Memory leaky v PHP a jak je najít

IT workshop v Plzni

19. 10. 2022 v 18:00

Restaurace U Salzmannů

PeoplePath

php

salonek v 1. patře

vstup zdarma

občerstvení zajištěno

Today’s agenda

- What is memory leak/space leak?
- How PHP handle memory?
- How to debug memory usage in PHP?
- How to solve common problems?

What is memory leak in layman's term

Program memory unintentionally contains data that are no longer need it and developer/program don’t properly react to this.

Talking about memory leaks in context of PHP can be confusing as…

…PHP manage memory for us:

- Automatic memory management.

- Share-nothing architecture.

- Memory safe.

Should we care about memory usage?

Yes we should, because…

- Inefficient memory management.
- Long running processes.
- Wrong usage of FFI.
- Bugs in PHP or extensions.

Yes you should, because…

- **Inefficient memory management.**
- Long running processes.
- Not so common:
    - Wrong usage of FFI.
    - Bugs in PHP or extensions.

Did you ever encounter following fatal error?

```
Fatal error: Allowed memory size of x bytes
exhausted (tried to allocate x bytes)
```

Memory leak vs space leak

- Not every issue with memory implies memory leak.
- Some programs an ineffective thanks to wrong design.
- Seemingly good looking implementation can have an issue.

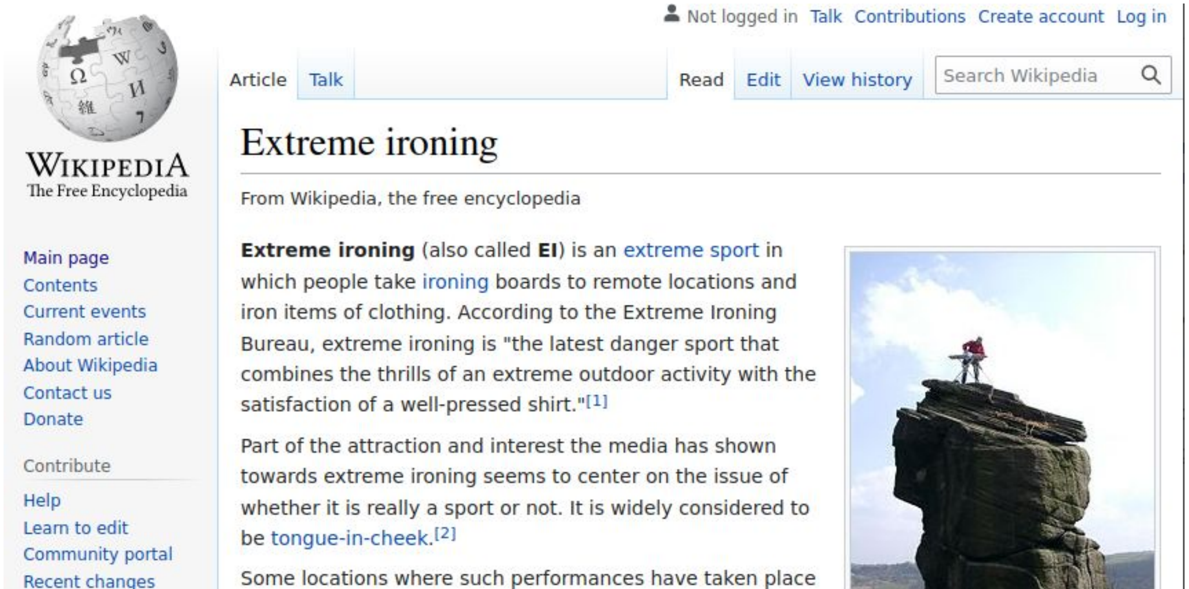
Some of the programs can cause “space leak“.

Space leak

- A space leak occurs when a computer program uses more memory than necessary.
- In contrast to memory leaks:
    - The memory consumed by program is released, but later than expected.

Space leak

- For example: you buy a 26-volume printed encyclopedia with intention to just read article about extreme ironing.



Extreme ironing

From Wikipedia, the free encyclopedia

Extreme ironing (also called **EI**) is an extreme sport in which people take ironing boards to remote locations and iron items of clothing. According to the Extreme Ironing Bureau, extreme ironing is "the latest danger sport that combines the thrills of an extreme outdoor activity with the satisfaction of a well-pressed shirt."[1]

Part of the attraction and interest the media has shown towards extreme ironing seems to center on the issue of whether it is really a sport or not. It is widely considered to be tongue-in-cheek.[2]

Some locations where such performances have taken place

Space leak

Border between memory leaks and space leaks is blurry.

X

Space leaks are usually more prevalent.

How PHP handle memory

How does PHP handle memory?

- Automatic memory management:
    - reference counting and garbage collection.
- Individual memory optimization.
- Memory limitation built into language.
- Specific memory patterns for PHP process managers.

Reference counting

- Keep a counter of references to every object/array/string.
    - reference does not means “PHP reference” **&**
- Basic rules:
    - Add 1 when copying the reference (eg. pass to fce).
    - Subtract 1 when clearing a reference (eg. unset).
    - If counter drop to 0, remove the object from memory.

Reference counting - few facts

- Is predictable.
- Frees memory as soon as possible.
- No pause for cycle collection.

X

- Fails to address circular references.

Code

```php
function bar() {
  $array  = [];
  $parent = new StdClass();
  $child  = new StdClass();
  $parent->child = $child;
  $child->parent = $parent;
}

bar();
echo 'end';
```

Memory

main scope:

Code

```php
function bar() {
  $array  = [];
  $parent = new StdClass();
  $child  = new StdClass();
  $parent->child = $child;
  $child->parent = $parent;
}

bar();
echo 'end';
```

Memory

function bar scope:

$array, ref count: 1

Code

```php
function bar() {
  $array  = [];
  $parent = new StdClass();
  $child  = new StdClass();
  $parent->child = $child;
  $child->parent = $parent;
}

bar();
echo 'end';
```

Memory

function bar scope:

$array, ref count: 1

$parent, ref count: 1

Code

```php
function bar() {
  $array  = [];
  $parent = new StdClass();
  $child  = new StdClass();
  $parent->child = $child;
  $child->parent = $parent;
}

bar();
echo 'end';
```

Memory

function bar scope:

$array, ref count: 1

$parent, ref count: 1

$child, ref count: 1

Code

```php
function bar() {
  $array  = [];
  $parent = new StdClass();
  $child  = new StdClass();
  $parent->child = $child;
  $child->parent = $parent;
}

bar();
echo 'end';
```

Memory

function bar scope:

$array, ref count: 1

$parent, ref count: 1

$child, ref count: 2 (+1)

Code

```php
function bar() {
  $array  = [];
  $parent = new StdClass();
  $child  = new StdClass();
  $parent->child = $child;
  $child->parent = $parent;
}

bar();
echo 'end';
```

Memory

function bar scope:

$array, ref count: 1

$parent, ref count: 2 (+1)

$child, ref count: 2 (+1)

Code

```php
function bar() {
  $array  = [];
  $parent = new StdClass();
  $child  = new StdClass();
  $parent->child = $child;
  $child->parent = $parent;
}

bar();
echo 'end';
```

Memory

function bar scope:

$array, ref count: 0 (-1) -> removed

$parent, ref count: 2

$child, ref count: 2

Ref count must must be lower by one for any variable that won't leave function and its **not referenced by other variable**.

