Troubleshooting PhpStorm debugging

When using Xdebug or Zend Debugger with PhpStorm, there are cases where a bit of configuration is missed or some setting prevents the debugger from working. In this tutorial, we'll go over some common issues and see how we can troubleshoot them.

The individual debugging tutorials may also contain useful troubleshooting tips for specific scenarios, such as Multi-user debugging in PhpStorm with Xdebug and DBGp proxy and Remote debugging in PhpStorm via SSH tunnel.

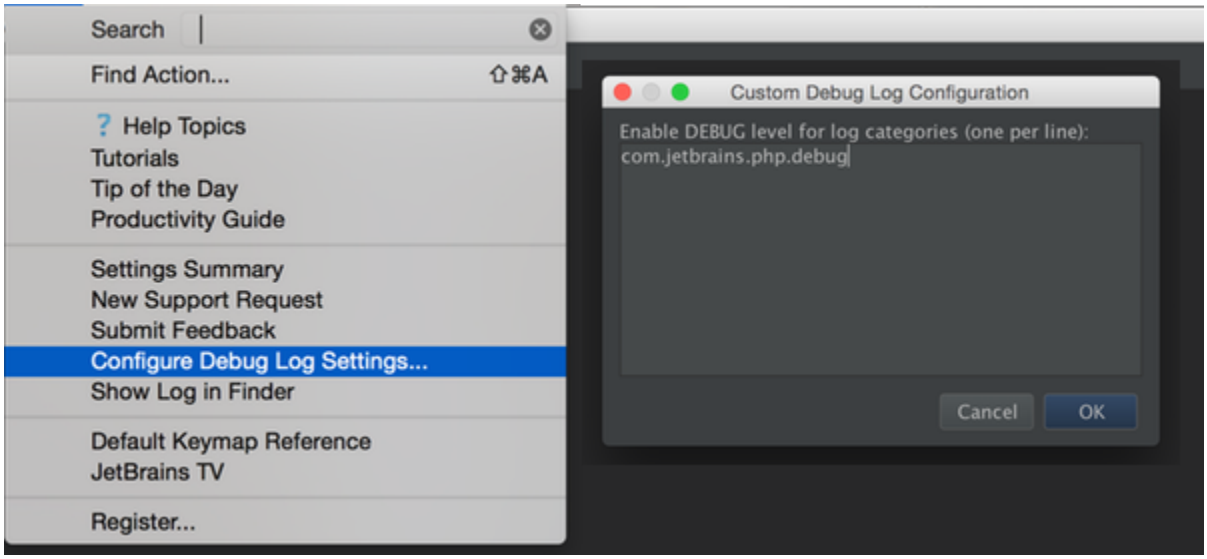
How to collect logs and provide them to our support engineers?

If the answer to a debugging issue is not on this page, it is possible to contact our support engineers. They might request you to capture logs of the operations at hand. If that is the case, please follow instructions to collect logs.

Show instructions here...

Collecting logs with PhpStorm 7.1.3 or above:

1. Open Help | Configure Debug Log Settings
2. Add the following line: `com.jetbrains.php.debug`



3. Click OK

4. Reproduce the problem.
5. Log files are to be found at:

   - from PhpStorm (since 2.1 EAP):
     - Windows\Linux: Help | Show Log in Explorer
     - Mac OS X: Help | Show Log in Finder
   - manually (~ stands for user home directory):
     - Windows and Linux: `~\.WebIde70\system\log`
     - Mac OS X: `~/Library/Logs/WebIde70`

   The latest log file is named `idea.log`; older files names end with a number, i.e. `idea.log.1`, `idea.log.2` etc. In most cases we only need the latest one.

6. Attach the log file(s) to the issue or forum thread.

Collecting logs with PhpStorm 7.0 or below:

1. Close PhpStorm application.
2. Open file `<PhpStorm installation>/bin/log.xml` with the text editor
3. Scroll to the bottom and add the following lines before the `<root>` tag:

```
<category name="#com.jetbrains.php.debug">
     <priority value="DEBUG"/>
 </category>
```

4. Start PhpStorm and reproduce the problem.

5. Log files are to be found at:

   - from PhpStorm (since 2.1 EAP):
     - Windows\Linux: Help | Show Log in Explorer
     - Mac OS X: Help | Show Log in Finder
   - manually (~ stands for user home directory):
     - Windows and Linux: `~\.WebIde70\system\log`
     - Mac OS X: `~/Library/Logs/WebIde70`

   The latest log file is named `idea.log`; older files names end with a number, i.e. `idea.log.1`, `idea.log.2` etc. In most cases we only need the latest one.

