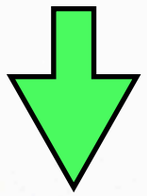
Hidden Features of PHP

Ilia Alshanetsky
@iliaa

__DIR__ Magic

The __DIR__ constant is a simple and fast solution to the "where am i?" question for PHP scripts.

```
ilia@s3 /tmp $ php a.php
```



```
<?php echo __DIR__;
```

```
/tmp
```

We ♥ Perl

Allows quick retrieval of a non-empty value from 2 values and/or expressions

```
$a = true ?: false; // true
$a = false ?: true; // true
$a = "" ?: 1; // 1
$a = 0 ?: 2; // 2
$a = array() ?: array(1); // array(1)
$a = strlen("") ?: strlen("a"); // 1
```

** The variable or array key must exist

GOTO ...

```
restart:
if (error_condition1) {
     goto error;
}
if (error_condition2) {
     goto restart;
}

error:
     report_error();
     exit;
```

My favourite 5.3 feature ;-)

Encryption Functions

```php
$pwd = 'very secret';
$data = 'test 123';
// over 50 supported algorithms
foreach (openssl_get_cipher_methods() as $v) {
    // really bad iv generation
    $iv = substr(md5(time()), 0, openssl_cipher_iv_length($v));

    // encrypt
    $enc = openssl_encrypt($data, $v, $pwd, false, $iv);

    // decrypt
    $dec = openssl_decrypt($enc, $v, $pwd, false, $iv);
}
```

Double Encoding

Prevent double encoding of html-entities via 4th argument to htmlspecialchars() and htmlentities()

```
$foo = "bar > foo &amp; that&quot;s all";

htmlspecialchars($foo, ENT_COMPAT, 'UTF-8');
htmlentities($foo, ENT_COMPAT, 'UTF-8');
```

bar > foo **&** that**"**s all

Double Encoding

Prevent double encoding of html-entities via 4th argument to htmlspecialchars() and htmlentities()

```
$foo = "bar > foo &amp; that&quot;s all";

htmlspecialchars($foo, ENT_COMPAT, 'UTF-8');
htmlentities($foo, ENT_COMPAT, 'UTF-8');
```

bar > foo **&** that**"**s all

```
htmlspecialchars($foo, ENT_COMPAT, 'UTF-8', false);
htmlentities($foo, ENT_COMPAT, 'UTF-8', false);
```

