

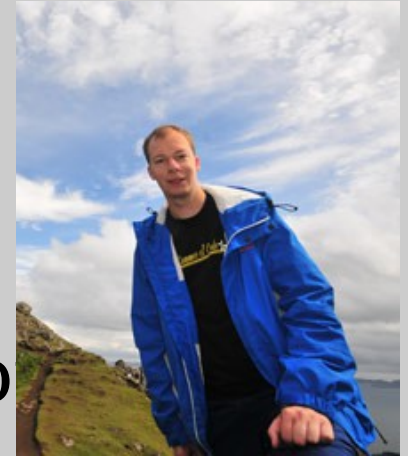
Welcome!



ZendCon - Santa Clara, US - Oct 18th, 2011
Derick Rethans - derick@derickrethans.nl - twitter:
@derickr
<http://joind.in/3792>

Derick Rethans

- Dutchman living in London
- PHP development
- Freelancer doing PHP internals development
ie.: writing extensions for a living
- Anderskor's Camouflage
- Author of Xdebug
- Author of the `mcrypt`, `input_filter`, `dbus`, `translit` and `date/time` extensions
- Contributor to the Apache Zeta Components Incubator project (formerly eZ Components)



Development Aid
Live Debugging
Profiling

Xdebug Pimps Your Ride



Error Messages

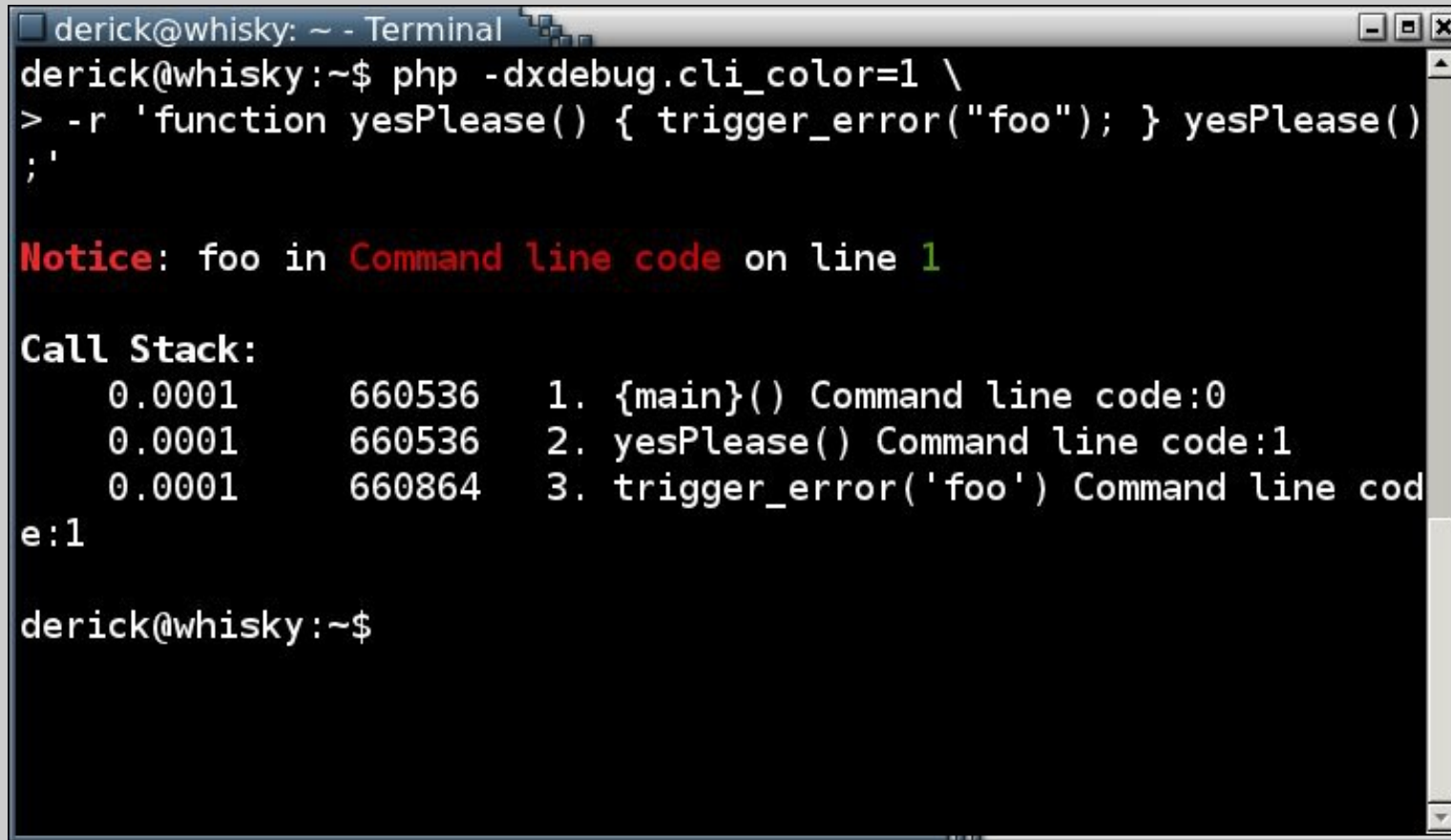
Record and Delayed Display

xdebug_start_error_collection()
xdebug_stop_error_collection()
xdebug_get_collected_errors()

```
<?php
xdebug_start_error_collection();
?>
<html>
<body>
<div id='main'>
...
</div>
<div id='errors'>
    <?php echo xdebug_get_collected_errors(); ?>
</div>
</html>
```


Colours on the Command Line

new in 2.2