6. Attach the log file(s) to the issue or forum thread.

Collecting log files from Xdebug

When using Xdebug, it is possible to make it log its actions. Edit `php.ini` and enable `xdebug.remote_log` (see http://www.xdebug.org/docs/all_settings), for example by adding:

```
xdebug.remote_log=path_to_log/xdebug.log
```

The log file contains the raw communication between PhpStorm and Xdebug as well as any warnings or errors:

```
Log opened at 2015-01-08 08:14:28
I: Connecting to configured address/port: 127.0.0.1:9000.
I: Connected to client. :-)
-> <init xmlns="urn:debugger_protocol_v1" xmlns:xdebug="http://xdebug.org/dbgp/xdebug"
fileuri="file:///C:/Projects/Samples/DebuggingTutorial/debugging.php" language="PHP"
protocol_version="1.0" appid="3040" idekey="11774"><engine
version="2.2.5"><![CDATA[Xdebug]]></engine><author><![CDATA[Derick
Rethans]]></author><url><![CDATA[http://xdebug.org]]></url><copyright><![CDATA[Copyright
(c) 2002-2014 by Derick Rethans]]></copyright></init>

<- step_into -i 5
-> <response xmlns="urn:debugger_protocol_v1" xmlns:xdebug="http://xdebug.org/dbgp/xdebug"
command="step_into" transaction_id="5" status="break" reason="ok"><xdebug:message
filename="file:///C:/Projects/Samples/DebuggingTutorial/debugging.php"
lineno="2"></xdebug:message></response>

...
```

Disable this log when its no longer needed. It will not automatically roll-over or be truncated and may grow to a vast file size.

Make sure Xdebug or Zend Debugger are installed and configured

To debug PHP code with PhpStorm, we will need Xdebug or Zend Debugger installed into our PHP runtime. Make sure either Xdebug or Zend Debugger are installed and configured with PhpStorm.

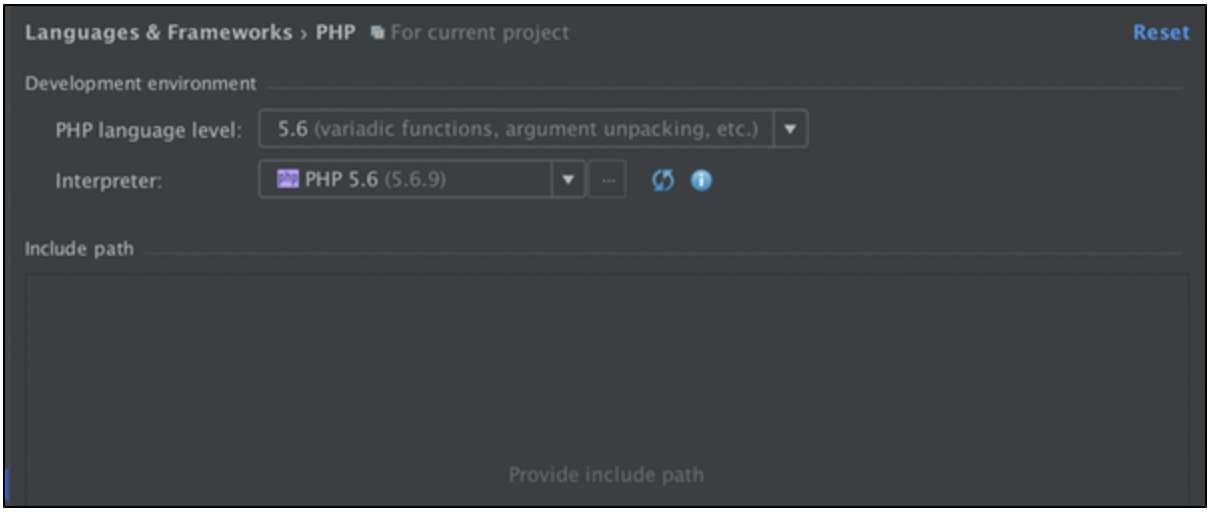
Make sure only one debugger is installed. Xdebug and Zend Debugger can not be configured for the same PHP interpreter without conflicts.

Verifying a debugger is installed using PhpStorm

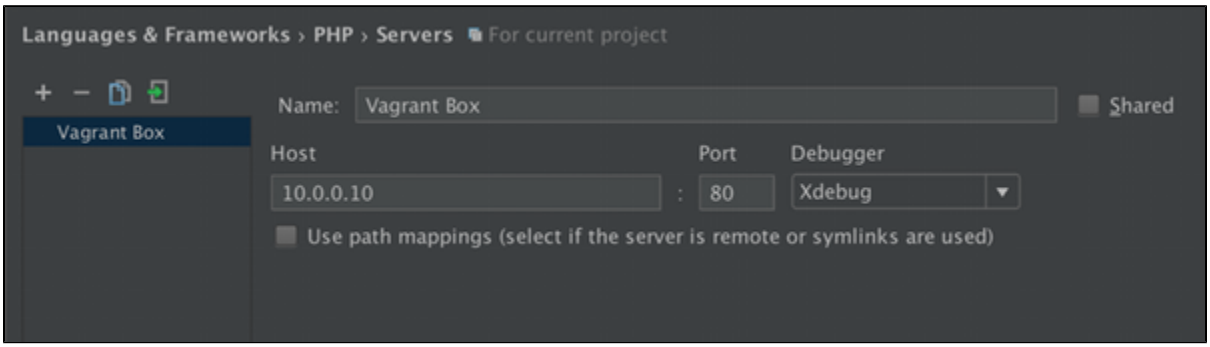
Please note that debugger should be configured for the proper PHP interpreter (= so the proper php.ini file should be edited).