bar > foo **&** that**"**s all

Date Parsing

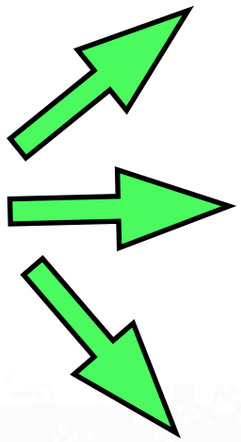
October 5, 2012

05-10-12 → May 10, 2012

December 10, 2005

Date Parsing

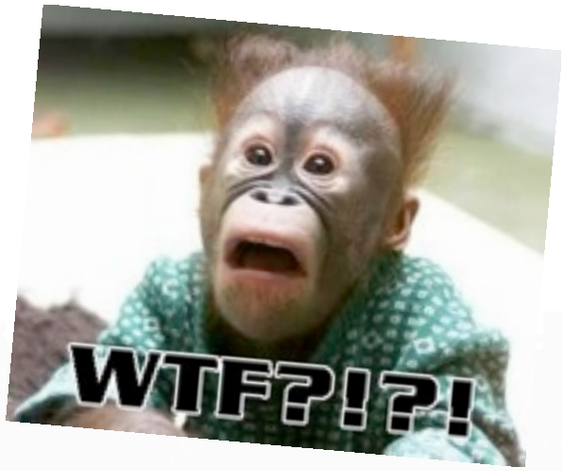
05-10-12



October 5, 2012

May 10, 2012

December 10, 2005

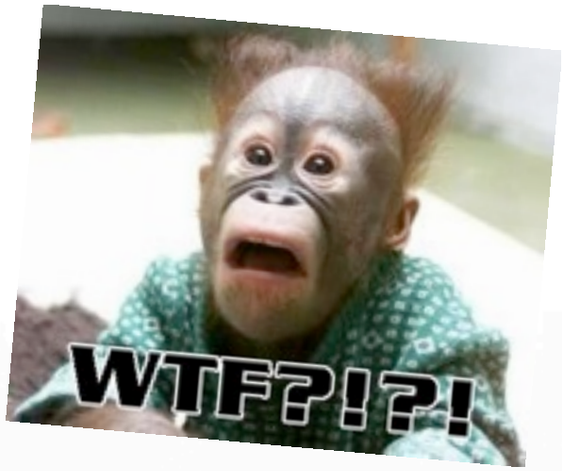


Date Parsing

October 5, 2012

05-10-12 → May 10, 2012

December 10, 2005



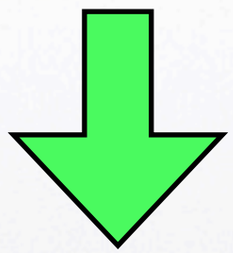

```php
$date =
date_create_from_format('y-m-d', '05-10-12');

var_dump(date_format($date, 'F d, Y'));
```

string(16) "October 12, 2005"

Dude, where is my code?

PHP does a lot of magic to resolve partial file paths for include/require. Now you can too.

```
stream_resolve_include_path("PEAR.php");
```



/usr/share/php/PEAR.php

session ini magic

- Improve randomness of session id via the use of /dev/urandom

```
session.entropy_file = /dev/urandom
session.entropy_length = 32
```

- Secure your session cookies from JavaScript

```
session.use_only_cookies = 1
session.cookie_httponly = 1
```

mail logging

Want to know what scripts are sending out e-mail?
Well, now you can!

;;This will log every mail() call
mail.log = /path/to/file

```
mail() on [/tmp/script.php:2]: To: ilia@ilia.ws -- Headers:
```

;;Adds X-PHP-Originating-Script header
;; Contains UID & filename of the script
mail.add_x_header = On

```
X-PHP-Originating-Script: 1000:script.php
```

Better Hashing

A built-in PHP mechanism for generating HMAC (Hash-based Message Authentication Code) secured hashes for many algorithms

```php
// 8b266369505f0c90bf193c856aa8f49dc87a759088fd31f28217e4db42a5ac4c
echo hash_hmac('sha256', '1337PwD', 'secretkey') , "\n";

// d82bba393cb2f5e38a4021efc30d43883b1e0a40e30d8ed1c89e7ec263023ec0
echo hash_hmac_file('sha256', __FILE__, 'secretkey') , "\n";
```

SPL FS Tricks

Simple recursive directory traversal

```
foreach (
  new RecursiveIteratorIterator(
    new RecursiveDirectoryIterator('.')
  ) as $file)
{
    echo $file , "\n";
}
```

SPL FS Tricks

Recursive directory traversal with pattern matching

```php
$it = new RecursiveIteratorIterator(
    new RecursiveDirectoryIterator('.')
);
$regx = new RegexIterator(
    $it,
    '/^.*\.php$/i', // returns only matching text
    RecursiveRegexIterator::GET_MATCH
);

foreach ($regx as $file) {
    echo $file[0] , "\n";
}
```

igbinary

- The awesome PHP Serializer you should use!
  - Faster
  - More Compact

```
;; Load igbinary extension
extension=igbinary.so

;; Use igbinary as session serializer
session.serialize_handler=igbinary
```