Code

```php
function bar() {
  $array  = [];
  $parent = new StdClass();
  $child  = new StdClass();
  $parent->child = $child;
  $child->parent = $parent;
}

bar();
echo 'end';
```

Memory

function bar scope:

$parent, ref count: 2

$child, ref count: 2

Parent and child remains in memory.

Code

```php
function bar() {
  $array  = [];
  $parent = new StdClass();
  $child  = new StdClass();
  $parent->child = $child;
  $child->parent = $parent;
}

bar();
echo 'end';
```

Memory

main scope:

function bar scope:

$parent, ref count: 2

$child, ref count: 2

Parent and child remains in memory.

Garbage collector to rescue!

- Reference counting fails to address circular references.
- Only garbage collector can solve this.
- Since 5.3.0, PHP has a garbage collector.
- Before 5.3.0 - regular memory leaks.

Garbage collector to rescue!

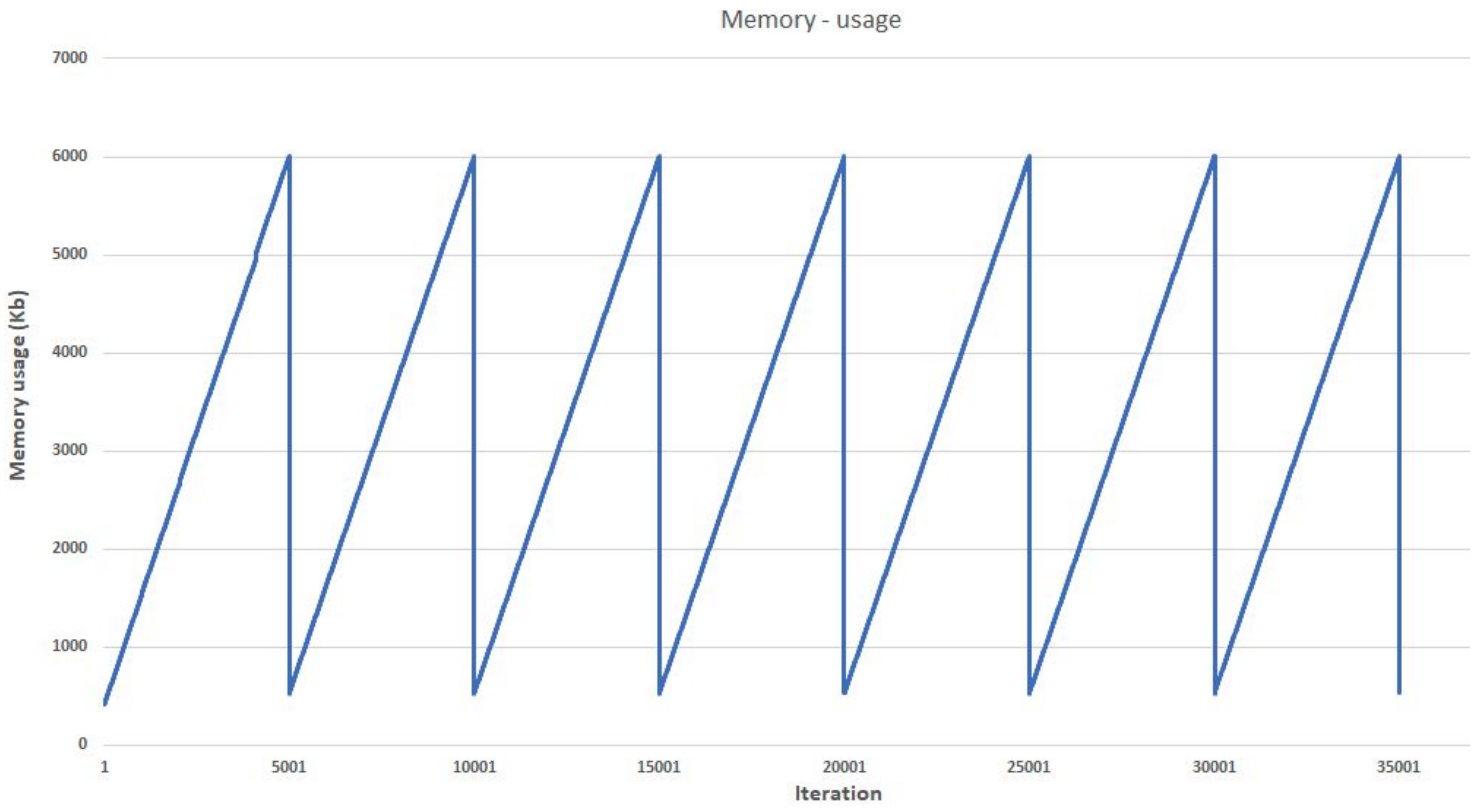
- Garbage collector handle circular references.

X

- Collection cycle is not done after every function run.
- Fixed threshold - 10 000 objects with cycle references.
- Sawtooth effect on memory usage.

Sawtooth effect - wait is this a memory leak?

```php
function bar() {
    $parent = new StdClass();
    $child  = new StdClass();
    $parent->child = $child;
    $child->parent = $parent;
}

for ($i = 1; $i <= 40000; $i++) {
   bar();
   echo $i.';'.(memory_get_usage()/1024)."\n";
}
```


Memory - usage
Memory usage (Kb)
7000
6000
5000
4000
3000
2000
1000
0
1
5001
10001
15001
20001
25001
30001
35001
Iteration

Memory optimizations - copy on write

A

```
function array_test(array $test) {
    echo $test[0]."\n";
}

$array = range(1, 10000000);

array_test($array);
```

```
Memory usage: 512.40MB
```

B

```
function array_test(array $test) {
    $test[] = 10000001;
    echo $test[0]."\n";
}

$array = range(1, 10000000);

array_test($array);
```

```
Memory usage: 1024.41MB
```

Whats happens - copy on write

```php
function array_test(array $test) {
    $test[] = 10000001;   // duplicate the array
    echo $test[0]."\n";
}

$array = range(1, 10000000);

array_test($array);
```

Whats happens?