A terminal window titled "derick@whisky: ~ - Terminal" showing a PHP CLI command and its output. The command is "php -dxdebug.cli_color=1 \> -r 'function yesPlease() { trigger_error(\"foo\"); } yesPlease();'". The output shows a notice: "Notice: foo in Command line code on line 1" and a call stack with three entries: 1. {main}() Command line code:0, 2. yesPlease() Command line code:1, and 3. trigger_error('foo') Command line code:1. The terminal prompt "derick@whisky:~\$" is visible at the bottom.

```
derick@whisky: ~ - Terminal
derick@whisky:~$ php -dxdebug.cli_color=1 \
> -r 'function yesPlease() { trigger_error("foo"); } yesPlease()
;'

Notice: foo in Command line code on line 1

Call Stack:
 0.0001    660536    1. {main}() Command line code:0
 0.0001    660536    2. yesPlease() Command line code:1
 0.0001    660864    3. trigger_error('foo') Command line code:1
e:1

derick@whisky:~$
```

xdebug.cli_color=1

Scream

- PHP's @ operator hides warnings and errors
- `xdebug.scream=1` makes PHP ignore @



Recording headers

- Xdebug collects all headers being set, implicitly and explicitly
- It's very useful for testing and unit-tests
- `xdebug_get_headers()`

