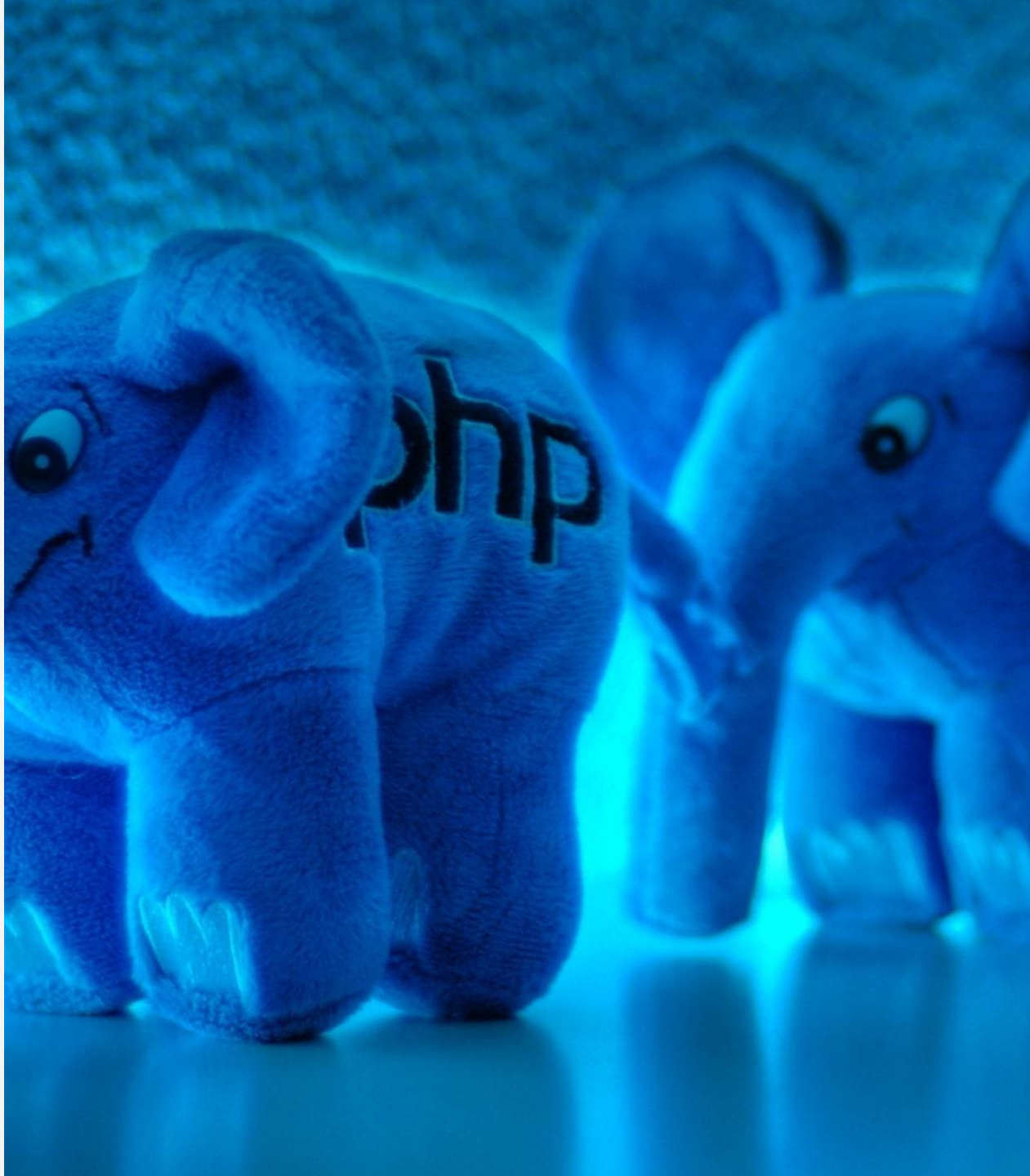
Novinky v PHP 8.*

Antonín Neumann



PeoplePath


ONE DOES NOT SIMPLY
UPGRADE PHP VERSION
imgflip.com

PHP 8.0

26. 11. 2020 - 26. 11. 2023

PHP 8.0 / Named arguments

- Nepovinné paramtry není nutné specifikovat.
- Nezávislé na pořadí a jsou více samopopisné.

- Vhodné zejména pro legacy projekty, kde existuji často metody s mnoho argumenty.
- Nebo projekty se špatnou code base.
- Nebo pro metody kde má více nepoviných parametrů stejnou pravdepodobnost.

PHP 8.0 / Named arguments

```
htmlspecialchars($string, double_encode: false);

// vs

htmlspecialchars($string, ENT_COMPAT | ENT_HTML401, 'UTF-8', false);


// better readability

str_contains(needle: 'Bar', haystack: 'Foobar');
```

PHP 8.0 / Attributes

- Instead of PHPDoc annotations, you can now use structured metadata with PHP's native syntax.

NYNÍ

```
class PostsController
{
    #[Route("/api/posts/{id}", methods: ["GET"])]
    public function get($id) { /* ... */ }
}
```

PHP 8.0 / Attributes

DŘÍVE

```
class PostsController
{
    /**
     * @Route("/api/posts/{id}", methods={"GET"})
     */
    public function get($id) { /* ... */ }
}
```

PHP 8.0 / Constructor property promotion

- Přehlednější definice vlastností třídy.
- Dovoluje definovat viditelnost i datový typ.