Pass array by value into function

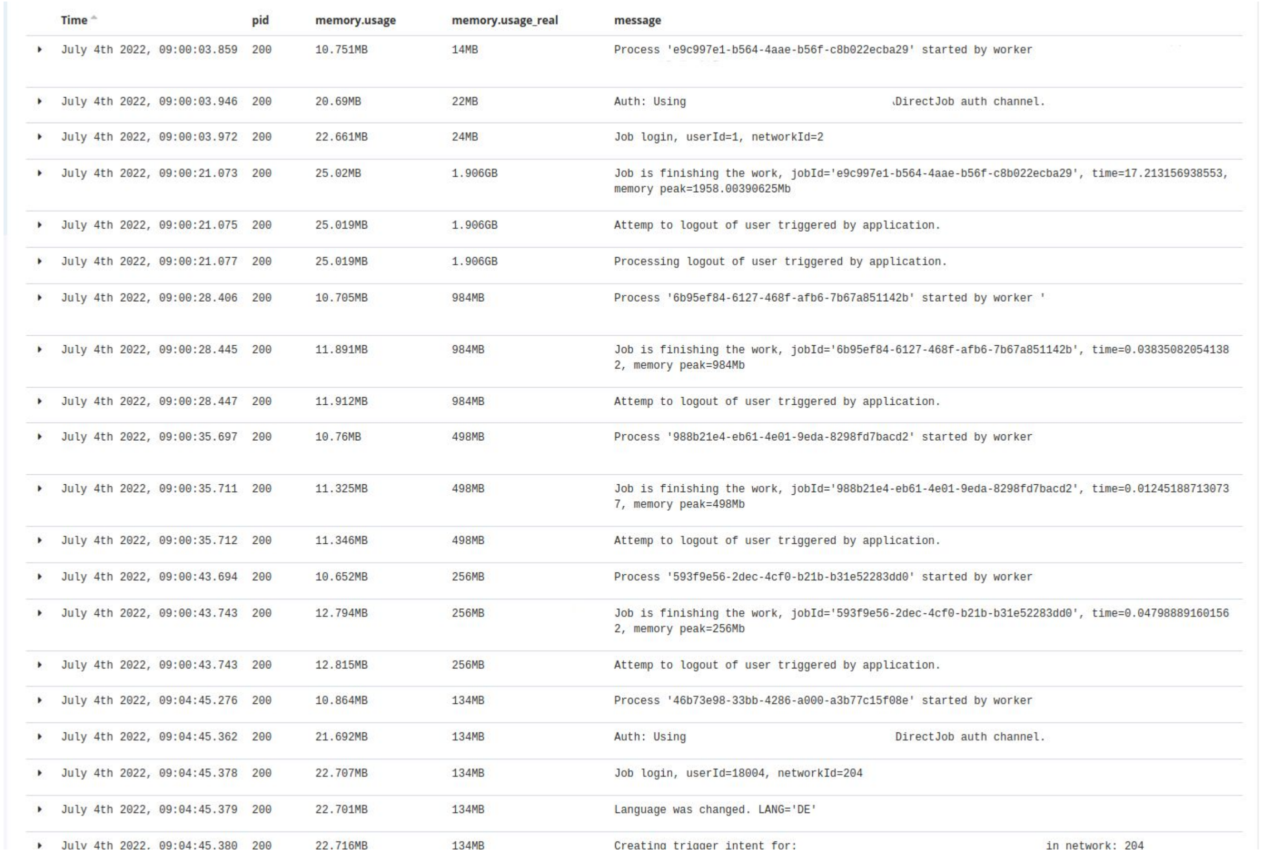
$\neq$

Immediately duplication of variable

Copy on write

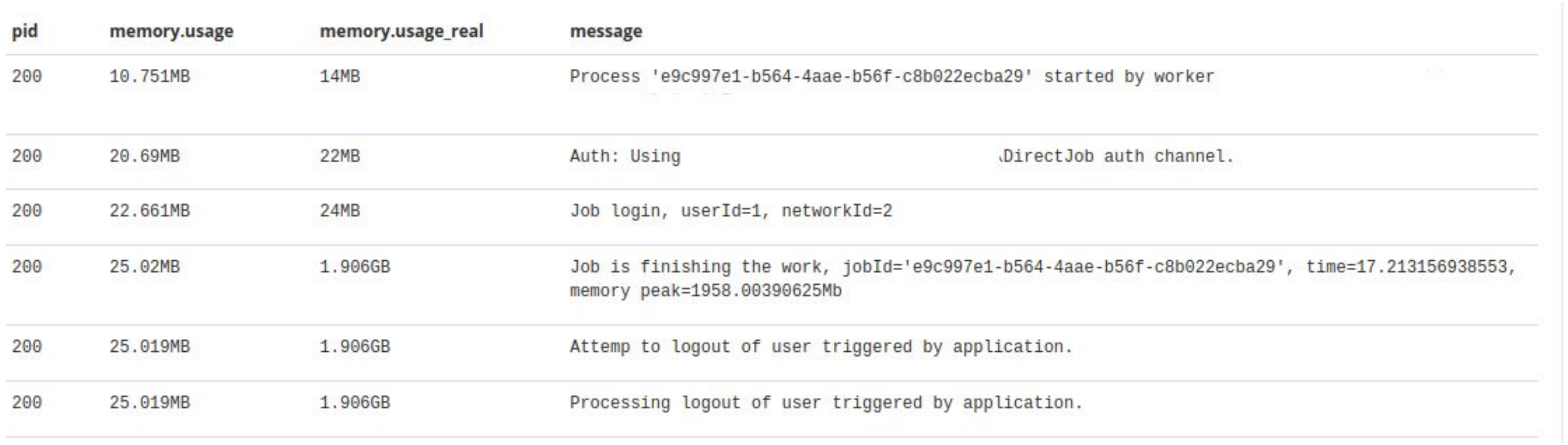
- What about writing into passed array?
- You can use references, but it's better to use objects.
- It has also other advantages:
  - More expressiveness.
  - Encapsulation of business.