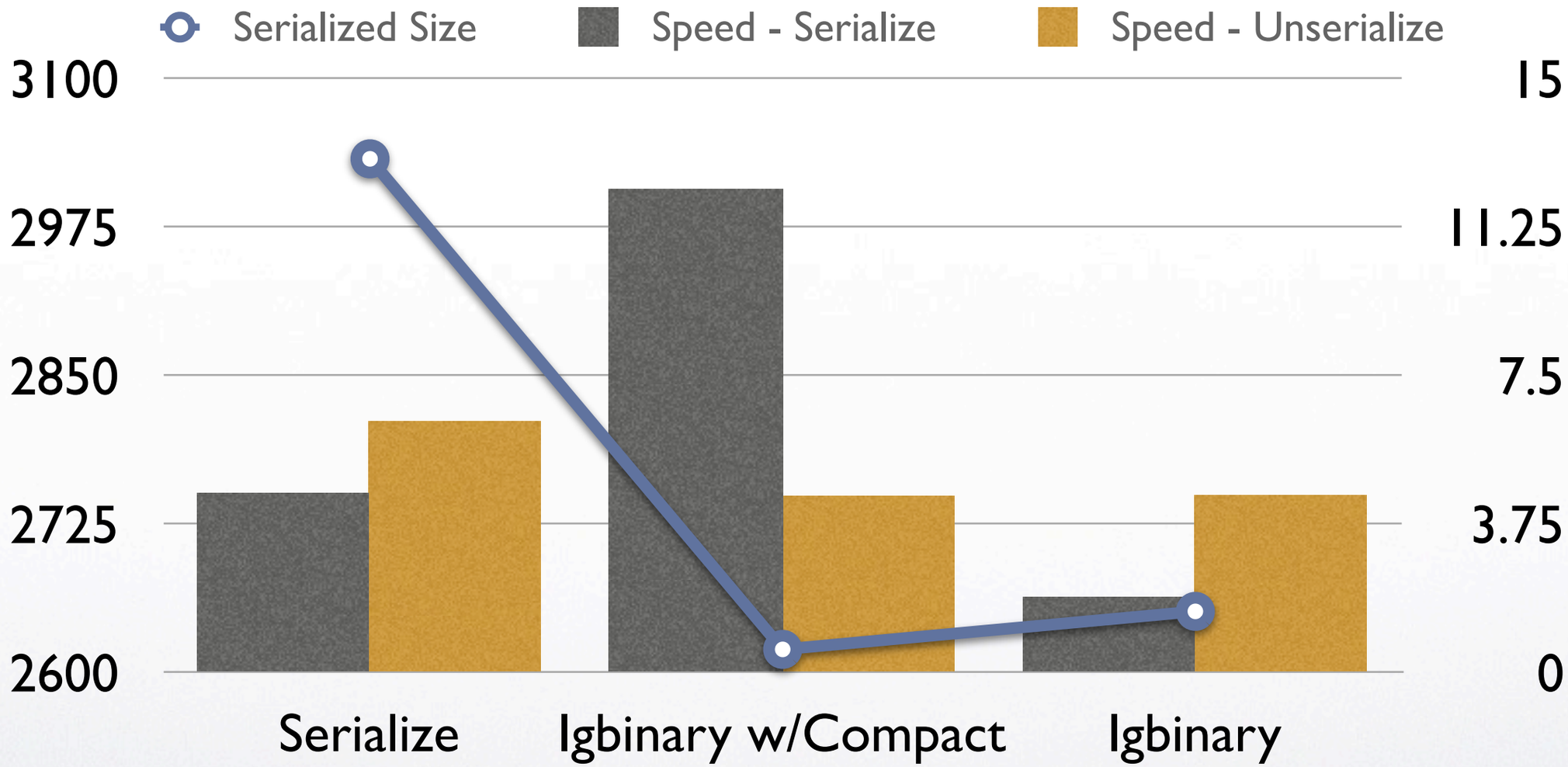
http://github.com/igbinary

igbinary

Provides functions you can use for non-session data.

```
ini_set("igbinary.compact_strings", 0);
igbinary_serialize($_SERVER);

ini_set("igbinary.compact_strings", 1);
igbinary_serialize($_SERVER);

// Un-serialize
igbinary_unserialize($x);
```

Igbinary speed test



FileInfo

- A reliable mechanism for identifying files
  - Not dependant on file extension
  - Can provide mime types
  - Identifies hundreds of file types

FileInfo How-To

```php
$finfo = finfo_open();
$file = __FILE__;

// mime description -- PHP script text
finfo_file($finfo, $file);

// mime type -- text/x-php
finfo_file($finfo, $file, FILEINFO_MIME_TYPE);

// mime -- text/x-php; charset=us-ascii
finfo_file($finfo, $file, FILEINFO_MIME);

// mime encoding -- us-ascii
finfo_file($finfo, $file, FILEINFO_MIME_ENCODING);
```

StatGrab

- An system informatics extension that provides information on:

Memory	Running Processed	System Load
CPU	Swap Utilization	System Info
Network	User Statistics	etc...