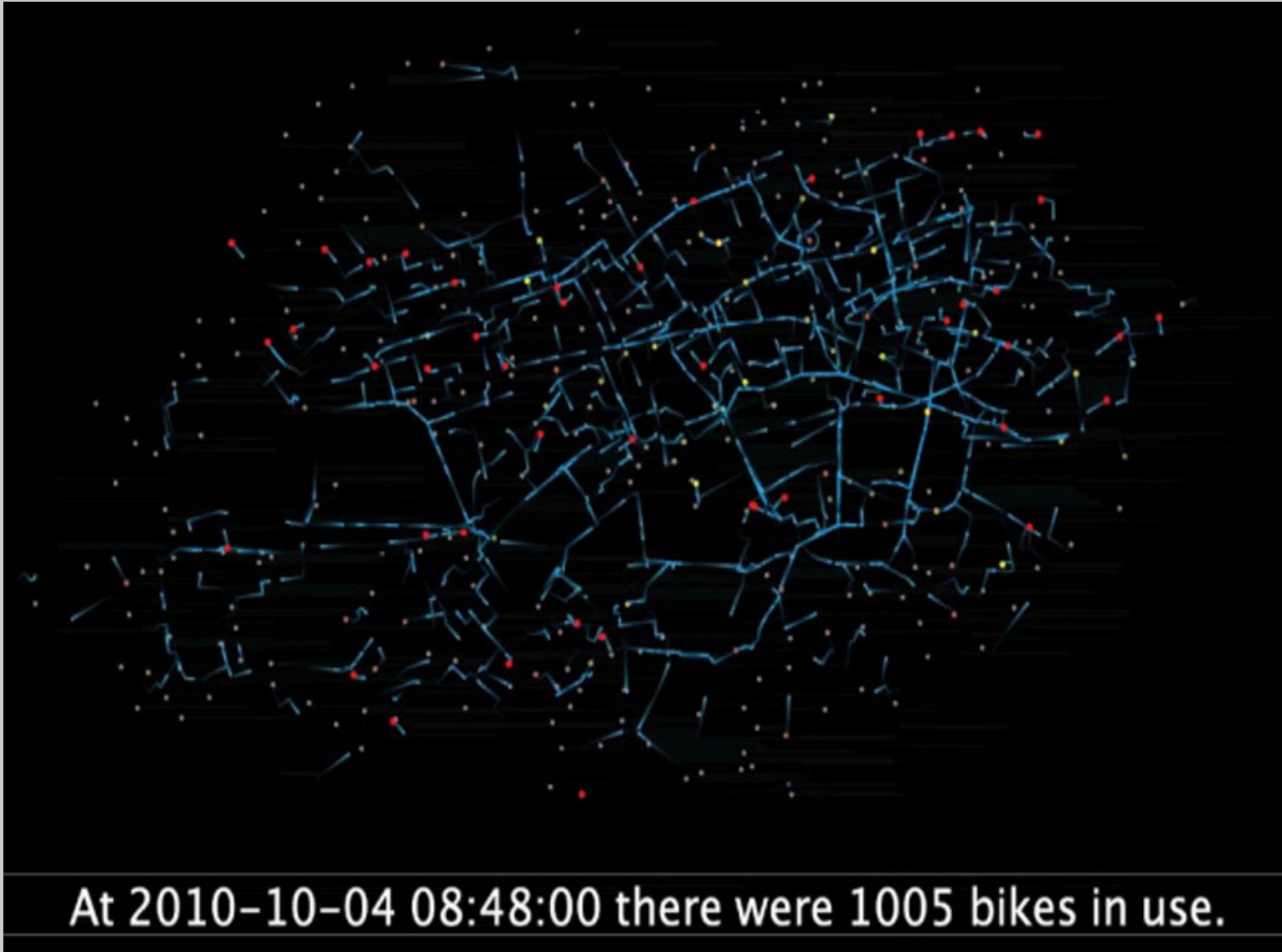
```
<?php
session_start();

setcookie( 'key', 'value', time() + 86400 );

header( "Status: 403" );

var_dump( xdebug_get_headers() );
?>
```

Tracing



Function traces

- Textual traces
- Parsable traces
- HTML traces
- Tracing only parts of an application with `xdebug_start_trace()` and `xdebug_stop_trace()`.
- Fetching the trace file name that is being used with `xdebug_get_tracefile_name()`.
- Changing how much data is shown with `xdebug.var_display_max_children`, `xdebug.var_display_max_data` and `xdebug.var_display_max_depth`.

demo

Live (Remote) Debugging



- DBGp, common Debugging protocol
- Cross-language: PHP, Python, Perl...

Clients:

- Eclipse/PDT (Java based — free)
- Komodo (Linux, Windows, Mac — commercial)
- MacGDBp (free)
- Netbeans (Java-based — free)
- PHPStorm (Java-based — commercial)
- Zend Studio for Eclipse (Eclipse-based — commercial)
- Stand-alone client (Linux (for now)):
- (and many others:)

Activating the Remote Debugger

php.ini settings:

```
xdebug.remote_enable=1
```

```
xdebug.remote_host=localhost
```

```
xdebug.remote_port=9000
```

php.ini settings:

```
xdebug.remote_enable=1
```

```
xdebug.remote_host=localhost
```