FPM memory hysteresis



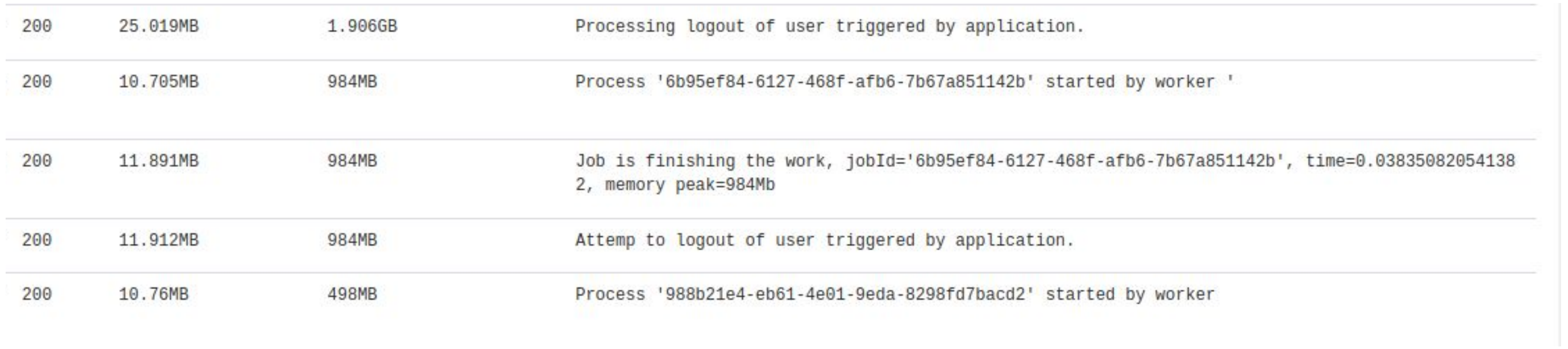
Time	pid	memory.usage	memory.usage_real	message
July 4th 2022, 09:00:03.859	200	10.751MB	14MB	Process 'e9c997e1-b564-4aae-b56f-c8b022ecba29' started by worker
July 4th 2022, 09:00:03.946	200	20.69MB	22MB	Auth: Using DirectJob auth channel.
July 4th 2022, 09:00:03.972	200	22.661MB	24MB	Job login, userId=1, networkId=2
July 4th 2022, 09:00:21.073	200	25.02MB	1.906GB	Job is finishing the work, jobId='e9c997e1-b564-4aae-b56f-c8b022ecba29', time=17.213156938553, memory peak=1958.00390625Mb
July 4th 2022, 09:00:21.075	200	25.019MB	1.906GB	Attemp to logout of user triggered by application.
July 4th 2022, 09:00:21.077	200	25.019MB	1.906GB	Processing logout of user triggered by application.
July 4th 2022, 09:00:28.406	200	10.705MB	984MB	Process '6b95ef84-6127-468f-afb6-7b67a851142b' started by worker '
July 4th 2022, 09:00:28.445	200	11.891MB	984MB	Job is finishing the work, jobId='6b95ef84-6127-468f-afb6-7b67a851142b', time=0.038350820541382, memory peak=984Mb
July 4th 2022, 09:00:28.447	200	11.912MB	984MB	Attemp to logout of user triggered by application.
July 4th 2022, 09:00:35.697	200	10.76MB	498MB	Process '988b21e4-eb61-4e01-9eda-8298fd7bacd2' started by worker
July 4th 2022, 09:00:35.711	200	11.325MB	498MB	Job is finishing the work, jobId='988b21e4-eb61-4e01-9eda-8298fd7bacd2', time=0.012451887130737, memory peak=498Mb
July 4th 2022, 09:00:35.712	200	11.346MB	498MB	Attemp to logout of user triggered by application.
July 4th 2022, 09:00:43.694	200	10.652MB	256MB	Process '593f9e56-2dec-4cf0-b21b-b31e52283dd0' started by worker
July 4th 2022, 09:00:43.743	200	12.794MB	256MB	Job is finishing the work, jobId='593f9e56-2dec-4cf0-b21b-b31e52283dd0', time=0.047988891601562, memory peak=256Mb
July 4th 2022, 09:00:43.743	200	12.815MB	256MB	Attemp to logout of user triggered by application.
July 4th 2022, 09:04:45.276	200	10.864MB	134MB	Process '46b73e98-33bb-4286-a000-a3b77c15f08e' started by worker
July 4th 2022, 09:04:45.362	200	21.692MB	134MB	Auth: Using DirectJob auth channel.
July 4th 2022, 09:04:45.378	200	22.707MB	134MB	Job login, userId=18004, networkId=204
July 4th 2022, 09:04:45.379	200	22.701MB	134MB	Language was changed. LANG='DE'
July 4th 2022, 09:04:45.380	200	22.716MB	134MB	Creating trigger intent for: in network: 204

FPM memory hysteresis



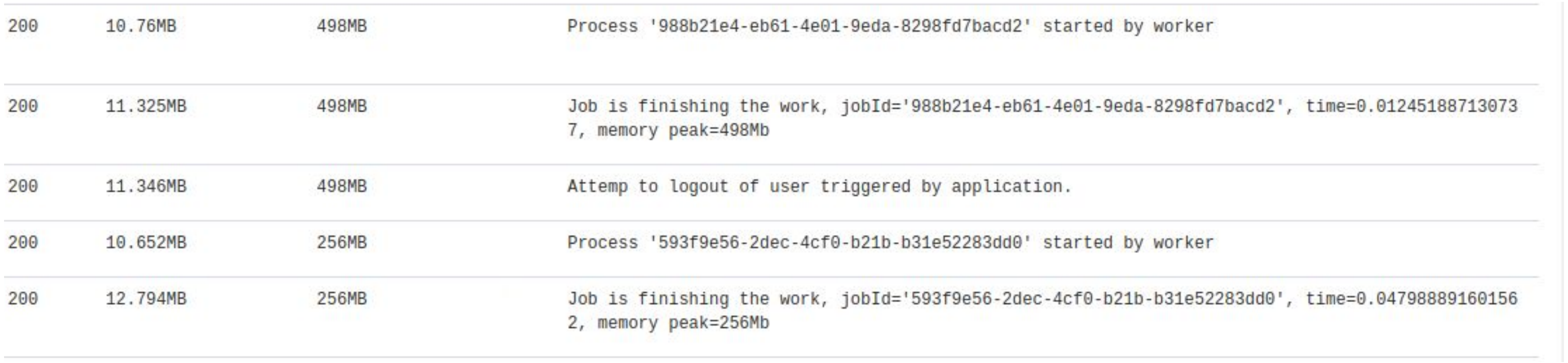
pid	memory.usage	memory.usage_real	message
200	10.751MB	14MB	Process 'e9c997e1-b564-4aae-b56f-c8b022ecba29' started by worker
200	20.69MB	22MB	Auth: Using DirectJob auth channel.
200	22.661MB	24MB	Job login, userId=1, networkId=2
200	25.02MB	1.906GB	Job is finishing the work, jobId='e9c997e1-b564-4aae-b56f-c8b022ecba29', time=17.213156938553, memory peak=1958.00390625Mb
200	25.019MB	1.906GB	Attemp to logout of user triggered by application.
200	25.019MB	1.906GB	Processing logout of user triggered by application.