There is a difference between PHP for web applications and PHP for CLI scripts, make sure that you've configured (and checked configuration for) the one you need.

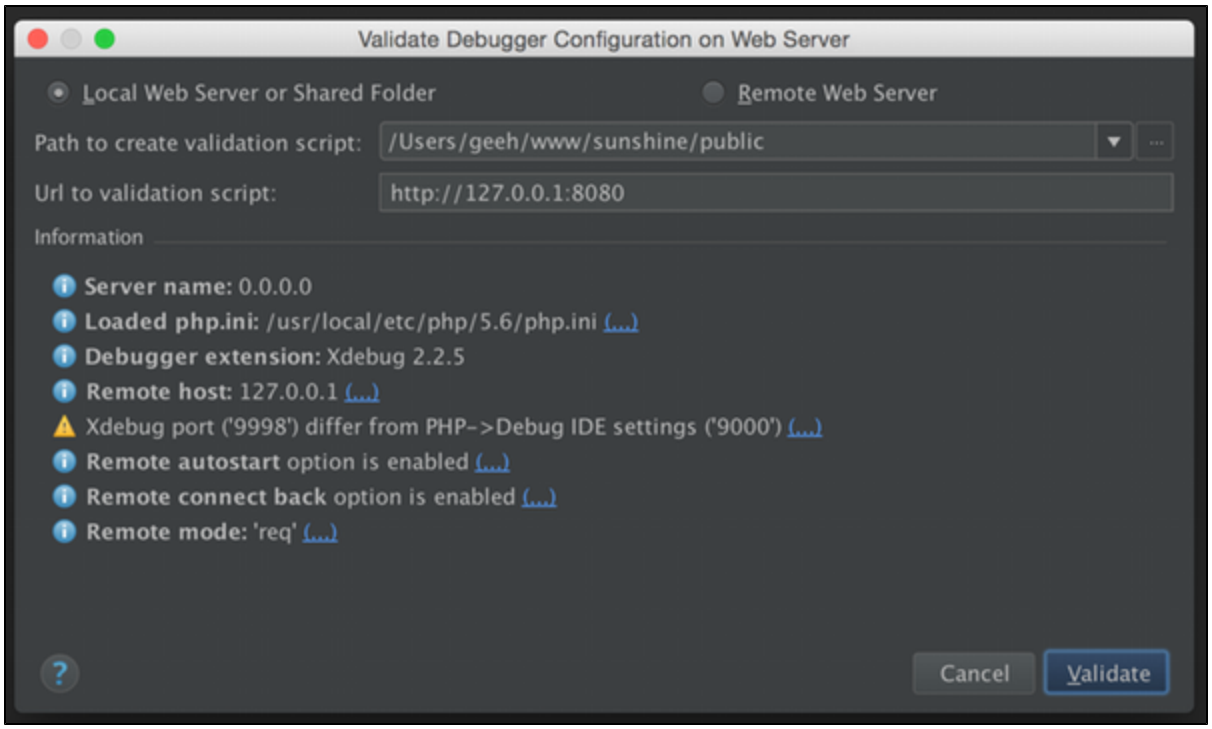
For PhpStorm versions 7 and up, we can make use of the built-in Debugger Configuration Validation. When configuring the PHP interpreter for our project (under settings, Languages & Frameworks | PHP) , PhpStorm informs us if a debugger is installed and tells us whether Xdebug or Zend Debugger are used.



Firstly, we need to check that our PHP server is configured correctly using Languages & Frameworks | PHP | Servers . Note that deployment server and path mapping must be created beforehand to make this work.



Now, we can use the Run | Web Server Debug Validation dialog to verify if the web server's debugger can communicate with PhpStorm. It's important that the Path to create validation script and Url to validation script are correct in order that the tool connects to your web server correctly.



Note that you can use the (...) after each check to take you directly to the configuration section that will solve that particular

problem.