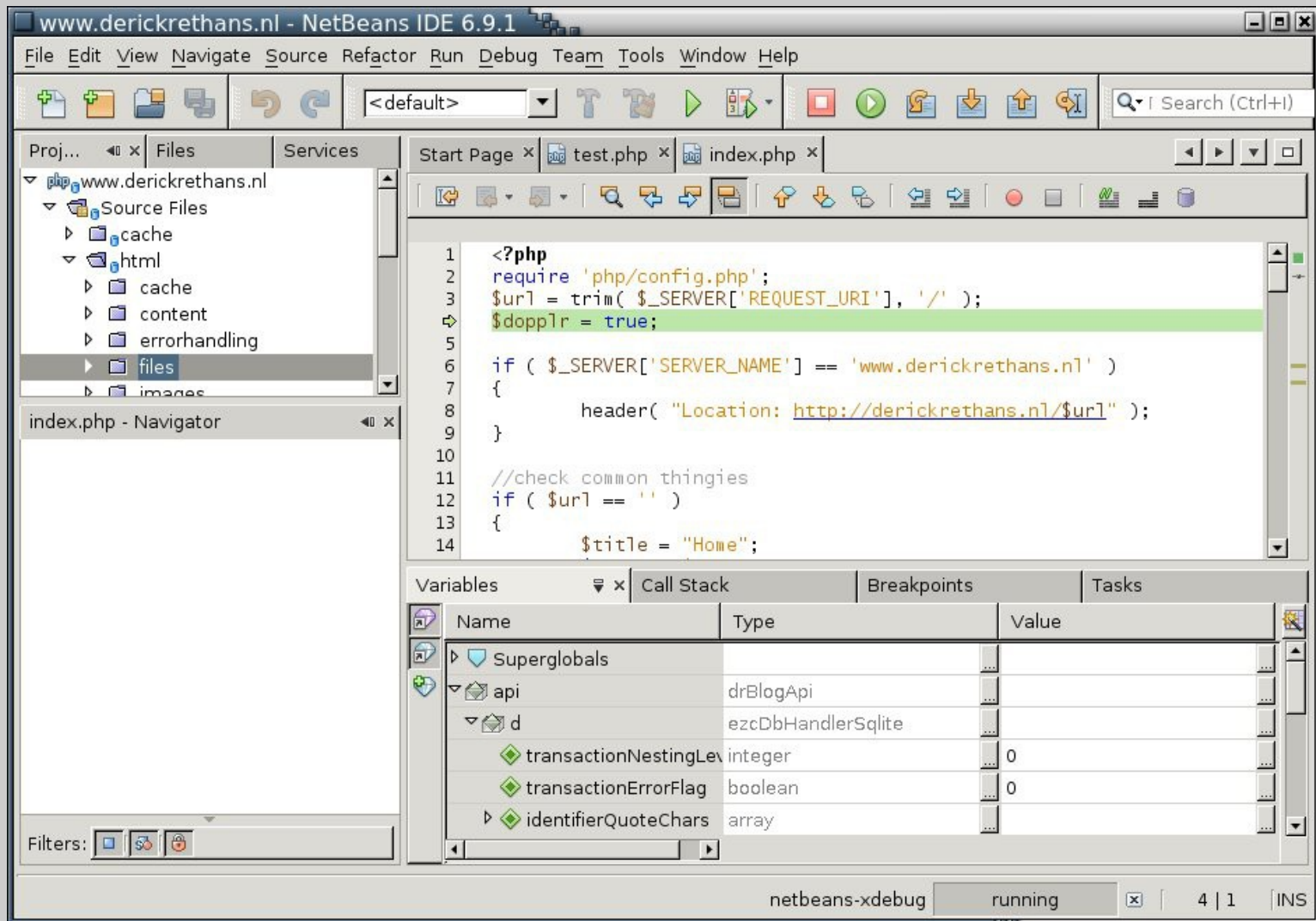
```
xdebug.remote_port=9000
```

```
xdebug.remote_connect_back=1
```

On the shell:

```
export XDEBUG_CONFIG="idekey=xdebugrocks"
```

With browser extensions



test.php (~dev/php/xdebug-demos/cli-debug) - ActiveState Komodo IDE 6.1 - Debugger

File Edit Code Navigation View Debug Project Tools Help

Start Page test.php x

```
4 private $priv;
5 protected $prot;
6 public $publ;
7
8 function __construct($a)
9 {
10     $this->priv = $a * 2;
11     $this->prot = $a * 2.5;
12     $this->publ = $a * M_PI;
13 }
14
15
16 function foo($a)
17 {
```

Break at /home/derick/dev/php/xdebug-demos/cli-debug/test.php, line 11.