http://pecl.php.net/statgrab

StatGrab

sg_cpu_percent_usage()

```
Array
(
    [user] => 6.543484210968
    [kernel] => 3.2332751750946
    [idle] => 90.219924926758
    [iowait] => 0
    [swap] => 0
    [nice] => 0.0033116859849542
    [previous_run] => 1306895319
)
```

sg_cpu_totals() (in ticks)

```
Array
(
    [user] => 255652817
    [kernel] => 126323501
    [idle] => 3524877111
    [iowait] => 0
    [swap] => 0
    [nice] => 129387
    [total] => 3906982816
    [previous_run] => 1306895319
)
```

sg_cpu_diff() (in ticks)

```
Array
(
    [user] => 0 [kernel] => 1 [idle] => 1 [iowait] => 0
    [swap] => 0 [nice] => 0 [total] => 2 [previous_run] => 0
)
```

StatGrab

```
sg_diskio_stats()
Array (
    [sda] => Array (
            [read] => 447667436032
            [written] => 1098788713472
            [time_frame] => 1306895319
        )
)
```

```
sg_diskio_stats_diff()
Array (
    [sda] => Array (
            [read] => 0
            [written] => 0
            [time_frame] => 0
        )
)
```

StatGrab

```
sg_fs_stats()
Array (
    [0] => Array (
            [device_name] => /dev/sda2
            [fs_type] => ext3
            [mnt_point] => /
            [size] => 152892084224
            [used] => 126607642624
            [avail] => 18392698880
            [total_inodes] => 38535168
            [used_inodes] => 2324584
            [free_inodes] => 36210584
            [avail_inodes] => 36210584
            [io_size] => 4096
            [block_size] => 4096
            [total_blocks] => 37327169
            [free_blocks] => 6417100
            [used_blocks] => 30910069
            [avail_blocks] => 4490405
        )
)
```

StatGrab

```
sg_general_stats()
Array
(
    [os_name] => Linux
    [os_release] => 2.6.34-gentoo-r1
    [os_version] => #1 SMP Tue Aug 3 13:42:41 EDT 2010
    [platform] => x86_64
    [hostname] => dev.linux
    [uptime] => 9856588
)
```

StatGrab

sg_load_stats()

```
Array
(
    [min1] => 0.48
    [min5] => 0.46
    [min15] => 0.45
)
```

sg_memory_stats()

```
Array
(
    [total] => 5272272896
    [free] => 978161664
    [used] => 4294111232
    [cache] => 2546839552
)
```

sg_swap_stats()

```
Array
(
    [total] => 2097438720
    [free] => 556400640
    [used] => 1541038080
)
```

sg_process_count()

```
Array
(
    [total] => 324
    [running] => 1
    [sleeping] => 322
    [stopped] => 0
    [zombie] => 1
)
```

StatGrab

```
sg_network_stats()
Array
(
    [eth1] => Array
        (
            [sent] => 512193380346
            [received] => 88677061172
            [packets_received] => 481445357
            [packets_transmitted] => 527259007
            [receive_errors] => 0
            [transmit_errors] => 0
            [collisions] => 0
            [time_frame] => 1306895319
        )
)
```

StatGrab

```
sg_network_stats_diff()
Array
(
        [eth1] => Array
            (
                [sent] => 211456
                [received] => 33257
                [packets_received] => 228
                [packets_transmitted] => 231
                [receive_errors] => 0
                [transmit_errors] => 0
                [collisions] => 0
                [time_frame] => 5
            )
    )
```

StatGrab

sg_process_stats()

```
Array
(
    [87] => Array
        (
            [process_name] => postgres
            [proc_title] => postgres: autovacuum launcher process
            [pid] => 3650
            [parent_pid] => 3643
            [leader_pid] => 3650
            [uid] => 70
            [gid] => 70
            [euid] => 70
            [egid] => 70
            [size] => 65830912
            [size_in_mem] => 823296
            [time_spent] => 119
            [cpu_percent] => 0.0012106672247309
            [nice] => 0
            [state] => 1
        )
)
```

MailParse

- A very good mechanism for parsing complex e-mails
- Does a lot of the legwork for you
- Stable & Fast

http://pecl.php.net/mailparse

MailParse

```php
$msg = file_get_contents("php://stdin");

$mime = mailparse_msg_create();
if (!mailparse_msg_parse($mime, $msg)) {
  exit("could not parse email!\n");
}
```

```php
if (!($msg_info = mailparse_msg_get_part_data($mime))) {
  exit("could not get message info!\n");
} // $msg_info['headers'] is the headers associated array

$structure = mailparse_msg_get_structure($mime);
```