FPM memory hysteresis



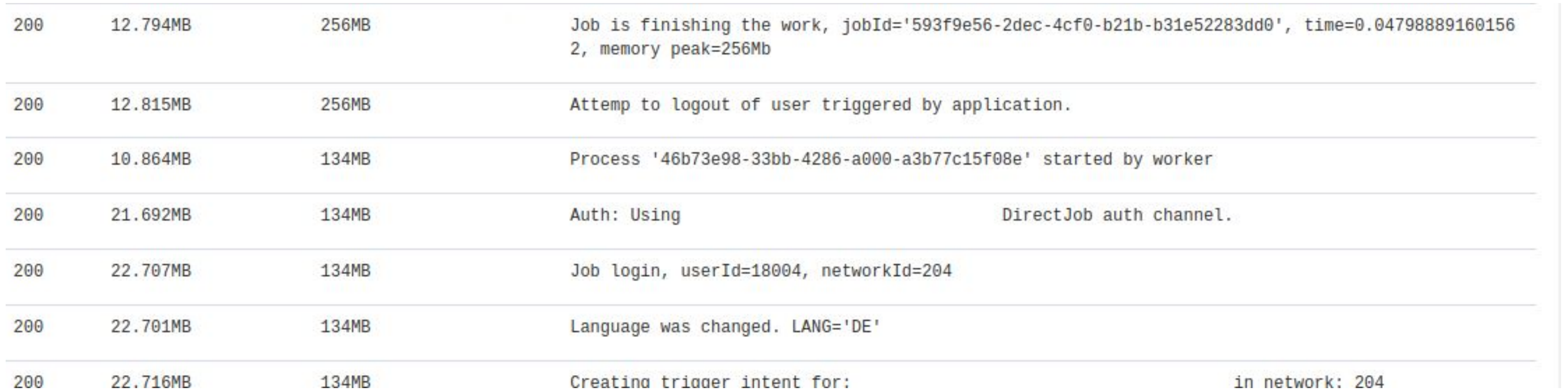
```
200    25.019MB    1.906GB    Processing logout of user triggered by application.
200    10.705MB    984MB      Process '6b95ef84-6127-468f-afb6-7b67a851142b' started by worker '
200    11.891MB    984MB      Job is finishing the work, jobId='6b95ef84-6127-468f-afb6-7b67a851142b', time=0.03835082054138
                              2, memory peak=984Mb
200    11.912MB    984MB      Attemp to logout of user triggered by application.
200    10.76MB     498MB      Process '988b21e4-eb61-4e01-9eda-8298fd7bacd2' started by worker
```

FPM memory hysteresis



```
200   10.76MB    498MB   Process '988b21e4-eb61-4e01-9eda-8298fd7bacd2' started by worker

200   11.325MB   498MB   Job is finishing the work, jobId='988b21e4-eb61-4e01-9eda-8298fd7bacd2', time=0.01245188713073
                         7, memory peak=498Mb

200   11.346MB   498MB   Attemp to logout of user triggered by application.

200   10.652MB   256MB   Process '593f9e56-2dec-4cf0-b21b-b31e52283dd0' started by worker

200   12.794MB   256MB   Job is finishing the work, jobId='593f9e56-2dec-4cf0-b21b-b31e52283dd0', time=0.04798889160156
                         2, memory peak=256Mb
```

FPM memory hysteresis



```
200    12.794MB    256MB    Job is finishing the work, jobId='593f9e56-2dec-4cf0-b21b-b31e52283dd0', time=0.04798889160156
                            2, memory peak=256Mb

200    12.815MB    256MB    Attemp to logout of user triggered by application.

200    10.864MB    134MB    Process '46b73e98-33bb-4286-a000-a3b77c15f08e' started by worker

200    21.692MB    134MB    Auth: Using                          DirectJob auth channel.

200    22.707MB    134MB    Job login, userId=18004, networkId=204

200    22.701MB    134MB    Language was changed. LANG='DE'

200    22.716MB    134MB    Creating trigger intent for:                                  in network: 204
```

FPM memory hysteresis

- PHP process spawned by FPM won’t release allocated memory during request.
- Even after the request end it won’t be fully released.
- Just half of it will be freed.

How to debug memory usage?

Tools

- Builtin functions,
- debuggers (Xdebug) and
- various single purpose extensions (php-meminfo).

Builtin functions

- Always available.
- A "`var_dump`" debug strategy - trial and error.
- Three functions:
  - **`memory_get_usage`**
  - **`memory_get_peak_usage`**
  - **`memory_reset_peak_usage`** (from PHP 8.2)

```php
function get_data(string $file_path) : array {
    $file = fopen($file_path, 'r');
    $lines = [];
    while (($line = fgets($file)) !== false) {
        $lines[] = trim($line);
    }

    return $lines;
}

$lines = get_data('heap.txt');
foreach ($lines as $line_number => $line) {
    if ($line === 'needle') {
        echo 'found at line: '.$line_number."\n";
        break;
    }
}
```

Example

```
Fatal error: Allowed memory size of 31457280 bytes
exhausted (tried to allocate 8388616 bytes) in
/usr/src/myapp/readDataFromFile.php on line 5
```

```php
function get_data(string $file_path) : array {
    $file = fopen($file_path, 'r');
    $lines = [];
    while (($line = fgets($file)) !== false) {
        $lines[] = trim($line); // Allowed memory size of x bytes exhausted
    }

    return $lines;
}

$lines = get_data('heap.txt');
foreach ($lines as $line_number => $line) {
    if ($line === 'needle') {
        echo 'found at line: '.$line_number."\n";
        break;
    }
}
```

```php
function get_data(string $file_path) : array {
    echo sprintf("Fce enter: %dKB\n", memory_get_usage()/1024);
    $file = fopen($file_path, 'r');
    $lines = [];
    while (($line = fgets($file)) !== false) {
        echo sprintf("Read line: %dKB\n", memory_get_usage()/1024);
        $lines[] = trim($line);
    }

    return $lines;
}

$lines = get_data('heap.txt');
foreach ($lines as $line_number => $line) {
    if ($line === 'needle') {
        echo 'found at line: '.$line_number."\n";
        break;
    }
}
```

Example

```
php readDataFromFileDebug.php
```

```
Fce enter: 385KB
Read line: 394KB
... // abbreviated for sake of clarity
Read line: 937KB
... // abbreviated for sake of clarity
Read line: 1847KB

Fatal error: Allowed memory size of 2097152 bytes exhausted (tried to
allocate 4096 bytes) in readDataFromFile.php on line 8
```

Example - temporary disable memory limit

```
php -d memory_limit=-1 readDataFromFileDebug.php
```

```
Fce enter: 385KB
Read line: 394KB
Read line: 394KB
... // abbreviated for sake of clarity
Read line: 260957KB
Read line: 260957KB
Read line: 260958KB
Read line: 260958KB
found at line: 497096
```

```php
function get_data(string $file_path) : array {
    $file = fopen($file_path, 'r');
    $lines = [];
    while (($line = fgets($file)) !== false) {
        $lines[] = trim($line);
    }

    return $lines;
}

$lines = get_data('heap.txt');
foreach ($lines as $line_number => $line) {
    if ($line === 'needle') {
        echo 'found at line: '.$line_number."\n";
        break;
    }
}
```

(one of possible)