Name	Type	Value
a	float	2.718281828459
▼ this	TheClass	
priv	float	5.4365636569181
prot	null	
publ	null	

Watch Locals Superglobals

Output Call Stack HTML

Ready ● __construct UTF-8 Ln: 11 Col: 1 PHP

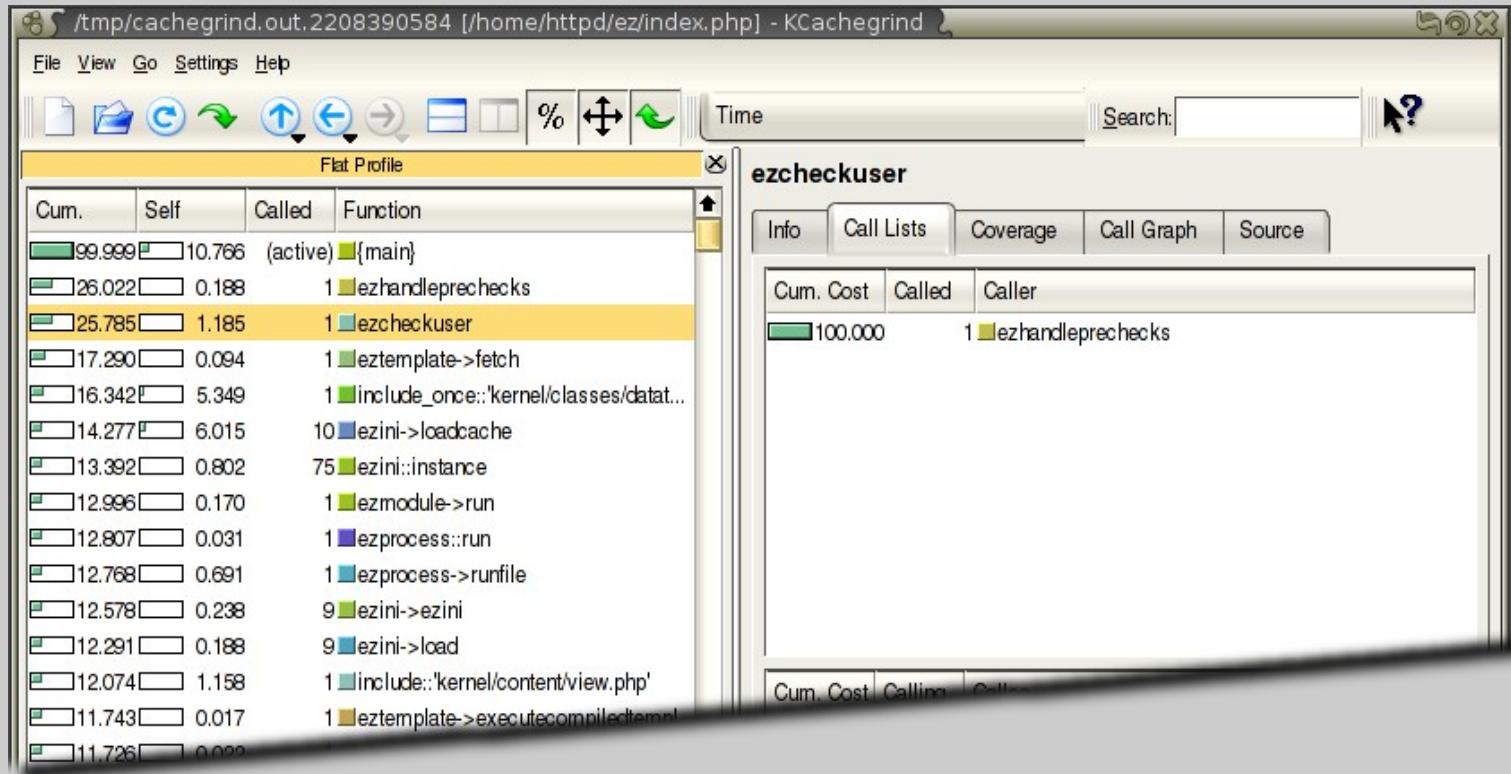
demo

Profiling



