



PHP Starter Day

Debugging with Xdebug

International PHP Conference 2008, Mainz
Tobias Schlitt <toby@php.net>



Agenda

- About Xdebug
- Installation
- Development goodies
- Tracing
- Profiling
- Remote debugging
- Testing
- Q & A

Agenda

- **About Xdebug**
- Installation
- Development goodies
- Tracing
- Profiling
- Remote debugging
- Testing
- Q & A

Why a debugger?

- „I don't need a debugger“
 - ▷ There is no bug free code
 - ▷ Make live easier, use a debugger!
- „var_dump(), print_r() and echo are sufficient“
 - ▷ In a lot of cases they are
 - ▷ Debugging with them is slow and a lot of work
- „Using a debugger means using an IDE“
 - ▷ No. But you can.

What is Xdebug?

- Open Source debugger for PHP
- PHP (Zend Engine) extension
- Works (at least) on Linux, Mac and Windows
- About 4 years old
- Version 2 (stable) released last year
- Created and maintained by Derick Rethans
- <http://xdebug.org>

Xdebug features

- Enhances your daily work with PHP
- Allows to trace PHP program runs
- Can profile your PHP applications
- Provides code coverage data (for e.g. PHPUnit)
- Allows remote-debugging with an external client
 - Including features like break-points, stepping,...

Agenda

- About Xdebug
- **Installation**
- Development goodies
- Tracing
- Profiling
- Remote debugging
- Testing
- Q & A

PEAR (*nix)

- The easiest way to install Xdebug
- Working PEAR installation assumed:

```
$ pecl install xdebug
```
- The PEAR Installer takes the necessary steps:
 - ▷ Downloading the source
 - ▷ Compiling the module
 - ▷ Copying the module to the right directory
- After that: Add the module to your php.ini

Manual (*nix)

- The (not that) hard way to install Xdebug
- **Manual compilation:**

```
wget http://xdebug.org/link.php?url=xdebug201
tar -xzf xdebug-2.0.2.tgz
cd xdebug-2.0.2
phpize
./configure --enable-xdebug
      --with-php-config=/usr/bin/php-config
make
cp modules/xdebug.so /<module_path>/xdebug.so
```
- **After that: Add the module to your php.ini**

Windows

- Download binary module for your PHP version from
 - ▷ <http://xdebug.org>
 - ▷ http://pecl4win.php.net/ext.php/php_xdebug
- Copy module into your modules directory
- After that: Add the module to your php.ini

Loading the module

- Xdebug is not a „normal“ extension
- To enable it, add to your php.ini:
 - `zend_extension = "/path/to/xdebug.so"`
- On Windows use `zend_extension_ts`
- For debug builds use `zend_extension_debug`
- Check „`php -v`“ or `phpinfo()`; for Xdebug signature

Pitfalls

- The `extension_dir` directive does not take effect!
- Binary modules (Windows) do not work with PHP debug builds
- `--enable-versioning` prevents loading
- Other zend-extensions interfere with Xdebug

Configuration

- Huge variety of options
- Many shown here, some not
- Refer to manual on <http://xdebug.org>
- Most of Xdebugs options can be adjusted using `ini_set()` at runtime, except for
 - `xdebug.extended_info`
 - `xdebug.profiler_ [...]`

Agenda

- About Xdebug
- Installation
- **Development goodies**
- Tracing
- Profiling
- Remote debugging
- Testing
- Q & A

Everyday development

- Xdebug offers new debugging features
- But also improves the daily work with PHP
- Simply
 - ▷ switch it on
 - ▷ tune some settings
 - ▷ work as usual

Error messages

- PHP error messages are mostly useless
 - ▷ Very few information on the error location
 - ▷ Almost no information about affected data
 - ▷ No information about the code context
- Xdebug enhances them for you!
- Let's take a look...



Error messages

Live demo

Dumping data

- Using `var_dump()`
 - ▷ Everyone does it
 - ▷ There is nothing bad about it
- `var_dump()` in the browser sucks
- Xdebug enhances `var_dump()`
- Let's take a look...

Infinity

- Endless recursions
 - ▷ Extremely hard to find
 - ▷ Usually your script just times out or worse
- Xdebug can protect you from this
 - ▷ Note: Not from endless runs of loops!
- Let's take a look...

Useful functions 1 / 3

- `xdebug_[en/dis]able()`
 - ▷ Manually switch stack traces on or off
- `xdebug_call_[class/function/file/line]()`
 - ▷ Get the call point of the currently running function
- `xdebug_dump_superglobals()`
 - ▷ Dump super-globals as specified by INI setting

Useful functions 2/3

- `xdebug_get_declared_vars()`
 - ▷ Returns an array containing the names of all variables in the current scope
- `xdebug_get_function_stack()`
 - ▷ Returns the function stack trace as an array
- `xdebug_get_stack_depth()`
 - ▷ Get the current depth in the function stack
 - ▷ Note: includes make also a level!

Useful functions 3/3

- `xdebug_time_index()`
 - ▷ Returns the seconds since script run start
- Let's take a look...

Agenda

- About Xdebug
- Installation
- Development goodies
- **Tracing**
- Profiling
- Remote debugging
- Testing
- Q & A

What is a trace?

