Vítáme Vás na workshopu

Moderní PHP

Přednáška, live coding, diskuze

@esler @intraworlds

INTRAWORLDS
STRONGER RELATIONS

Ondřej Ešler

Software Architect

IntraWorlds s.r.o.

@esler @intraworlds

Brief history of PHP

- Created in 1994 by Rasmus Lerdorf [1]
- Many names over the years
  - Personal Home Page Tools
  - Forms Interpreter
  - Personal Home Page Construction Kit
  - PHP: Hypertext Preprocessor [2]
- PHP 2.0 1997 (1%)
- PHP 3.0 1998 (10%)
- PHP 4.0 2000 ('Zend Engine' - Zeev and Andi)
- PHP 5.0 2004
- PHP 7.0 2015

Example PHP/FI Code

```
<!--include /text/header.html-->

<!--getenv HTTP_USER_AGENT-->
<!--ifsubstr $exec_result Mozilla-->
  Hey, you are using Netscape!<p>
<!--endif-->

<!--sql database select * from table where user='$username'-->
<!--ifless $numentries 1-->
  Sorry, that record does not exist<p>
<!--endif exit-->
  Welcome <!--$user-->!<p>
  You have <!--$index:0--> credits left in your account.<p>

<!--include /text/footer.html-->
```

source: http://php.net/manual/en/history.php.php

Key features of modern PHP

- Simple to learn C/Perl-like dynamic language
- Namespaces
- Closures
- Generators
- Optional types
- Security
  - Libsodium
- Allows write code in different programming paradigms (moreless)
  - Object-oriented
  - Procedural
  - Functional
- Rich standard library

And many more...

- Basic syntax
- Types
- Variables
- Constants
- Expressions
- Operators
- Control Structures
- Functions
- Classes and Objects
- Namespaces
- Errors
- Exceptions
- Generators
- References Explained
- Predefined Variables
- Predefined Exceptions
- Predefined Interfaces and Classes
- Context options and parameters
- Supported Protocols and Wrappers

Namespaces

```php
namespace MyProject;

const CONNECT_OK = 1;
class Connection { /* ... */ }
function connect() { /* ... */ }
```

source: http://php.net/manual/en/language.namespaces.definition.php

Closure

```
$fib = function(int $n) use(&$fib) : int {
    return ($n === 0 || $n === 1) ? $n : $fib($n - 1) + $fib($n - 2);
};
```

source: https://en.wikipedia.org/wiki/Functional_programming#PHP

Generators

```
function iterable_range(int $start, int $end, int $step=1): iterable {
  for ($i = $start; $i <= $end; $i += $step) {
    yield $i;
  }
}

$range = iterable_range(1, 10, 2);

foreach ($range as $i) {
    echo $i . PHP_EOL;
}
```

Types (Hints)

```php
<?php
//declare(strict_types=1);

function to_int(string $str): int {
    return $str;
}

var_dump(to_int('123')); // return int(123)
var_dump(to_int('abc')); // throw an exception
```

Security - libsodium

```
$alice_kp = sodium_crypto_sign_keypair();
$alice_sk = sodium_crypto_sign_secretkey($alice_kp);
$alice_pk = sodium_crypto_sign_publickey($alice_kp);

$message = 'This is a test message.';
$signature = sodium_crypto_sign_detached($message, $alice_sk);
if (sodium_crypto_sign_verify_detached($signature, $message, $alice_pk)) {
    echo 'OK', PHP_EOL;
} else {
    throw new Exception('Invalid signature');
}
```

source: https://github.com/paragonie/sodium_compat
see: https://github.com/paragonie/halite

Security - passwords

```php
$password = 'heslo123';

$hash = password_hash($password, PASSWORD_DEFAULT, ['cost' => 11]); // using old algorithm

if (password_verify($password, $hash)) {
    echo 'VERIFIED!' . PHP_EOL;

    if (password_needs_rehash($hash, PASSWORD_DEFAULT, ['cost' => 12])) {
        echo 'NEEDS REHASH!' . PHP_EOL;
        // $newHash = password_hash($password, PASSWORD_DEFAULT, ['cost' => 12]);
    }
    // user is authenticated
}
```

Object-oriented

```php
class Point {
  private $x, $y;

  public function __construct(int $x, int $y) {
    $this->x = $x;
    $this->y = $y;
  }

  public function draw() { /* draw a point */ }
}

class Circle extends Point {
  private $radius;
  public function __construct(int $x, int $y, float $radius) {
    $this->radius = $radius;
    parent::__construct($x, $y);
  }

  public function draw() { /* draw a circle */ }
}
```

Procedural

```
namespace Shop\Items {
    function find(string $name): array { return ['name' => $name, 'price' => rand(1, 999) / 10.0]; }
}

namespace Shop\Invoice {
    function bill(array $basket): string {
        return implode(PHP_EOL, array_map(function ($item) {
            return "{$item['name']} ... {$item['price']}";
        }, $basket['items']));
    }
}

namespace {
    $basket['items'][] = Shop\Items\find('Socks');
    $basket['items'][] = Shop\Items\find('Shorts');

    echo Shop\Invoice\bill($basket);
}
```

Functional

```php
function map_reduce(array $values, callable $map, callable
$reduce) {
  $mapped = [];

  $emit = function ($key, $value) use (&$mapped) {
    $mapped[$key][] = $value;
  };

  foreach ($values as $value) {
    $map($value, $emit);
  }

  $emit = function ($key, $value) {
    echo "$key: $value" . PHP_EOL;
  };

  foreach ($mapped as $key => $values) {
    $reduce($key, $values, $emit);
  }
}
```

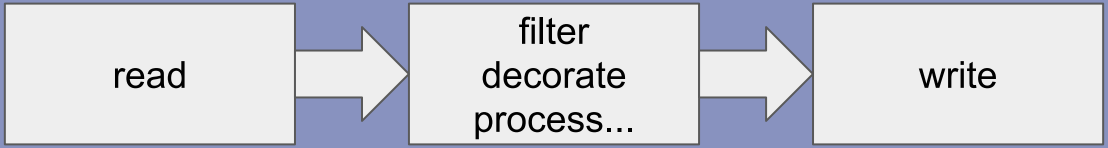
```php
$map = function ($value, callable $emit) {
  $value % 2 ? $emit('liche', $value) : $emit('sude',
$value);
};

$reduceWith = function(callable $callback): Closure {
  return function ($key, $values, $emit) use ($callback) {
    $emit($key, $callback($values));
  };
};

map_reduce(range(1,6), $map, $reduceWith('array_sum'));
```

INTRAWORLDS
STRONGER RELATIONS

Bad “features” of modern PHP

- Simple to learn C/Perl-like dynamic language
- Security
- Allows write code in different programming paradigms (moreless)
- “Rich” standard library
- One request lifecycle?

Let’s do some live coding


read
filter
decorate
process...
write

Discussion

Děkuji!

We’re hiring :-)

INTRAWORLDS
STRONGER RELATIONS