Profiling

KCacheGrind's Flat Profile and Call List

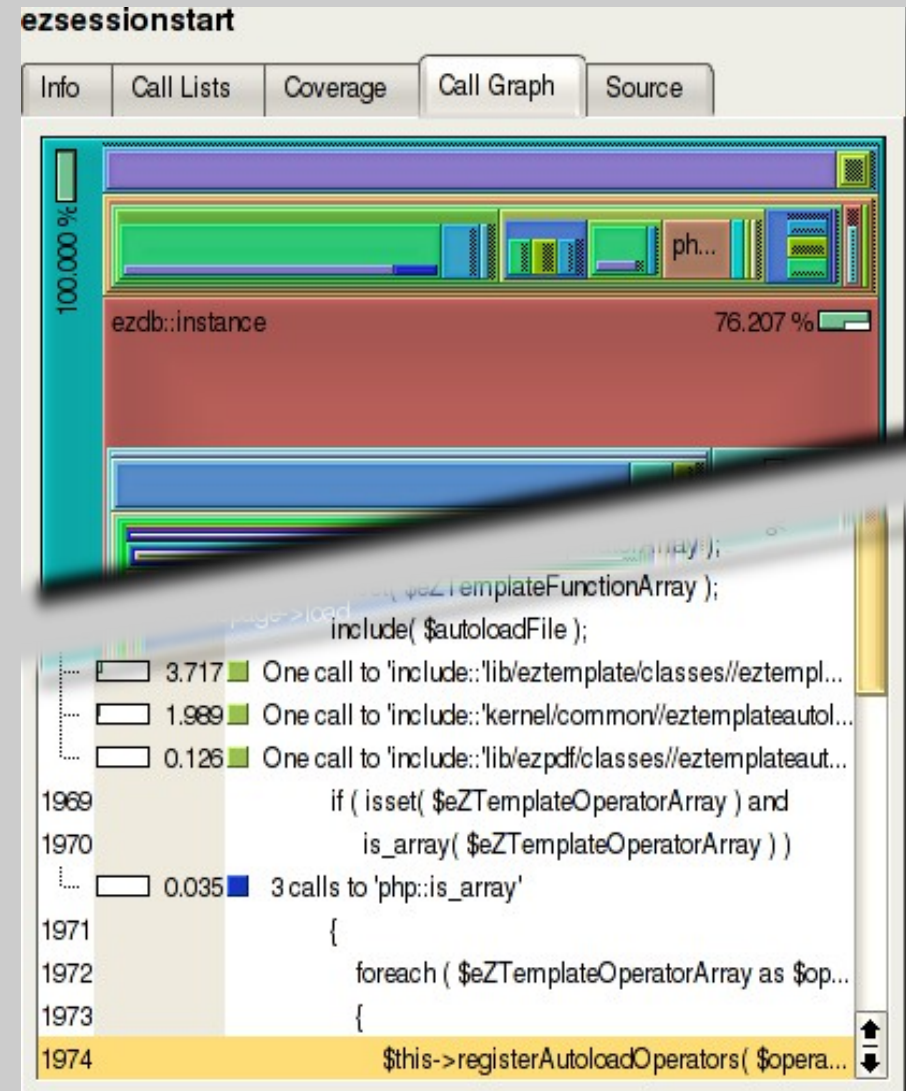


```
xdebug.profiler_enable=1           ; enable profiler  
xdebug.profiler_output_dir=/tmp     ; output directory  
xdebug.profiler_output_name=cachegrind.out.%p
```