Verifying Xdebug is installed

To verify a debugger is installed for CLI debugging, run the following command from the terminal:

```
php --version
```

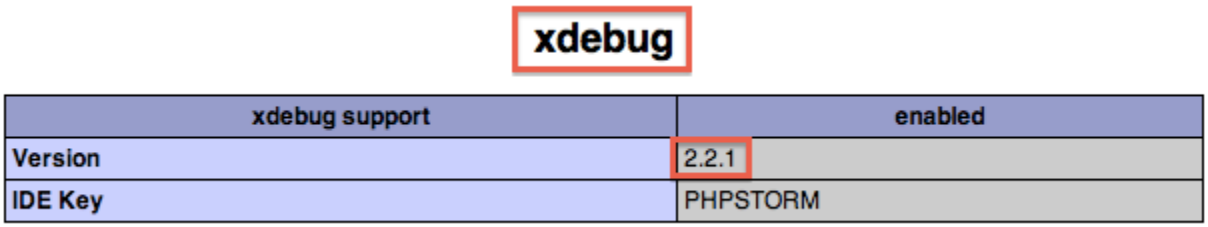
The output should reveal Xdebug is installed:

```
PHP 5.5.3 (cli) (built: Aug 20 2013 16:11:53)
Copyright (c) 1997-2013 The PHP Group
Zend Engine v2.5.0, Copyright (c) 1998-2013 Zend Technologies
    with Xdebug v2.2.3, Copyright (c) 2002-2013, by Derick Rethans
```

To verify a debugger is installed for web debugging, create a file containing the following code and run it in the browser:

```
<?php
phpinfo();
```

The output of this page should contain an Xdebug section:



xdebug

xdebug support	enabled
Version	2.2.1
IDE Key	PHPSTORM

Check the Xdebug Installation Guide for additional information.

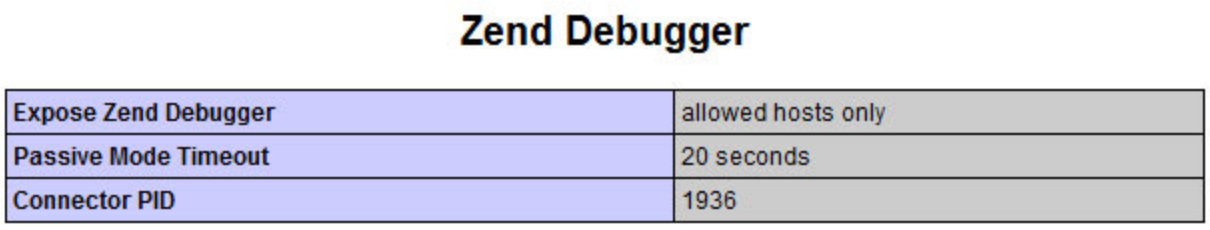
Verifying Zend Debugger is installed

Verification is done in a similar way as for Xdebug.

Create a file containing the following code and run it in the browser:

```
<?php
phpinfo();
```

A Zend Debugger section should be present:



Zend Debugger

Expose Zend Debugger	allowed hosts only
Passive Mode Timeout	20 seconds
Connector PID	1936

Check the Zend Debugger Installation Guide for additional information.

Startup warnings and errors preventing the debugger from working

When running PHP, it can happen that a startup warning or error is displayed. When this is the case, the debugger may fail to work. PhpStorm will also not be able to recognize the debugger being used.

Verify no startup warnings or errors are shown by running:

```
php --version
```

This will return a message similar to the following:

```
<some error/warning>
PHP 5.5.3 (cli) (built: Aug 20 2013 16:11:53)
Copyright (c) 1997-2013 The PHP Group
Zend Engine v2.5.0, Copyright (c) 1998-2013 Zend Technologies
    with Xdebug v2.2.3, Copyright (c) 2002-2013, by Derick Rethans
```

If the first line(s) contain errors and warnings, it's recommended to fix these before continuing.