NYNÍ

```
class Point {
  public function __construct(
    public float $x = 0.0,
    public float $y = 0.0,
    public float $z = 0.0,
  ) {}
}
```

PHP 8.0 / Constructor property promotion

DŘÍVE

```
class Point {
  public float $x;
  public float $y;
  public float $z;

  public function __construct(
    float $x = 0.0,
    float $y = 0.0,
    float $z = 0.0,
  ) {
    $this->x = $x;
    $this->y = $y;
    $this->z = $z;
  }
}
```

PHP 8.0 / Union types

- Dovoluje deklarovat více datových typů najednou.
- Není nutné k tomu využívat pouze PHPDoc anotace jako dosud.

NYNÍ

```
class Number {
  public function __construct(
    private int|float $number
  ) {}
}

new Number('NaN'); // TypeError
```

PHP 8.0 / Union types

DŘÍVE

```
class Number {
  /** @var int|float */
  private $number;

  /**
   * @param float|int $number
   */
  public function __construct($number) {
    $this->number = $number;
  }
}

new Number('NaN'); // Ok
```

PHP 8.0 / Match expression

- Konstukce **`match`** je velmi podobná konstrukci **`switch`** s několika rozdíly
  - Match je výraz (expression), takže jeho výsledek můžu uložit do proměnné.
  - Match podporuje pouze jednořádkové výrazy a nepotřebuje použití **`break;`**
  - Match dělá striktní porovnání (tedy **`===`**).
  - Match musí pokrýt všechny stavy, jinak **UnhandledMatchError**

PHP 8.0 / Match expression

NYNÍ

```
$result = match ($condition) {
    1, 2 => foo(),
    3, 4 => bar(),
    default => baz(),
}
```

PHP 8.0 / Match expression

DŘÍVE

```
switch ($condition) {
  case 1:
  case 2:
    $result = foo();
    break;
  case 3:
  case 4:
    $result = bar();
    break;
  default:
    $result = baz();
}
```

PHP 8.0 / Saner string to number comparisons

- Změna při porovnání čísla a řetězce.
- Dříve se při porovnání **0 == "something"** převedlo číslo na řetězec a porovnávali se dva řetězce.
- Nyní se převede řetězec na číslo a porovnávají se jako dvě čísla.

```
// PHP 8.0

0 == 'foobar' // false
```

PHP 8.0 / Inheritance with private methods

- Odstranění kontroly pravidel dědičnosti pro private metody.
- Něco se už přeskakovalo (počet a datový typ proměnných), ale další zůstaly:
  - metoda má stejný název jako rodičovská **final private** metody,
  - metoda má stejný název jako rodičovská **static private** metoda a potomek není static (nebo naopak),
  - metoda má stejnž název jako rodičovská private metoda a potomek je **abstraktní**.
- Final a private dohromady vyhodí warning.
  - *Warning: Private methods cannot be final as they are never overridden by other classes in ...*
  - Vyjímku má konstruktor - `final private __construct()`

PHP 8.0 / Inheritance with private methods

```php
class A {
    function callYourPrivate() {
        $this->myPrivate();
    }
    function notOverriden_callYourPrivate() {
        $this->myPrivate();
    }
    final private function myPrivate() {
        echo __METHOD__ . PHP_EOL;
    }
}
```

```php
class B extends A {
    function callYourPrivate() {
        $this->myPrivate();
    }
    private function myPrivate() {
        echo __METHOD__ . PHP_EOL;
    }
}
```

```php
$a = new A();
$a->callYourPrivate(); // A::myPrivate
$a->notOverriden_callYourPrivate(); // A::myPrivate

$b = new B();
$b->callYourPrivate(); // B::myPrivate
$b->notOverriden_callYourPrivate(); // A::myPrivate
```

PHP 8.0 / Others fixes and improvements

- Correct signatures of magic methods
    - __call(), __callStatic(), __get(), __set(), __isset(), __unset(), __sleep(), __wakeup(), __serialize(), __unserialize(), __invoke(), __set_state(), __clone(), and __debugInfo()
- The **@** operator no longer silences fatal errors.
- **Mixed** type (ekvivalent pro **array|bool|callable|int|float|null|object|resource|string**)
- Static return type
- **Consistent type errors for internal functions**
    - Většina interních funkcí nyní vyhazuje Error exception ve chvíli kdy selže validace parametrů

PHP 8.0 / Others fixes and improvements

- Allow a trailing comma in parameter lists and closure use lists
- Non-capturing catches
  - Vyjímka nemusí být zacycena do proměnné
  - `try {} catch (Exception) {}`
- Throw is now an expression
  - Dovoluje vyhazovat vyjímky na dříve nedovolených místech.
  - Např. arrow functions (fn() =>), the coalesce operator (??) a ternary/elvis operator.

PHP 8.0 / Others fixes and improvements

- **Stringable** interface
  - Automaticky všechny objekty které implementují `__toString()` metodu
  - Zavedeno zejména jako podpora pro union typ `string|Stringable`
- Methods for working with strings
  - **str_contains()** místo `strpos()` a `strstr()`
  - **str_starts_with(), str_ends_with()** místo `substr()`, `strpos()/strrpos()`, `strncmp()`
- `token_get_all()` object implementation
  - Vytvoření objektové alternativy **PhpToken**

PHP 8.1

25. 11. 2021 - 25. 11. 2024

PHP 8.1 / Enumerations

- Konečně nativní implementace výčtového datového typu.
- Oproti konstantám nabízí validaci a typovost.

```
enum Status
{
    case Draft;
    case Published;
    case Archived;
}
function acceptStatus(Status $status) {...}
```

PHP 8.1 / Readonly Properties

- Readonly property nemůže být změněna po initializaci, tj. po prvním přiřazení.
- