Debug PHP with Xdebug

with examples for Vagrant with Ubuntu

Christian Pojoni

www.secret-source.eu

Mar 2019

Contents

Why use Xdebug

Common Approach

```
var_dump($thing);
// or
print_r($data);
// or
error_log(print_r($someArray, true));
// or
echo __METHOD__ . '' . __LINE__;
```

Why use Xdebug

Common Approach

```php
var_dump($thing);
// or
print_r($data);
// or
error_log(print_r($someArray, true));
// or
echo __METHOD__ . '' . __LINE__;
```

Remove that again

don't commit these lines into VCS

Traditional Way

1. insert outputs
2. reload page
3. look for output
4. change some code

Traditional Way

1. insert outputs
2. reload page
3. look for output
4. change some code
5. repeat 1. until error is fixed

Traditional Way

1. insert outputs
2. reload page
3. look for output
4. change some code
5. repeat 1. until error is fixed
6. clean up outputs
7. load final version of the page

Xdebug Way

1. add breakpoints

Xdebug Way

1. add breakpoints
2. reload page

Xdebug Way

1. add breakpoints
2. reload page
3. step into the code

Advantages

- Faster development time

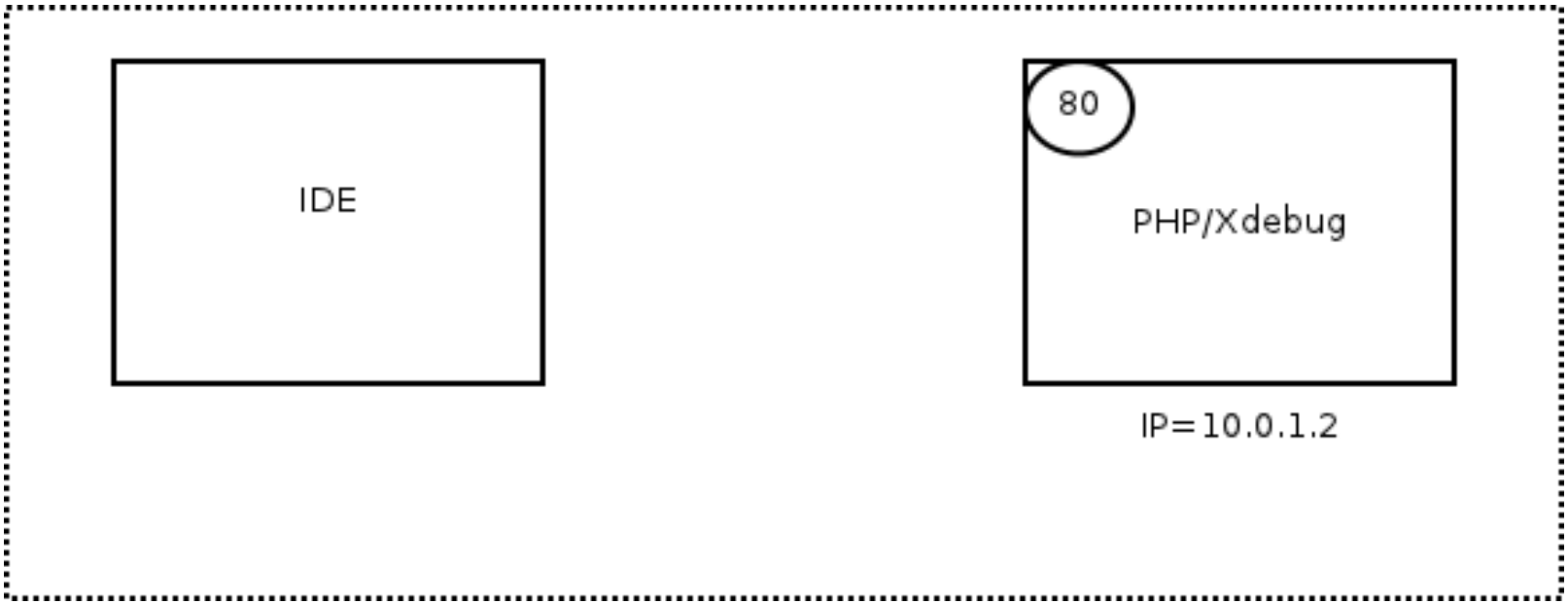
Advantages

- Faster development time
- Better understanding of what the code is doing

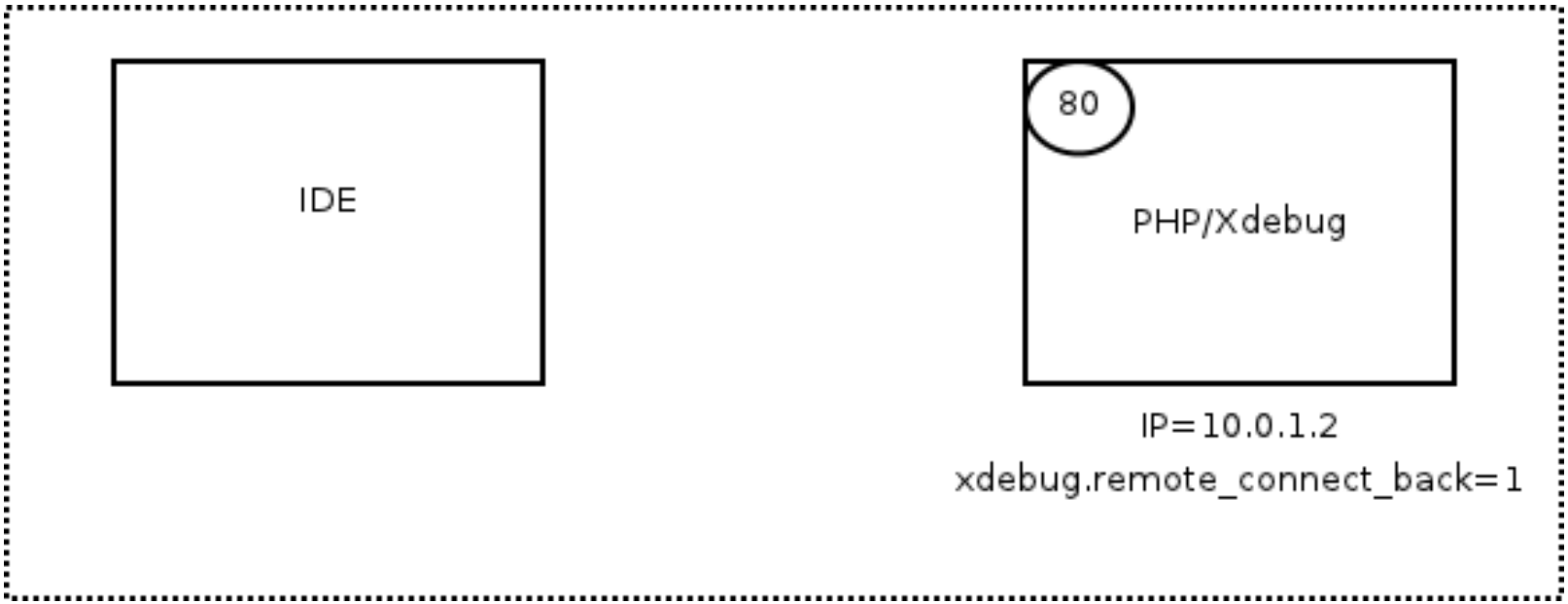
Advantages

- Faster development time
- Better understanding of what the code is doing
- Does not require code changes

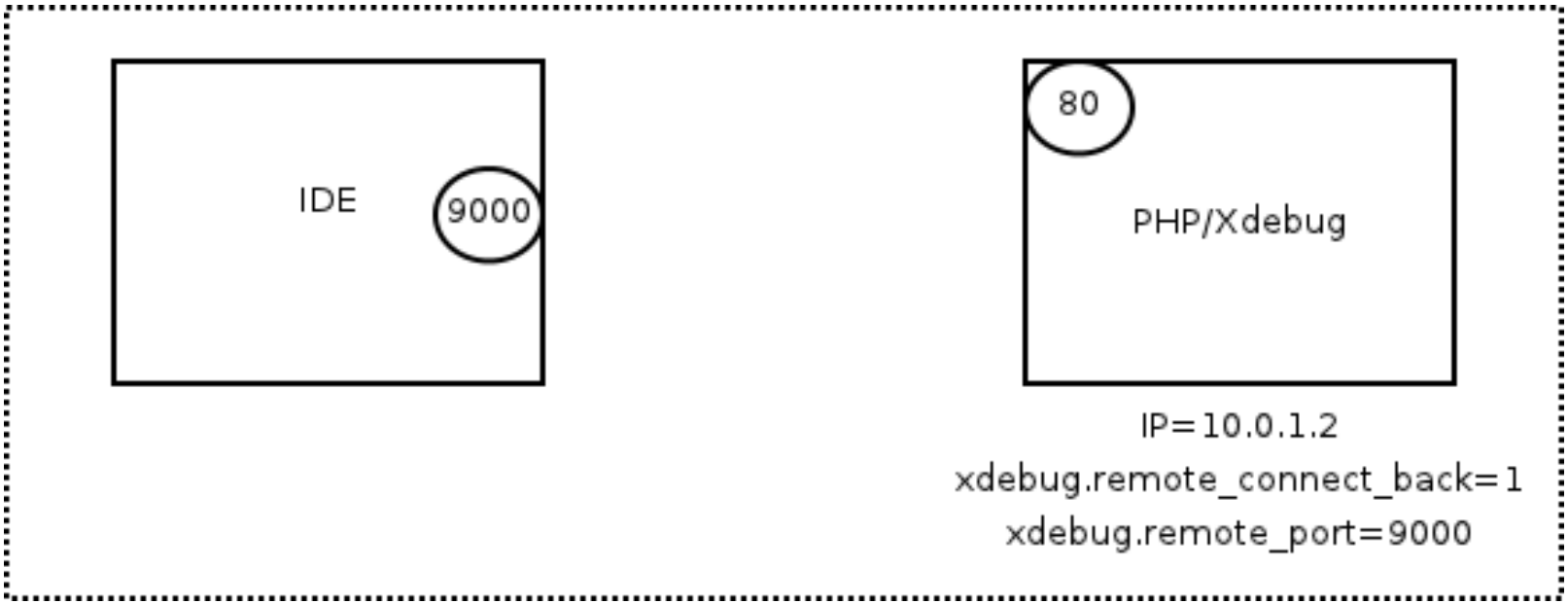
How it works



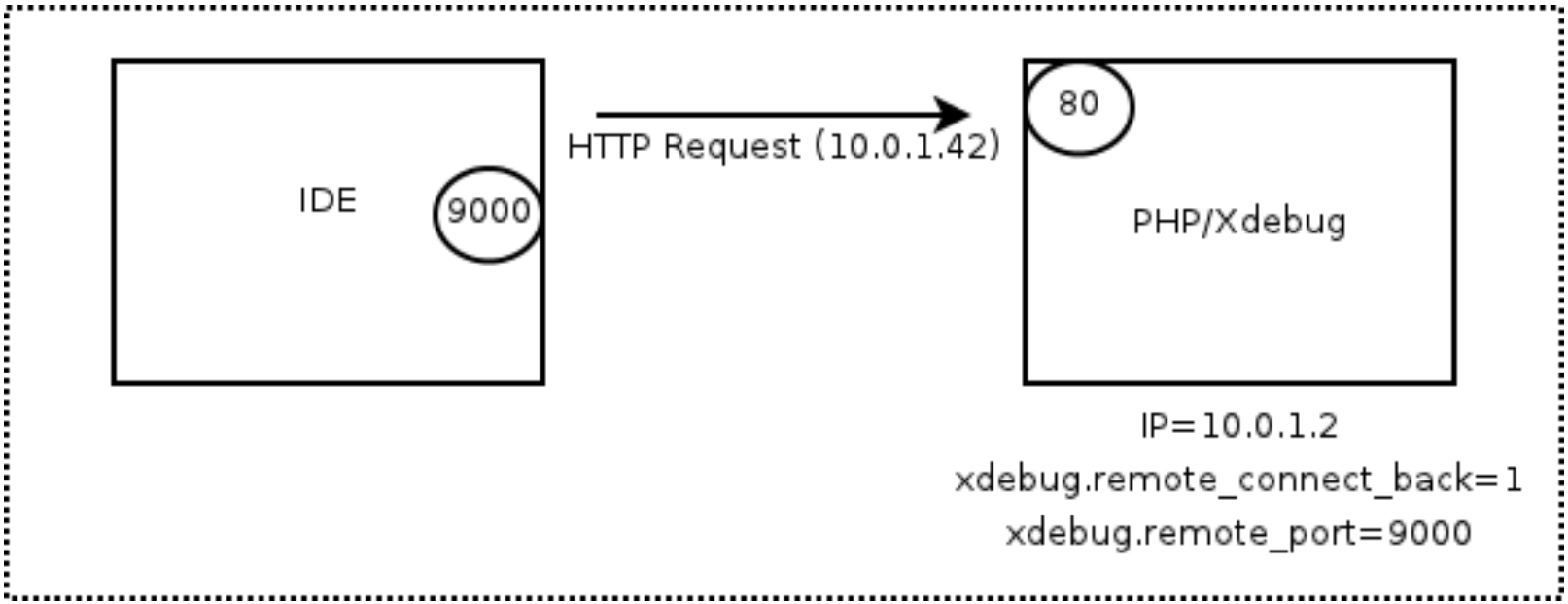
How it works



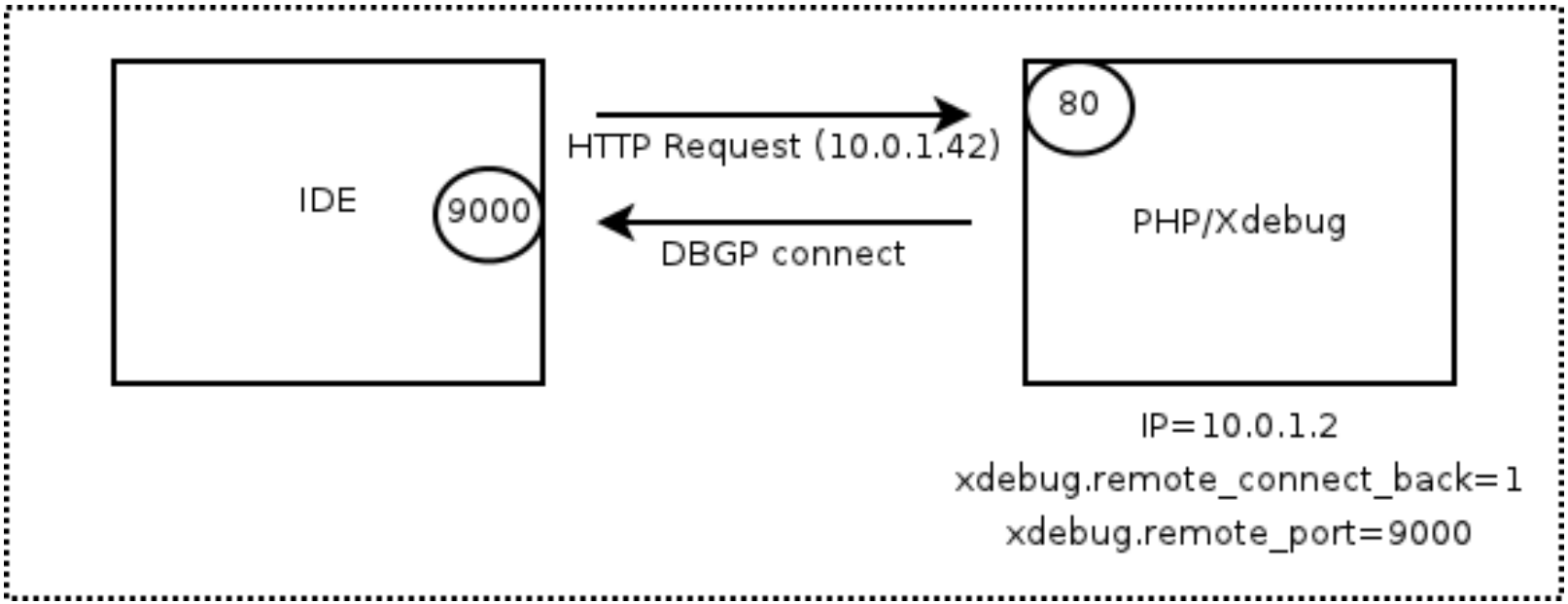
How it works


IDE
9000
80
PHP/Xdebug
IP=10.0.1.2
xdebug.remote_connect_back=1
xdebug.remote_port=9000

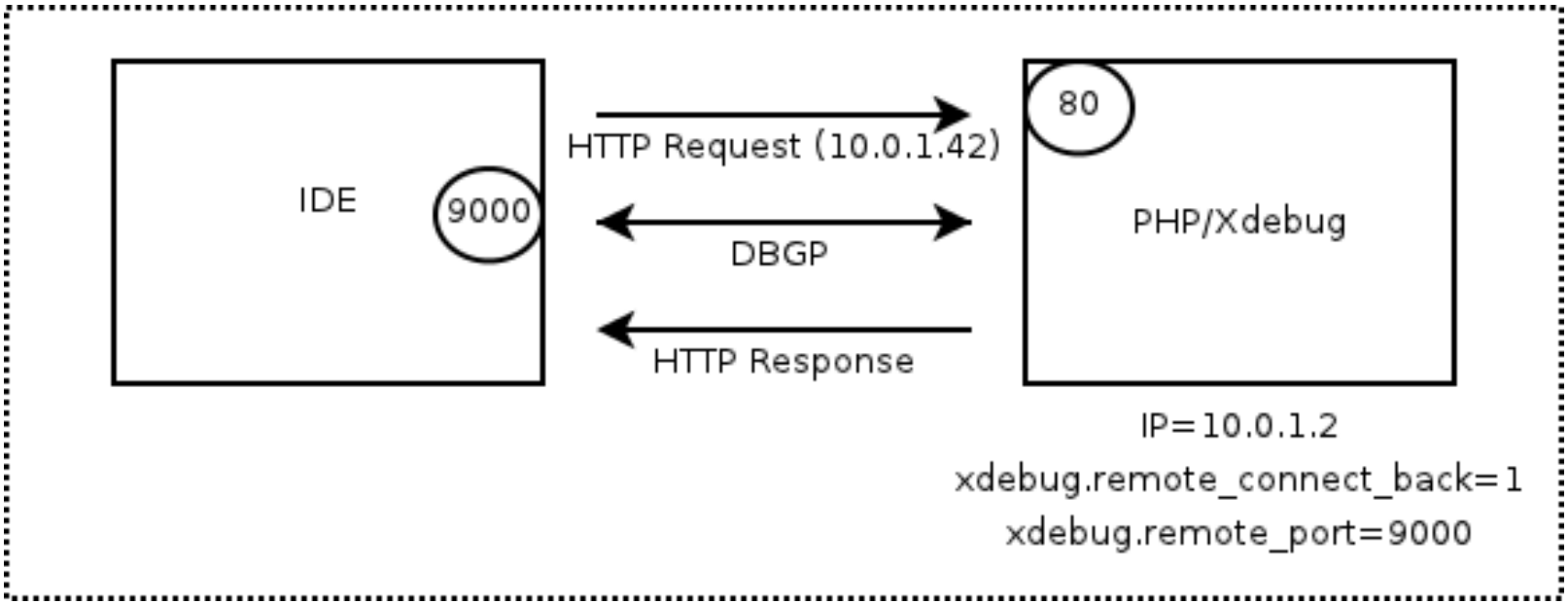
How it works


IDE
9000
HTTP Request (10.0.1.42)
80
PHP/Xdebug
IP=10.0.1.2
xdebug.remote_connect_back=1
xdebug.remote_port=9000

How it works


IDE
9000
HTTP Request (10.0.1.42)
DBGP connect
80
PHP/Xdebug
IP=10.0.1.2
xdebug.remote_connect_back=1
xdebug.remote_port=9000

How it works


IDE
9000
HTTP Request (10.0.1.42)
DBGP
HTTP Response
80
PHP/Xdebug
IP=10.0.1.2
xdebug.remote_connect_back=1
xdebug.remote_port=9000

Browser extensions

- Xdebug Helper for Firefox
- Xdebug Helper for Chrome
- Xdebug Toggler for Safari
- Xdebug launcher for Opera

Browser extensions

- Xdebug Helper for Firefox
- Xdebug Helper for Chrome
- Xdebug Toggler for Safari
- Xdebug launcher for Opera

or

- HTTP GET/POST variable XDEBUG_SESSION_START
- Set a Cookie in the header XDEBUG_SESSION

VS Code launch.json

Install Extension: PHP Debug

VS Code launch.json

Install Extension: PHP Debug 

VS Code launch.json

Install Extension: PHP Debug 

```
{
    "name": "Listen for XDebug",
    "type": "php",
    "request": "launch",
    "port": 9000,
    "pathMappings": {
        "/var/www/html": "${workspaceRoot}"
    }
}
```

VS Code launch.json

This will be automatically generated

```
{
    "name": "Launch currently open script",
    "type": "php",
    "request": "launch",
    "program": "${file}",
    "cwd": "${fileDirname}",
    "port": 9000
}
```

Installation on Ubuntu with PHP-FPM and Nginx

1. `apt-get install php-xdebug`

Installation on Ubuntu with PHP-FPM and Nginx

1. `apt-get install php-xdebug`
2. `phpenmod xdebug`

Installation on Ubuntu with PHP-FPM and Nginx

1. `apt-get install php-xdebug`
2. `phpenmod xdebug`
3. `systemctl restart php7.3-fpm`

Installation on Ubuntu with PHP-FPM and Nginx

1. `apt-get install php-xdebug`
2. `phpenmod xdebug`
3. `systemctl restart php7.3-fpm`

Check your Target

Install on your target machine, e.g. Vagrant or Docker container.
Check the installation with `php -m | grep xdebug`.