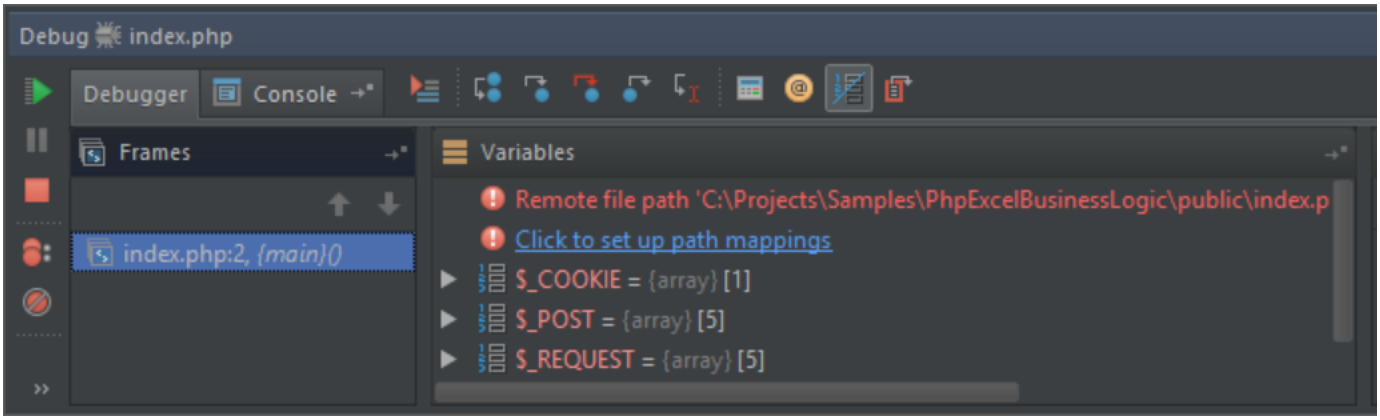
The debugger can not connect

When the debugger can not connect or refuses the connection, check the following:

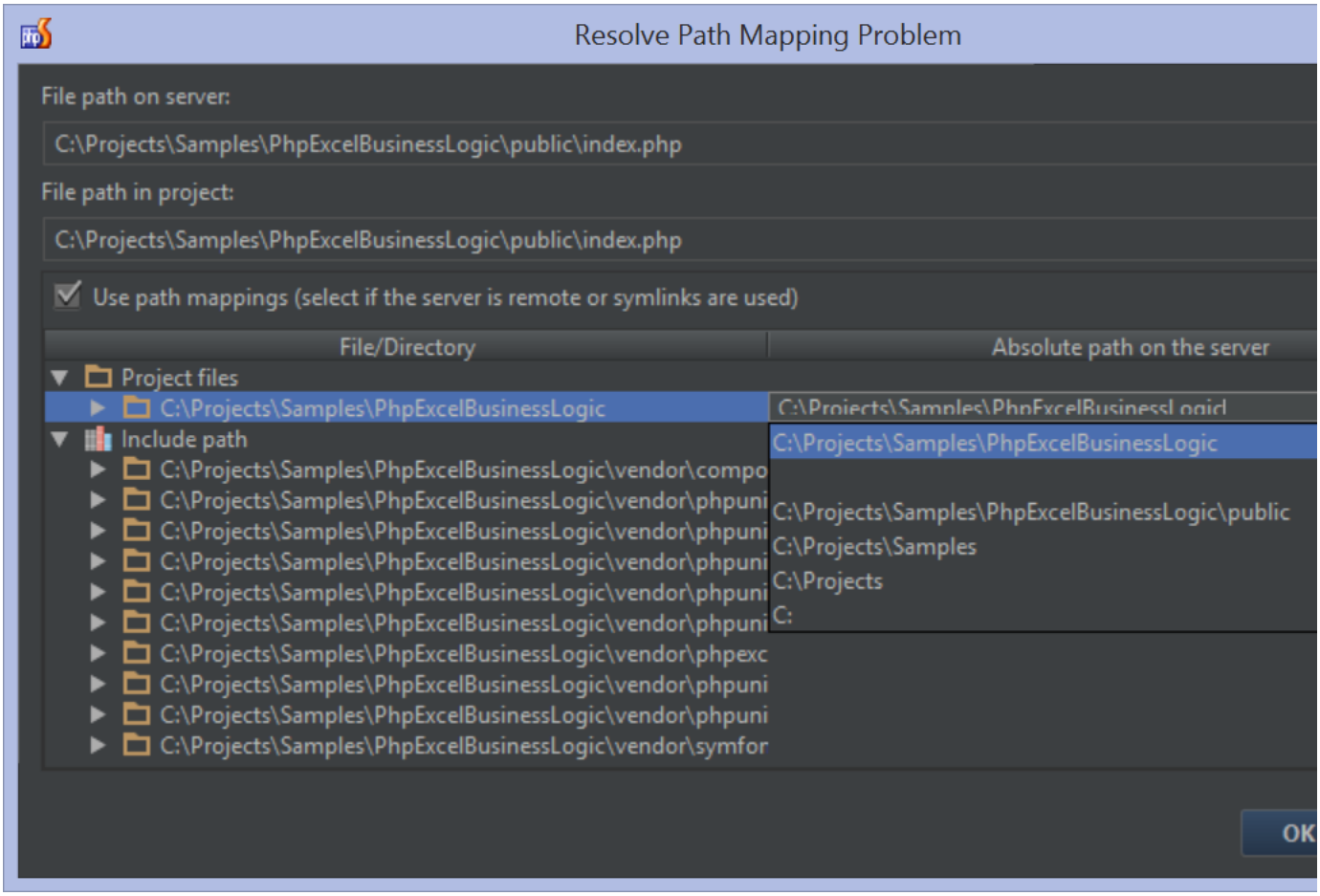
- Make sure Xdebug or Zend Debugger are configured to connect to the host and port PhpStorm is running on.
  - In the Xdebug configuration make sure `xdebug.remote_host` and `xdebug.remote_port` are correct. See Xdebug documentation for more info.
  - Using Zend Debugger, make sure the generated PhpStorm bookmarklets or Browser Debugging Extension is configured to use the correct IP address and port.
  - When using a remote PHP interpreter, verify the steps outlined in Remote debugging in PhpStorm via SSH tunnel
- Make sure PhpStorm and Xdebug / Zend Debugger are configured with the same port numbers. Configuring port numbers in PhpStorm can be done under settings / preferences, expanding Languages & Tools | PHP | Debug .
- Make sure PhpStorm is listening for incoming debugger connections.
- Verify no firewall, router or ISP is blocking the connection. This can be verified running `telnet host 9000` (Xdebug) or `telnet host 10137` (Zend Debugger) from remote server (where host is an IP address of your local machine running PhpStorm) and checking a connection is established. You can use canyouseeme.org or similar service to check for opened inbound ports.
- Make sure that xdebug.remote_host (for Xdebug) or zend_debugger.allow_hosts (for Zend Debugger) are properly configured. The value can be a host name (e.g. localhost) or an IP address (those are host name or IP address of the machine where PhpStorm is running), and it must be pingable from the server and should lead to the host where PhpStorm is running. When using Xdebug, xdebug.remote_connect_back can be used for troubleshooting.

Remote file path is not mapped to any file path in project

In some cases, the debugger can connect but we get the error message "Remote file path 'path/to/script/on/the/server.php' is not mapped to any file path in project" or "The script 'path/to/script/on/the/server.php' is outside the project." This means that PhpStorm is not sure which local file corresponds to the file being debugged.



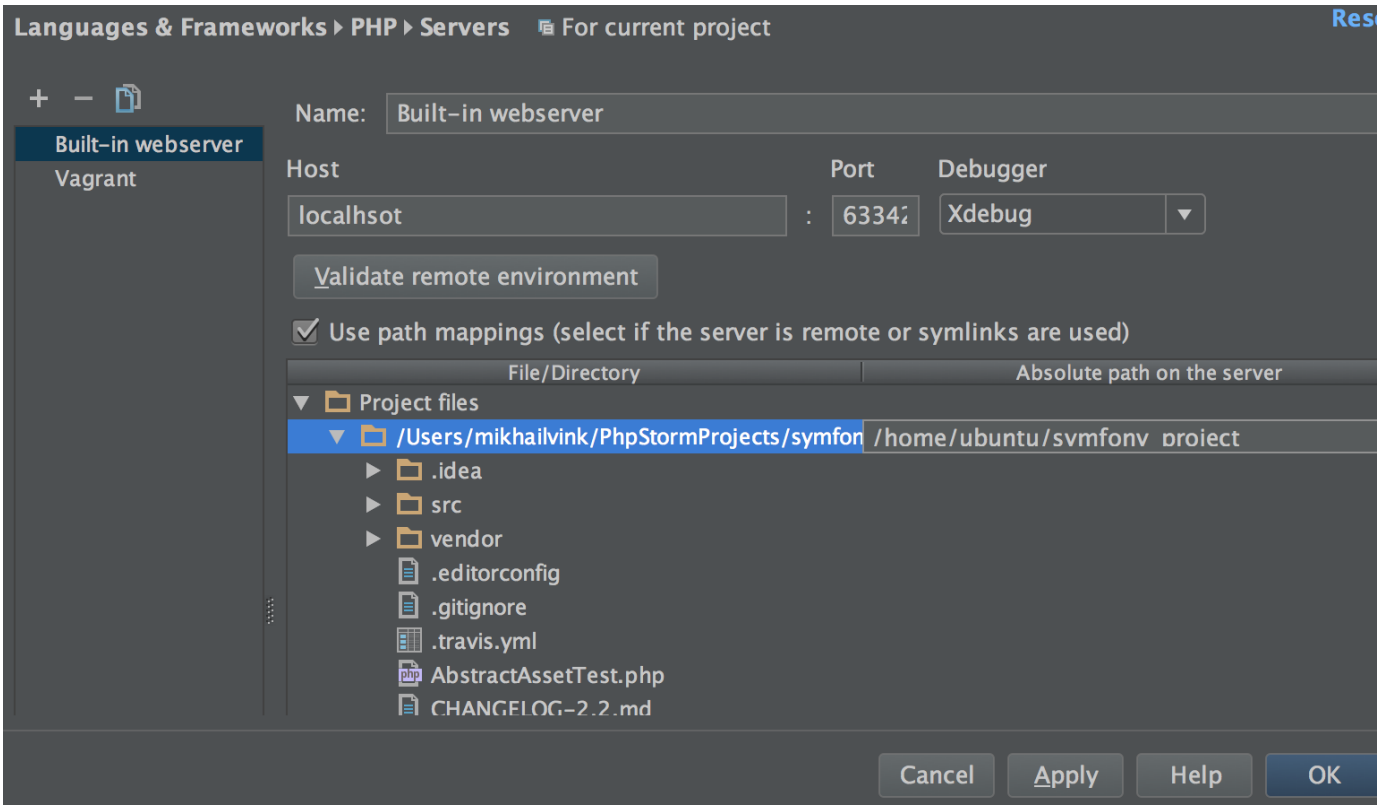
We can solve this problem quickly by clicking Click to set up path mappings and setting up the necessary path mappings.



