
58 // dd($details->remarks);
59 // }
60 // });
61
62
63 $remittance_transactions = RemittanceTran
64
65 $results = array();
66 $results['total_paid'] = 0;
67 if($remittance_transactions) {
68 $remittance_transactions->each(functi
69 $detail = $item->remittance_trana
70
71 if($detail) {
eric-sprint-26*
5 0
Listen for Xdebug (paywho_api_system)