$msg_info['headers']

```
Array
(
   [return-path] => <apache@sender.com>
   [envelope-to] => ilia@ilia.ws
   [delivery-date] => Mon, 20 Jul 2009 13:15:32 -0400
   [received] => Array
      (
         [0] => from attendant.sender.net ([208.68.18.236] helo=hydrogen.sender.net) by ilia.ws with esmtp (Exim 4.69)
(envelope-from <apache@sender.com>) id 1MSwSa-0006IY-4O for ilia@ilia.ws; Mon, 20 Jul 2009 13:15:32 -0400
         [1] => from hydrogen.sender.net (hydrogen.sender.net [127.0.0.1]) by hydrogen.sender.net (8.13.8/8.13.8) with
ESMTP id n6KHFQ1g022841 for <ilia@ilia.ws>; Mon, 20 Jul 2009 11:15:26 -0600
         [2] => (from apache@localhost) by hydrogen.sender.net (8.13.8/8.13.8/Submit) id n6KHFQR4022840; Mon, 20
Jul 2009 11:15:26 -0600
      )

   [date] => Mon, 20 Jul 2009 11:15:26 -0600
   [message-id] => <200907201715.n6KHFQR4022840@hydrogen.sender.net>
   [to] => ilia@ilia.ws
   [subject] => 2027197
   [from] => from@sender.com
   [mime-version] => 1.0
   [content-type] => multipart/mixed; boundary="_____b7bfcc06f00337de46276f92b6833d5c++++++"
)
```

MailParse

```
// go through components of the e-mail
$body = ''; $atms = array();
foreach ($structure as $element) {
  $part_mime = mailparse_msg_get_part($mime, $element);
  $part = mailparse_msg_get_part_data($part_mime);

  if (!$part) continue;

$part_content = substr($msg,
                  $part['starting-pos-body'],
                  $part['ending-pos-body'] -
                     $part['starting-pos-body']);
```

MailParse

```
if ($part['transfer-encoding'] == 'base64') {
  $part_content = base64_decode($part_content);
} else if ($part['transfer-encoding']
                          == 'quoted-printable') {
  $part_content = quoted_printable_decode($part_content);
}
```

MailParse

```
if ($part['content-type'] == 'text/plain') {
  $body .= $part_content;
} elseif ($part['content-type'] == 'text/html') {
  $body .= strip_tags($part_content);
} elseif (!empty($part_info['content-disposition'])
    && $part['content-disposition'] == 'attachment'
) {
  $atms[] = array('headers' => $part, 'content' => $part_content);
} else { //inline attachments
  $atms[] = array('headers' => $part, 'content' => $part_content);
}
```

MailParse

```
$to = '"Ilia A." <ilia@ilia.ws>,
Test <test@test.com>,
foo@bar.com';

print_r(
 mailparse_rfc822_parse_addresses($to)
);
```

```
Array
(
    [0] => Array
        (
            [display] => Ilia A.
            [address] => ilia@ilia.ws
            [is_group] =>
        )
    [1] => Array
        (
            [display] => Test
            [address] => test@test.com
            [is_group] =>
        )
    [2] => Array
        (
            [display] => foo@bar.com
            [address] => foo@bar.com
            [is_group] =>
        )
)
```

PHP-Excel

- An interface to LibXL library
  - Allows generation of Excel Biff8 & XML documents
  - Can parse Excel Biff (5-8) and XML documents
  - Wickedly FAST! 200k rows in < 1 second

https://github.com/iliaal/php_excel

Creating Excel Docs

```php
$x = new ExcelBook();

$s = $x->addSheet("Sheet 1");
$s->write(1, 1, 'Test');
$s->write(2, 2, 123);

$x->save("file.xls");
```

	A	B	C
1			
2		Test	
3			123
4			
5			

Reading Excel Docs

```
$x = new ExcelBook();

$x->loadFile("file.xls");

$s = $x->getSheet();

for ($i = 0, $e = $s->lastRow(); $i < $e; $i++) {
    print_r(array_filter($s->readRow($i)));
}
```