Installation on Ubuntu with PHP-FPM and Nginx

1. `apt-get install php-xdebug`
2. `phpenmod xdebug`
3. `systemctl restart php7.3-fpm`

Check your Target

Install on your target machine, e.g. Vagrant or Docker container. Check the installation with `php -m | grep xdebug`.

Common Pitfall

Check your PHP version with `php -v` or with `<?php phpinfo();`

PHP settings / Remote Debugging

```
zend_extension = xdebug.so
xdebug.remote_enable = On
xdebug.remote_connect_back = On
; xdebug.remote_host = 10.0.49.1
```

/etc/php/7.3/mods-available/xdebug.ini

These values are already present in Ubuntu. No need to edit! Make sure you enabled the plugin with `phpenmod`.

PHP settings / Remote Debugging

```
zend_extension = xdebug.so
xdebug.remote_enable = On
xdebug.remote_connect_back = On
; xdebug.remote_host = 10.0.49.1
```

/etc/php/7.3/mods-available/xdebug.ini

These values are already present in Ubuntu. No need to edit! Make sure you enabled the plugin with `phpenmod`.

Tailored Installation Instructions for install from source

https://xdebug.org/wizard.php

Profiler

What does it do?

Xdebugs Profiler is a powerful tool that gives you the ability to analyze your PHP code and determine bottlenecks or generally see which parts of your code are slow and could use a speed boost.

Profiler

What does it do?

Xdebugs Profiler is a powerful tool that gives you the ability to analyze your PHP code and determine bottlenecks or generally see which parts of your code are slow and could use a speed boost.

- analyzes program execution to measure:
  - memory
  - duration
  - frequency of function calls
- generates files for external analysis (cache grind)

Profiler

What does it do?

Xdebugs Profiler is a powerful tool that gives you the ability to analyze your PHP code and determine bottlenecks or generally see which parts of your code are slow and could use a speed boost.

- analyzes program execution to measure:
  - memory
  - duration
  - frequency of function calls
- generates files for external analysis (cache grind)

Alternatives: XHprof and Blackfire

Cachegrind Analysis Tools

- Output Files: **cachegrind.out.%p**

Cachegrind Analysis Tools

- Output Files: **cachegrind.out.%p**
- Analysis Tools
  - KCacheGrind (Linux)
  - QCacheGrind (Linux, Mac)
  - WinCacheGrind (Windows)
  - Webgrind (web-based)

Enable Profiler

```
; append this to xdebug.ini
xdebug.profiler_enable_trigger = On
xdebug.profiler_output_dir = /tmp/

; don't do this
; xdebug.profiler_enable
```

Enable Profiler

```
; append this to xdebug.ini
xdebug.profiler_enable_trigger = On
xdebug.profiler_output_dir = /tmp/

; don't do this
; xdebug.profiler_enable

systemctl restart php7.3-fpm
```

Enable Profiler

```
; append this to xdebug.ini
xdebug.profiler_enable_trigger = On
xdebug.profiler_output_dir = /tmp/

; don't do this
; xdebug.profiler_enable
```

```
systemctl restart php7.3-fpm
```

or

- Set HTTP GET/POST variable XDEBUG_PROFILE
- Set a Cookie in the header XDEBUG_PROFILE

Function Traces

What does it do?

Those so-called *function traces* can be a help for when you are new to an application or when you are trying to figure out what exactly is going on when your application is running. The function traces can optionally also show the values of variables passed to the functions and methods, and also return values.

Configuration

```
xdebug.trace_enable_trigger = On
xdebug.trace_output_dir = /application/log
; xdebug.auto_trace   ; trigger tracer every pageload
```

Resources & Further Readings

- Official Xdebug Documentation https://xdebug.org/docs
- Tailored Installation Instructions https://xdebug.org/wizard.php

https://github.com/5queezer/xdebug-slides/blob/master/slides.pdf