Path mappings are used when the paths to the files processed by the server differ from the paths to the files in the PhpStorm project. This happens in the following cases:

- The server is remote and the project files are local copies of the originals.
- The files processed by the server and the files opened in the IDE are the same, but symlinks are used. Because the debugger resolves symlinks during a debugging session, you need to tell the IDE actual absolute physical paths to the files on the server.

Path mappings can be always configured in Settings/Preferences | Languages & Frameworks | PHP | Servers | ... (and, as described below, following the set up path mappings link when requested by PhpStorm during the debugging session.



If the files that the server processes are in the project and you are not using symlinks, clear the Use path mappings check box. In this case, the IDE will open files according to the paths received from the debugger.

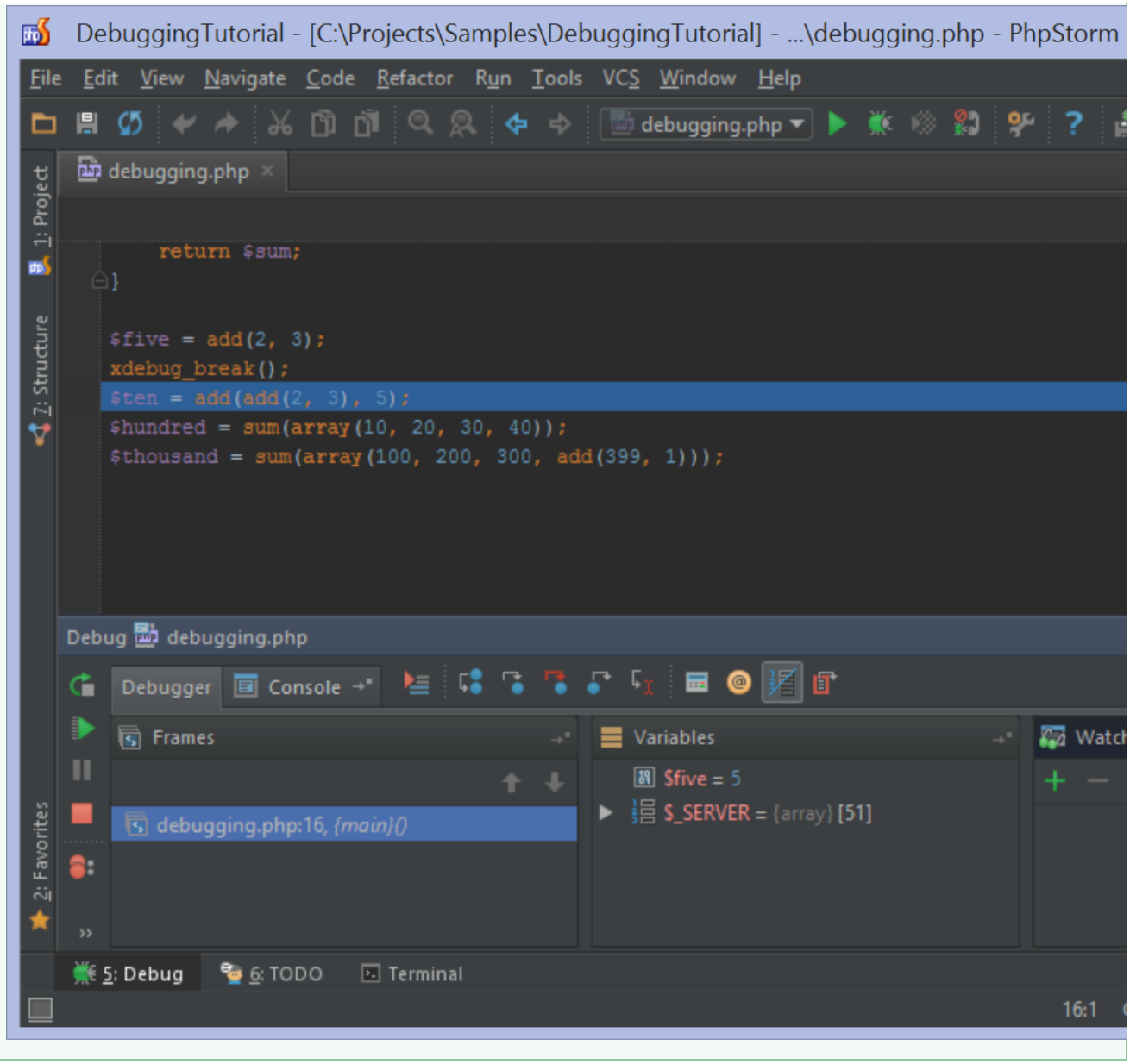
A path mapping specified for a parent directory is automatically applied to all its subdirectories. If necessary, path mappings for any subdirectory or file can be specified.