Array ([1] => Test)

Array ([2] => 123)

xhprof

- Light weight PHP profiler designed for in production use.
  - Aggregate run data
  - Web interface
  - In-Production “sampling” mode

http://pecl.php.net/package/xhprof

http://github.com/preinheimer/xhprof

Profiling

;; Pre-pended to every PHP script (init)

```
auto_prepend_file = /xhprof/external/header.php
```

```
include_once __DIR__ . '/xhprof_lib/config.php');
include_once __DIR__ . '/xhprof_lib/utils/xhprof_lib.php';
include_once __DIR__ . '/xhprof_lib/utils/xhprof_runs.php';
xhprof_enable(XHPROF_FLAGS_CPU + XHPROF_FLAGS_MEMORY);
```

;;Appended to every PHP script (store)

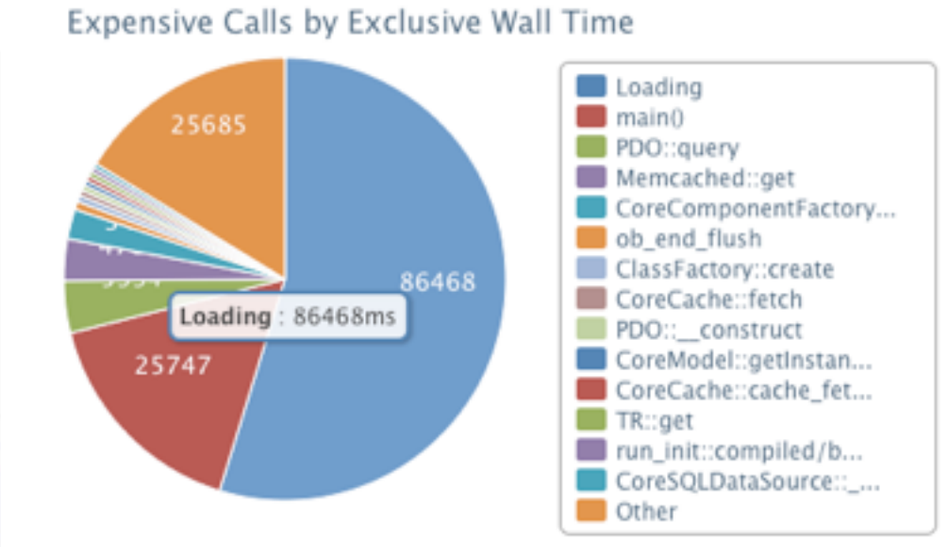
```
auto_append_file = /xhprof/external/footer.php
```

```
$xhprof_data = xhprof_disable();
$xhprof_runs = new XHProfRuns_Default();
$xhprof_runs->save_run($xhprof_data, 'AppName', null, $_xhprof);
```