Solution

Solution, eg.: by using generators

```php
function get_data(string $file_path) : iterable {
    $file = fopen($file_path, 'r');
    while (($line = fgets($file)) !== false) {
        yield trim($line);
    }
}

$lines = get_data('heap.txt');
foreach ($lines as $line_number => $line) {
    if ($line === 'needle') {
        echo 'found at line: '.$line_number."\n";
        break;
    }
}
```

Solution, eg.: by using generators

```php
function get_data(string $file_path) : iterable {
    $file = fopen($file_path, 'r'); // where is fclose?
    while (($line = fgets($file)) !== false) {
        yield trim($line);
    }
}

$lines = get_data('heap.txt');
foreach ($lines as $line_number => $line) {
    if ($line === 'needle') {
        echo 'found at line: '.$line_number."\n";
        break;
    }
}
```

Solution, eg.: by using generators

```php
function get_data(string $file_path) : iterable {
   $file = fopen($file_path, 'r');
   while (($line = fgets($file)) !== false) {
       yield trim($line);
   }
   fclose($file);
}

$lines = get_data('heap.txt');
foreach ($lines as $line_number => $line) {
   if ($line === 'needle') {
       echo 'found at line: '.$line_number."\n";
       break;
   }
}
```

Solution, eg.: by using generators

```php
function get_data(string $file_path) : iterable {
   $file = fopen($file_path, 'r');
   while (($line = fgets($file)) !== false) {
       yield trim($line);
   }
   fclose($file);
}

$lines = get_data('heap.txt');
foreach ($lines as $line_number => $line) {
   if ($line === 'needle') {
       echo 'found at line: '.$line_number."\n";
   }
}
```

Solution, eg.: by using generators

```php
function get_data(string $file_path) : iterable {
   $file = fopen($file_path, 'r');
   while (($line = fgets($file)) !== false) {
       yield trim($line);
   }
   fclose($file);
}

$lines = get_data('heap.txt');
foreach ($lines as $line_number => $line) {
   if ($line === 'needle') {
       echo 'found at line: '.$line_number."\n";
       break; // bug fix - it’s faster, yeah!
   }
}
```

Solution, eg.: by using generators

```php
function get_data(string $file_path) : iterable {
    $file = fopen($file_path, 'r');
    while (($line = fgets($file)) !== false) {
        yield trim($line);
    }
    fclose($file);
}

$lines = get_data('heap.txt');
foreach ($lines as $line_number => $line) {
    if ($line === 'needle') {
        echo 'found at line: '.$line_number."\n";
    }
}
```

Solution, don’t be lazy!

```php
function search(string $file_path, string $needle) : ?int {
   $file = fopen($file_path, 'r');
   $lineNumber = 1;
   while (($line = fgets($file)) !== false) {
       if (trim($line) === $needle) {
           fclose($file);
           return $lineNumber;
       }
       $lineNumber++;
   }
   fclose($file);
   return null;
}

if (($line_number = search('heap.txt', 'needle')) !== null) {
   echo 'found at line: '.$line_number."\n";
}
```

Debuggers - xdebug

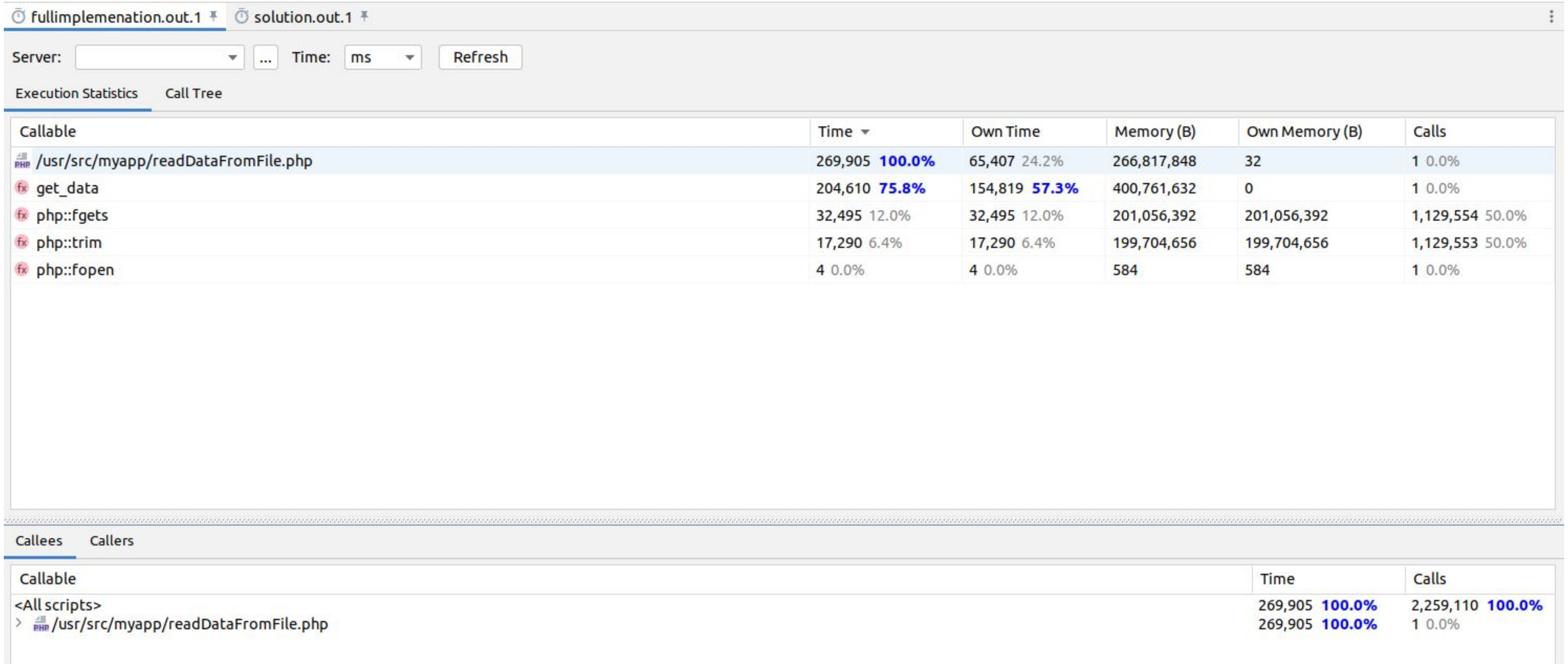
- Profiling function contains also information about memory.
- If you have xdebug already installed, you can turn it by environmental variables:

```
XDEBUG_MODE=profile
XDEBUG_CONFIG=output_dir=/usr/src/myapp/tmp/
```

Debuggers - xdebug

- Execute problematic code.
- Generate profile snapshot and open it in PHPStorm.
- Works on function level.
- Memory usage is aggregate per method.