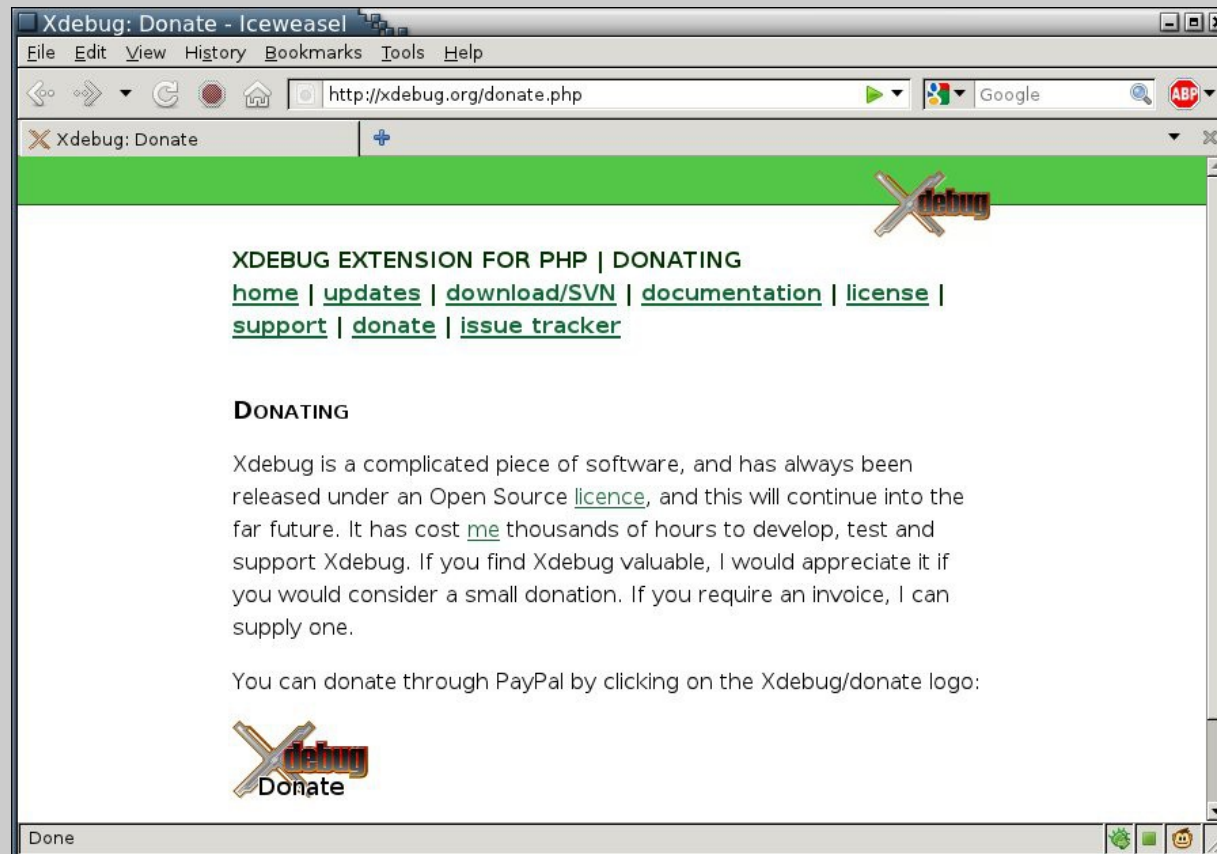
Profiling

KCacheGrind's Call Graph and Source Annotations

- Call graph
- Area shows time spend
- Stacked to show callees
- Source annotations
- Number of calls
- Total time per function



- It's Open Source and free (as in "free beer")
- Working on Xdebug takes up a lot of spare time
- I don't have a lot of spare time





- Xdebug site: <http://xdebug.org>
- Xdebug documentation:
<http://xdebug.org/docs.php>
- DBGp specification: <http://xdebug.org/docs-dbgp.php>
- Rate this talk! <http://joind.in/3792>
- If you like Xdebug: <http://xdebug.org/donate.php>
- These slides: <http://derickrethans.nl/talks.html>