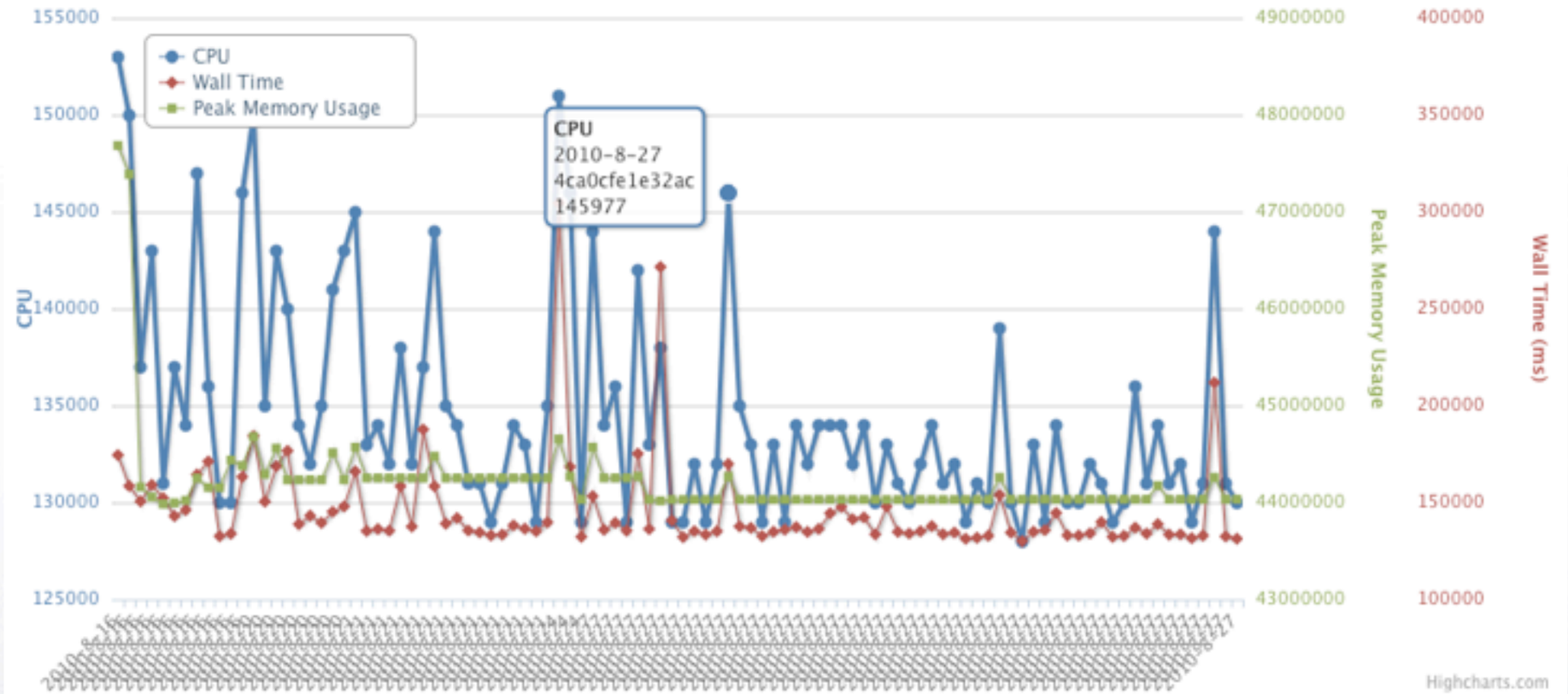
Profile Output

Stat	Exact URL	Similar URLs
Count	248	248
Min Wall Time	128.3070 ms	128.3070 ms
Max Wall Time	357.8990 ms	357.8990 ms
Avg Wall Time	148.6592 ms	148.6592 ms
95% Wall Time	211.8830 ms	211.8830 ms
Display run Incl. Wall Time (microsec)	**157,848 microsecs**	
Min CPU Ticks	124.9810 ms	124.9810 ms
Max CPU Ticks	225.9650 ms	225.9650 ms
Avg CPU Ticks	136.6728 ms	136.6728 ms
95% CPU Ticks	152.9780 ms	152.9780 ms
Display run Incl. CPU (microsecs)	**149,978 microsecs**	
Min Peak Memory Usage	43,985,208 bytes	43,985,208 bytes
Max Peak Memory Usage	47,683,560 bytes	47,683,560 bytes
Avg Peak Memory Usage	44,452,830 bytes	44,452,830 bytes
95% Peak Memory Usage	45,279,104 bytes	45,279,104 bytes
Display run Incl. PeakMemUse (bytes)	**44,909,544 bytes**	
Number of Function Calls:	**1,897**	

Profile Output



Profile Output



Profile Output

Call Count	Wall Time	CPU	Memory Usage	Peak Memory Usage	Exclusive Wall Time	Exclusive CPU	Exclusive Memory Usage	Exclusive Peak Memory Usage
1	174482	152978	47510112	47683560	20727	20996	4896	5832
1	73589	53993	5102248	5279096	97	0	4744	4064
1	72072	70990	38679368	38679144	72072	70990	38679368	38679144
1	69975	49993	4701800	4813208	179	0	4136	1720
2	43401	27996	4198312	4180048	50	0	-464	1008
2	43351	27996	4198776	4179040	319	0	10320	9304
2	39588	23997	3776408	3778320	125	0	4432	3376
2	39206	23997	3753400	3754384	78	0	-304	1112
5	38915	22998	3745776	3745624	403	0	10536	13808
5	38275	22998	3710200	3713040	2391	3000	29400	19160

Slides will be available
at **http://ilia.ws**

Ilia Alshanetsky
@iliaa