Xdebug



Callable	Time	Own Time	Memory (B)	Own Memory (B)	Calls
/usr/src/myapp/readDataFromFile.php	269,905 100.0%	65,407 24.2%	266,817,848	32	1 0.0%
get_data	204,610 75.8%	154,819 57.3%	400,761,632	0	1 0.0%
php::fgets	32,495 12.0%	32,495 12.0%	201,056,392	201,056,392	1,129,554 50.0%
php::trim	17,290 6.4%	17,290 6.4%	199,704,656	199,704,656	1,129,553 50.0%
php::fopen	4 0.0%	4 0.0%	584	584	1 0.0%

Callable	Time	Calls
<All scripts>	269,905 100.0%	2,259,110 100.0%
/usr/src/myapp/readDataFromFile.php	269,905 100.0%	1 0.0%

php-meminfo

- https://github.com/BitOne/php-meminfo
- Native PHP extension, PHP 7-8
- Two components:
  - extension itself,
  - analyzers written in PHP.

php-meminfo - how to use it

- Extension provide just one method (`meminfo_dump`).
- Call the method at end of request.

```
meminfo_dump(fopen('memory.json', 'w'));
```

- Execute the code.
- Analyze report in provided PHP tool (analyzer)

```php
function get_data(string $file_path) : array {
   $file = fopen($file_path, 'r');
   $lines = [];
   while (($line = fgets($file)) !== false) {
       $lines[] = trim($line);
   }

   return $lines;
}

$lines = get_data('heap.txt');
foreach ($lines as $line_number => $line) {
   if ($line === 'needle') {
       echo 'found at line: '.$line_number."\n";
       break;
   }
}

meminfo_dump(fopen('memory.json', 'w'));
```

php-meminfo

```
./bin/analyzer summary memory.json

+--------+-----------------+-----------------------------+
| Type   | Instances Count | Cumulated Self Size (bytes) |
+--------+-----------------+-----------------------------+
| string | 1129594         | 171884022                   |
| array  | 10              | 720                         |
| int    | 4               | 64                          |
| float  | 1               | 16                          |
+--------+-----------------+-----------------------------+
```

php-meminfo

```
./bin/analyzer top-children memory.json

+-----+----------------+-----------+
| Num | Item ids       | Children  |
+-----+----------------+-----------+
| 1   | 0x7f2138a14070 | 1129553   |
| 2   | 0x7f2138a59300 | 26        |
| 3   | 0x7f2138a592c0 | 17        |
| 4   | 0x7f2138a59280 | 1         |
| 5   | 0x7f2138a76900 | 1         |
+-----+----------------+-----------+
```

php-meminfo

```
 ./bin/analyzer ref-path  0x7f2138a14070 memory.json

Found 1 paths
Path to 0x7f2138a14070
(<GLOBAL>)$lines["<self>"]
```

More complex example

```php
use \Kambo\MemoryLeaks\{UserId,UserService};

$userService = new UserService();

for ($i = 1; $i <= 1000000; $i++) {
    $result = $userService->getUser(new UserId($i));
    echo $result."\n";
}
```

```
Annabelle Emard
Grady Abshire
Aliza Little

Fatal error: Allowed memory size of 3145728 bytes exhausted (tried to allocate
4096 bytes) in UserService.php on line 31
```

```php
use \Kambo\MemoryLeaks\{UserId,UserService};

$userService = new UserService();

for ($i = 1; $i <= 1000000; $i++) {
   $result = $userService->getUser(new UserId($i));
   echo $result."\n";
}

meminfo_dump(fopen('complex-dump.json', 'w'));
```

php-meminfo

```
./bin/analyzer summary complex-dump.json

+------------------------------------+------------------+------------------------------+
| Type                               | Instances Count |Cumulated Self Size (bytes)   |
+------------------------------------+------------------+------------------------------+
| string                             | 3000044          | 74046833                     |
| Kambo\MemoryLeaks\User             | 1000000          | 72000000                     |
| array                              | 25               | 1800                         |
| int                                | 6                | 96                           |
| bool                               | 2                | 32                           |
| Kambo\MemoryLeaks\UserService      | 1                | 72                           |
| SQLite3                            | 1                | 72                           |
| Kambo\MemoryLeaks\ArrayBasedCache | 1                | 72                           |
| Composer\Autoload\ClassLoader      | 1                | 72                           |
| float                              | 1                | 16                           |
| null                               | 1                | 16                           |
+------------------------------------+------------------+------------------------------+
```

```php
use \Kambo\MemoryLeaks\{UserId,UserService};

$userService = new UserService();

for ($i = 1; $i <= 1000000; $i++) {
   $result = $userService->getUser(new UserId($i));
   echo $result."\n";
}

```

```php
namespace Kambo\MemoryLeaks;

class User
{
   private string $userName;
   private string $name;
   private string $surname;

   public function __construct($userName, $name, $surname) {
        $this->userName = $userName;
        $this->name     = $name;
        $this->surname  = $surname;
   }

   public static function fromArray(array $data) : self {
        return new self($data['username'], $data['name'], $data['surname']);
   }

   public function __toString() : string{
        return $this->name. ' ' . $this->surname;
   }
}
```

php-meminfo

```
./bin/analyzer top-children complex-dump.json

+-----+----------------+-----------+
| Num | Item ids       | Children  |
+-----+----------------+-----------+
| 1   | 0x7ff209458e28 | 1000000   |
| 2   | 0x7ff209459300 | 26        |
| 3   | 0x7ff2094592c0 | 17        |
| 4   | 0x7ff20947c380 | 11        |
| 5   | 0x7ff209456780 | 3         |
+-----+----------------+-----------+
```

php-meminfo

```
./bin/analyzer ref-path 0x7ff209458e28 complex-dump.json

Found 1 paths
Path to 0x7ff209475cd0
(<GLOBAL>)$userService
->cache
  ->cache
```

```
./bin/analyzer -v ref-path 0x7ff209458e28 complex-dump.json
```

```
Found 1 paths                                        +--------------------------------+
Path from 0x7ff209475cd0                             |                                |
+-------------------------+                          |                                |
| Id: 0x7ff209458e28      |                          |                         +-------------------------------------------+
| Type: array             |                          |                         | Id: 0x7ff209458e00                        |
| Size: 72 B              |                          |                         | Type: object                              |
| Is root: No             |                          |                         | Class: Kambo\MemoryLeaks\ArrayBasedCache |
| Children count: 1000000 |                          |                         | Object Handle: 4                          |
+-------------------------+                          |                         | Size: 72 B                                |
        ^                                            |                         | Is root: No                               |
        |                                            |                         | Children count: 1                         |
       cache                                         |                         +-------------------------------------------+
        |                                            |                                ^
        |                                            |                                |
        |                                            |                                cache
        |                                            |                                |
        |                                            |                                |
        |                                            |                         +--------------------------------------+
        |                                            |                         | Id: 0x7ff209475cd0                   |
        |                                            |                         | Type: object                         |
        |                                            |                         | Class: Kambo\MemoryLeaks\UserService |
        |                                            |                         | Object Handle: 3                     |
        |                                            |                         | Size: 72 B                           |
        |                                            |                         | Is root: Yes                         |
        |                                            |                         | Execution Frame: <GLOBAL>            |
        |                                            |                         | Symbol Name: userService             |
        |                                            |                         | Children count: 2                    |
        |                                            |                         +--------------------------------------+
        +--------------------------------------------+
```

```php
class UserService
{
    private Cache $cache;

    public function __construct() {
        $this->cache = new ArrayBasedCache();
    }

    public function getUser(UserId $userId) : ?User {
        if ($this->cache->has($userId)) {
            return $this->cache->get($userId);
        }

        // Get $userData from database

        $user = User::fromArray($userData);

        $this->cache->set($userId, $user);

        return $user;
    }
}
```

Multiple solutions

- Possible solutions:
  - Limit cache size
  - TTL
  - WeakMap (from PHP 8.0)

WeakMap to rescue!