Breakpoints are not being hit

When breakpoints are not being hit but the debugger seems to connect, verify the following:

- Verify the breakpoint is not disabled. Using the Run | View Breakpoints... menu (Ctrl+Shift+F8 / CMD+Shift+F8), check that breakpoints are enabled. Tick the checkbox in front of a breakpoint to enable it.
- Make sure path mappings are correct.
- On Mac OS X when renaming a file or folder using Finder and changing the letter case, PhpStorm may not be able to find the files. If a rename is required, make sure to do it in the IDE or using a terminal and the `mv` command. Renaming project files or folders using Finder may result in strange behaviour. See http://devnet.jetbrains.com/message/5488439#5488439 for a full discussion about this.
- There are some rare cases when breakpoints can't be hit due to technical limitations, e.g. caused by the way how PHP generates byte-code (see this comment for more information).

When using Xdebug, we can use the `xdebug_break()` function to force a breakpoint from within PHP. When Xdebug encounters this function during execution, it will pause at the next statement in the IDE even if no breakpoint was defined there originally.



The debugger does not work when using nginx

When using nginx webserver, debugging may fail if the `$_SERVER["SERVER_NAME"]` is not provided by PHP. To solve this, add a fastcgi parameter to the nginx configuration. An example is available on http://wiki.nginx.org/PHPFcgiExample and could look like the following:

```
fastcgi_param SERVER_NAME $server_name;
```

or

```
fastcgi_param SERVER_NAME $host;
```

See https://devnet.jetbrains.com/message/5494835#5494835 for a full discussion about this.

XDebug can't connect to PhpStorm

If you see the following messages in XDebug log:

```
Log opened at 2017-02-21 17:52:27
I: Connecting to configured address/port: 172.19.0.1:9000.
W: Creating socket for '172.19.0.1:9000', poll success, but error: Operation now in progress (29).
E: Could not connect to client. :disappointed:
Log closed at 2017-02-21 17:52:27
```

or

```
Log opened at 2017-02-22 13:17:13
I: Connecting to configured address/port: 10.10.10.10:9000.
E: Time-out connecting to client. :-(
Log closed at 2017-02-22 13:17:14
```

This means that XDebug tries to connect to the host and can't make the connection. To fix the issue set `xdebug.remote_connect_back=0` and/or make sure that `xdebug.remote_host` is set correctly.

Zend Debugger will not perform dropping of privileges

When using Zend Debugger on Linux, we may see the following error in some cases:

```
Zend Debugger: Cannot read a valid value of zend_debugger.httpd_uid or zend.httpd_uid, will not perform dropping of privileges
```

This error occurs when the debugger can not determine the uid of the user running the web server process. Find the uid of that user and add it to `php.ini`:

```
zend_debugger.httpd_uid=1234
```

Do not forget to restart the web server after updating this setting.

Debugging with ionCube installed

In some PHP environments, ionCube is used to load and run encoded PHP code. The ionCube loader extension does this by decoding the PHP code that is to be run and then making sure the PHP interpreter can execute it.

Officially, Xdebug nor Zend Debugger support running with ionCube enabled, however there are some workarounds we can try. Note that these are not supported by JetBrains, ionCube, Xdebug or Zend Debugger.

Workaround 1: Make sure the ionCube loader module is loaded first

In `php.ini`, verify the ionCube loader is loaded before any debugger extension is loaded. For example:

```
zend_extension=/usr/local/lib/php/extensions/no-debug-non-zts-20100525/ioncube_loader_lin_5.4.so
zend_extension=/usr/local/lib/php/extensions/no-debug-non-zts-20100525/xdebug.so
```

Workaround 2: Use ionCube on-demand loading

We can disable the ionCube extension in `php.ini` and make use of ionCube's on-demand loading feature. When the first

encoded PHP script is loaded, the PHP interpreter will check the configured `extenson_dir` to find the ionCube extension and run the encoded PHP script.

Note that this method is not officially supported. It should only be used for debugging purposes and never on a production server.