- Checks the control flow of an application
 - ▷ Correct function calls
 - ▷ Correct data
- Determine if input data produces correct workflow
 - ▷ Traces are (always) dependent on input data
 - ▷ It is (almost) possible to create a trace for every control flow of a script

Tracing

- Many developers trace like this:

```
echo „Here I am!!!“
```

```
// ...
```

```
echo „Now I'm here!!! Something is:“
```

```
var_dump($something);
```

- Works well in small scripts
- Is a horror in larger apps / libraries
- Debugging like this is a sh**load of work
- Doing so with 3rd party code is almost impossible

How to create a trace

- Using Xdebug functions
 - ▷ `xdebug_start_trace()`
 - ▷ `xdebug_stop_trace()`
- Automatically (`php.ini`)
 - ▷ `xdebug.auto_trace = "1"`
 - ▷ Cannot be set via `ini_set()`!
- Let's take a look...

Notes on tracing

- Never trace in production!
 - ▷ Tracing is extremely slow!
- grep & friends help a lot to dig into traces
- Tuning your configuration makes traces more usable
- Note: More information through Xdebug mean
 - ▷ Slower performance
 - ▷ Higher memory consumption

Agenda

- About Xdebug
- Installation
- Development goodies
- Tracing
- **Profiling**
- Remote debugging
- Testing
- Q & A

Profiling

- Profiling means „performance analysis“
- Xdebug can generate a profile for an application run
- The generated profile does not help you much, yet!
- Tools for visualization
 - ▷ Linux: KCacheGrind
 - <http://kcacheGrind.sourceforge.net/>
 - ▷ Windows: WincacheGrind

How to profile

- Switch on the profiler settings and start
- Use the [X]CacheGrind tool of your choice to analyze
 - ▷ Timing information of your code (functions/methods)
 - ▷ Call graphs
- Let's take a look...

Agenda

- About Xdebug
- Installation
- Development goodies
- Tracing
- Profiling
- Remote debugging
- Testing
- Q & A

Remote debugging

- Xdebug can also deal as a “real debugger”
 - ▷ as you might know it from IDEs
- An external debugging client is needed for this
 - ▷ Several debugging protocols supported
 - ▷ Xdebug ships with a very, very simple command line client
 - ▷ Integration into several commercial and non-commercial tools is available

Debugging clients

- We'll now take a look at some of these clients:
 - ▷ xdebugclient (Linux, free)
 - ▷ Protoeditor (Linux, free)
 - ▷ Komodo IDE (Linux/Mac/Windows, commercial)
 - ▷ VIM Plugin (Linux, free)
- All of these allow debugging using breakpoints, variable examination and more...

xdebugclient

- <http://xdebug.org> (shipped with Xdebug)
- Console based
- Binaries available on xdebug.org
- Allows to communicate with Xdebug via DBGp protocol directly
 - ▷ Just for testing purposes
 - ▷ Very rudimentary and uncomfortable
 - ▷ Can help you to develop your own client for Xdebug

Protoeditor

- <http://protoeditor.sourceforge.net/>
- KDE based editor for PHP, Perl and Python
- Can also deal with other PHP debuggers
- Based on KDE standard editor Kate
- Allows local and remote debugging with Xdebug

Komodo IDE

- <http://www.activestate.com/Products/kor>
- Commercial (21 day trial available)
- Multi-language IDE (Perl, PHP, Python, Ruby, Tcl,...)
- Allows remote-debugging via Xdebug only
- Do not confuse with Komodo Editor (free, but no debugging support)

VIM Plugin

- <http://www.vim.org/scripts/script.php?sci>
- DBGp client
 - ▷ Possibly usable with other DBGp implementations
 - ▷ Tested only with Xdebug
- Integrates directly into VIM
- Gives many remote features provided by “real IDEs”
- Still looks unstable (hopefully it works

Notes

- Manually set a break point:
 - ▷ `xdebug_break()`
Emits a break point for the remote debugger
- Xdebug affects the performance itself
 - ▷ Switch off as much of Xdebug as you can

Agenda

- About Xdebug
- Installation
- Development goodies
- Tracing
- Profiling
- Remote debugging
- **Testing**
- Q & A

Code coverage

- Xdebug allows you to retrieve code coverage information
 - ▷ `xdebug_start_code_coverage()`
 - ▷ `xdebug_get_code_coverage()`
Returns an array of code coverage information
- Useful for testing purposes
- PHPUnit (<http://phpunit.de>) can make use of this to generate nice stats
- Let's take a look...

Agenda

- About Xdebug
- Installation
- Development goodies
- Tracing
- Profiling
- Remote debugging
- Testing
- Q & A

Q & A

- Are there any questions left?
- Did you miss anything?
- Did you like this session?
 - ▷ Ideas?
 - ▷ Critics?
 - ▷ Recommendations?
 - ▷ Wishes?

The end...

- Thanks for listening!
- Slides and code online soonish
 - ▷ Watch: <http://blog.schlitt.info>
- Later questions: toby@php.net

License

- This set of slides and the source code included in the download package is licensed under the

Creative Commons Attribution-Noncommercial-Share Alike 2.0 Generic License

- <http://creativecommons.org/licenses/by-nc-sa/2.0/deed.en>