- A `WeakMap` is map that accepts objects as keys.
- An object in a key of WeakMap does not contribute toward the object's reference count.
- Offers a partial solution, item will be in cache as long as we have an instance of particular `UserId`.

WeakMap to rescue!

```php
use \Kambo\MemoryLeaks\{UserId,User};

$weakMap = new WeakMap();

$userId = new UserId(1);

$weakMap[$userId] = new User('foo', 'bar', 'baz');

var_dump(count($weakMap)); // int(1)
unset($userId);
var_dump(count($weakMap)); // int(0)
```

```php
namespace Kambo\MemoryLeaks;


class WeakMapBasedCache implements Cache
{
   private \WeakMap $cache;

   public function __construct() {
       $this->cache = new \WeakMap();
   }

   public function set(object $key, mixed $data) : void {
       $this->cache[$key] = $data;
   }

   public function get(object $key) : mixed {
       if ($this->has($key) === false) {
           throw new \LogicException("Value ".$key. " does not exists.");
       }

       return $this->cache[$key];
   }
}
```

Overview

Be aware of…

- Long running scripts.
- Don’t disable memory limit.
- Think about memory usage from beginning => proper architecture from start.
- Generators are not mandatory, but they can help.

Be aware of…

- Objects holding large sets of data, eg.:
  - cache,
  - identity maps (ORM).
- Generally - think outside box (eg.: logged SQLs 😂).

Be aware of…

- Large arrays:
    - Can be unpredictably duplicated.
    - By default they consume more memory then objects.
    - Best way is to wrap them into objects and provide convenient methods.

Be aware of…

- IO handling
    - Handle big files/DB/external data in chunks.
    - Size of development and production data can differ.
    - Use the best method for data parsing. (eg.: pull parsers for large XMLs)

A new challenges!

- FFI,
- Async PHP frameworks and
- alternative PHP process managers.

HOW YOU PLAN TO END TALKS:
AND THUS, THE CONCLUSION IS SELF-EVIDENT.
HOW YOU ALWAYS END TALKS:
UH. SO. YEAH.
SOLUTION: CONVERT IT TO LATIN
VAH. SIC. ETIAM.
smbc-comics.com

Questions?

Slides + source code:

https://github.com/peoplepath/workshop-memory-leaks

RTsoft
Softwarová řešení na míru

Informační systémy na míru

Business analytika

Mobilní aplikace

Weby a E-shopy

SENTRY

Error, performance & release tracking

30. listopad 2022 18:00 hod

Těšíme se na Vás!

Bonus

FFI - Memory leaks

```
function iWillLeakMemory() {
   $a = FFI::new("long[1024]", false);
}

while (true) {
   iWillLeakMemory();
   var_dump(memory_get_usage());
}
```

```
int(408968)
...
int(133528776)

Fatal error: Allowed memory size of 134217728 bytes exhausted (tried to
allocate 8192 bytes) in memoryleak.php on line 4
```

FFI - Memory leaks with true malloc™

```php
function iWillLeakMemory() {
    $a = FFI::new("long[1024]", false, true);
}

while (true) {
    iWillLeakMemory();
    var_dump(memory_get_usage());
}
```

```
int(408968)
...
int(400864)
int(400864)
int(400864)[1]    478479 killed     php memoryleak.php
```

FFI - Memory leaks, debug

- Valgrind can be used for leaks detection.
- This is same as in other C/C++ program.

```
valgrind --leak-check=full php test.php
```

Bug in PHP

```php
function iAlsoLeakMemory() {
   eval('return function() {};');
}

while (true) {
   iAlsoLeakMemory();
   var_dump(memory_get_usage());
}
```

```
int(129834248)
int(129834248)
int(129834248)
```

```
Fatal error: Allowed memory size of 134217728 bytes exhausted (tried to allocate
65536 bytes) in memory-leak2.php(3) : eval()'d code on line 1
```

You can list all variables

```
+----------------+----------------------------+-------------------------------------+
| Item ids       | Item data                  | Children                            |
+----------------+----------------------------+-------------------------------------+
| 0x7fc9fb059200 | Type: array                 |                                     |
|                | Size: 72 B                  |                                     |
|                | Is root: Yes                |                                     |
|                | Execution Frame: <GLOBAL>   |                                     |
|                | Symbol Name: _GET           |                                     |
+----------------+----------------------------+-------------------------------------+
| 0x7fc9fb059220 | Type: array                 |                                     |
|                  | Size: 72 B                |                                     |
|                  | Is root: Yes              |                                     |
|                  | Execution Frame: <GLOBAL> |                                     |
|                  | Symbol Name: _POST        |                                     |
+----------------+----------------------------+-------------------------------------+
| 0x7fc9fb059240 | Type: array                 |                                     |
|                  | Size: 72 B                |                                     |
|                  | Is root: Yes              |                                     |
|                  | Execution Frame: <GLOBAL> |                                     |
|                  | Symbol Name: _COOKIE      |                                     |
+----------------+----------------------------+-------------------------------------+
| 0x7fc9fb059260 | Type: array                 |                                     |
|                  | Size: 72 B                |                                     |
|                  | Is root: Yes              |                                     |
|                  | Execution Frame: <GLOBAL> |                                     |
|                  | Symbol Name: _FILES       |                                     |
+----------------+----------------------------+-------------------------------------+
| 0x7fc9fb059280 | Type: array                 | 0: 0x7fc9fb05e008                   |
|                  | Size: 72 B                |                                     |
|                  | Is root: Yes              |                                     |
|                  | Execution Frame: <GLOBAL> |                                     |
|                  | Symbol Name: argv         |                                     |
+----------------+----------------------------+-------------------------------------+
| 0x7fc9fb0592a0 | Type: int                   |                                     |
|                  | Size: 16 B                |                                     |
|                  | Is root: Yes              |                                     |
|                  | Execution Frame: <GLOBAL> |                                     |
|                  | Symbol Name: argc         |                                     |
```