Xdebug

Daily helper and professional debugger

TobiasSchlitt <toby@php.net>

PHP@FrOSCon 2009

2009-08-23

About me

- Tobias Schlitt <toby@php.net>
- PHP since 2001
- Freelancing consultant
- Qualified IT Specialist
- Studying CS at TU Dortmund
  (expect to finish this year)
- OSS addicted
  - PHP
  - eZ Components
  - PHPUnit

Overview

Outline

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 = 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 5 years old
- Current stable version: 2.0.3
- Created and maintained by Derick Rethans
- `http://xdebug.org`

Xdebug features

- Enhance daily work with PHP
- Trace PHP program runs
- Profile PHP applications
- Analyze code coverage (e.g. PHPUnit)
- Remote-debugging with an external client
  Including break-points, stepping, ...

Outline

Via PEAR

- Easiest way to install
- Working PEAR assumed:

  ```
  $ pear install pecl/xdebug
  ```

- PEAR Installer performs necessary steps:
  1. Download source
  2. Compiling module
  3. Copying module to destination
- After that: Add module to php.ini
- Only on *nix systems

Manual

- Compile by hand
- Typical PHP module compile process

```
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 module to php.ini
- Only on *nix systems

Windows

- Download binary module from
  - `http://xdebug.org`
  - `http://pecl4win.php.net/ext.php/php_xdebug.dll`
- Copy module to modules directory
- After that: Add module to php.ini

Add module to php.ini

- Not a “normal” extension
- Add to php.ini:

  `zend_extension = "/path/to/xdebug.so"`
- On Windows: `zend_extension_ts`
- For debug builds `zend_extension_debug`
- Check `$ php -v` or `phpinfo();`

Common pitfalls

- 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
- Most shown later, some not
- Overview: `http://xdebug.org/docs/all_settings`
- Most options adjustably with ini_set () at runtime, except for
  - xdebug.extended_info
  - xdebug. profiler_ *

Outline

Getting started

- Install module
- Switch it on
- Tweak some settings
- Work as usual

Error messages

- PHP error messages not very useful
  - Sometimes poor error location
  - Almost no info on affected data
  - No info on code context
- Xdebug enhances that for you!

Warnings

Functions producing warnings

```
function openSomeFile()
{
    return openMyFile( '/path/to/file', 'Reason to open file.
        ', array( 23, 42, 'Random data.' ) );
}

function openMyFile( $path, $reason, $data )
{
    return fopen( $path, 'r' );
}

function performOperationOnFile( $file )
{
    return fread( $file, 1024 );
}
```

Warnings

Triggering a warning

```
require_once 'warning/functions.php';

function performAComplexOperation()
{
    $file = openSomeFile();
    return performOperationOnFile( $file );
}

echo performAComplexOperation();
```

Warnings

Live

Let's look what it does

Producing an exception

```
require_once '/home/dotxp/dev/PHP/ezcomponents/trunk/Base/src
    /base.php';

function __autoload( $className )
{
    return ezcBase::autoload( $className );
}

$pop3 = new ezcMailPop3Transport( "localhost" );
```

Errors

Live

Let's look what it does

var_dump()

- Using var_dump()
  - Everyone does it
  - There is nothing bad about it
- var_dump() in the browser sucks
- Xdebug enhances var_dump()

var_dump()

- Using var_dump()
  - Everyone does it
  - There is nothing bad about it
- var_dump() in the browser sucks
- Xdebug enhances var_dump()

Live

Let's look what it does

Infinite loops

- Endless recursions
- Extremely hard to find
- Script just times out . . . or worse!
- Xdebug can protect you
- Note: Not from endless runs of loops!

Infinite recursion

Producing an infinite recursion

```
function doubleFaculty( $num )
{
    /*
    if ( $num < 3 )
    {
        return $num;
    }
    */
    return $num * doubleFaculty( $num - 2 );
}

echo doubleFaculty( 10 );
```

Infinite recursion

Live

Let's look what it does

Useful functions

- 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

- 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 also create a level!

Useful functions

- xdebug_time_index()
  Returns the seconds since script run start

Declared vars

Getting declared variables

```
function foo( $a, $b )
{
    global $c;
    $d = 'I am local';
    var_dump( xdebug_get_declared_vars() );
}

$c = 'I am global';

foo( 23, 42 );
```

Declared vars

Live

Let's look what it does

Stack depth

Retrieving stack level

```
echo "Stack depth in " . __FILE__ . ": " .
    xdebug_get_stack_depth() . "\n\n";

function foo()
{
    echo "Stack depth in " . __FUNCTION__ . ": " .
        xdebug_get_stack_depth() . "\n\n";
    bar();
}

include 'stack_depth/deeper_stack.php';

echo "Calling foo() from " . __FILE__ . "\n\n";
foo();
```

Stack depth

Live

Let's look what it does

Outline

What is tracing?

- Check control flow of application
  - Correct function calls
  - Correct data
- Does input data produce correct workflow?
  - Traces always depend on input data
  - Almost impossible to create traces for every possible control flow

Tracing

- Some developers trace like this:

```php
echo "Here I am!!!";
//  ...
echo "Now I'm here!!! Something is:";
var_dump($something);
```

- Works well in small scripts
- Horror in larger apps / libraries
- Debugging like this is a sh**load of work
- Almost impossible with 3rd party code

Tracing with Xdebug

- By functions:
  - xdebug_start_trace ()
  - xdebug_stop_trace()
- Automatically (php.ini)
  - xdebug.auto_trace = "1"
  - Cannot be set via ini_set ()!

Tracing example

A simple trace

```
function doubleFaculty( $num )
{
    if ( $num < 3 )
    {
        return $num;
    }
    return $num * doubleFaculty( $num - 2 );
}

xdebug_start_trace( 'traces/60_simple_trace.xt' );

echo doubleFaculty( 10 );
```

Tracing example

Live

Let's look what it does

Notes on tracing

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

Outline

Profiling

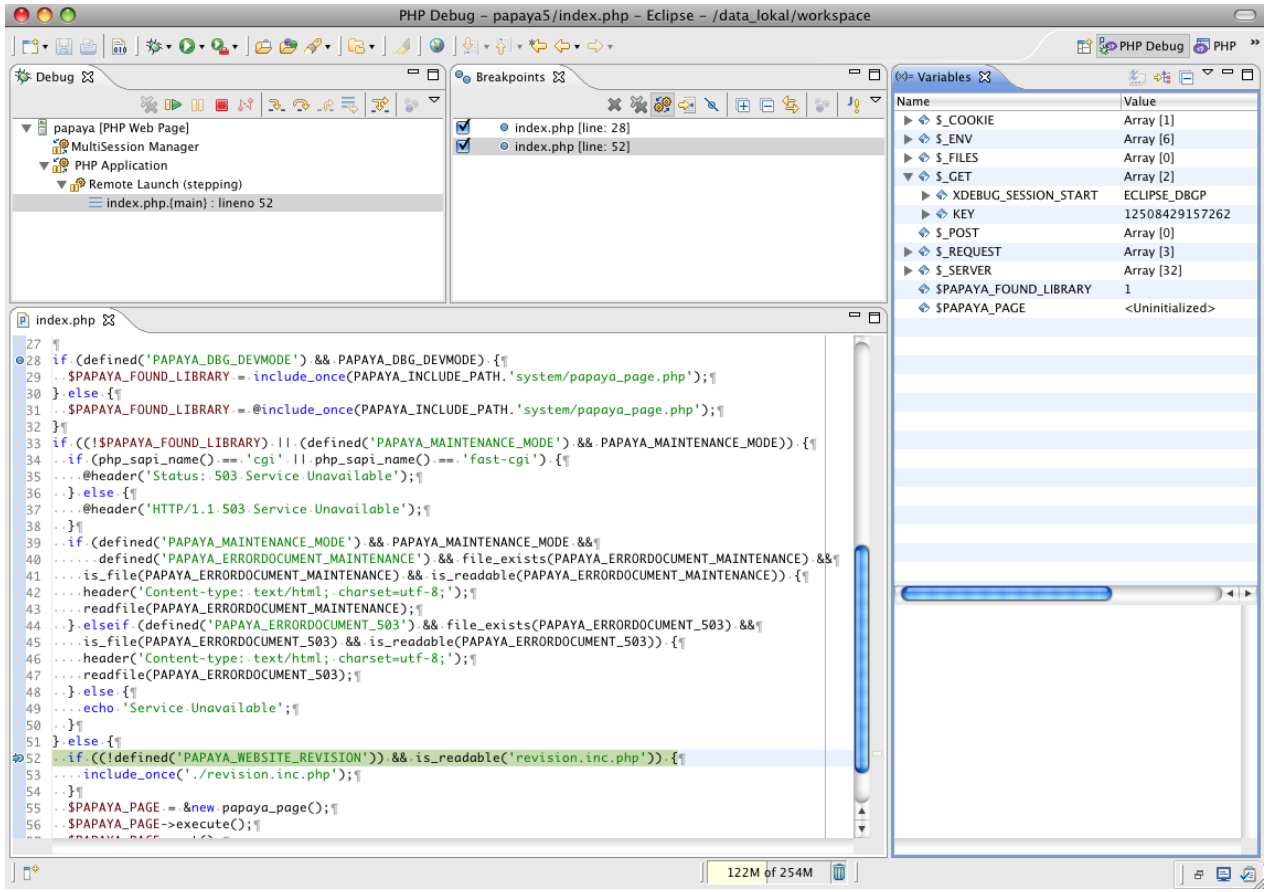
- Which code takes which time?
- Find bottle necks in code
- Check your database queries and setup first!

Code coverage

- Mostly used in testing
- Which code is not covered by a test?
- PHPUnit can utilize this
  `http://www.phpunit.de`

Remote debugging

- Integrate Xdebug into your IDE
  - Komodo IDE
  - Eclipse



The end

Thank you for listening!

The end

Thank you for listening!

- Are there any questions left?

The end

Thank you for listening!

- Are there any questions left?
- I hope you learned what you expected?

The end

Thank you for listening!

- Are there any questions left?
- I hope you learned what you expected?
- Contact me: Tobias Schlitt <toby@php.net>

The end

Thank you for listening!

- Are there any questions left?
- I hope you learned what you expected?
- Contact me: Tobias Schlitt <toby@php.net>
- Slides will be up on Slideshare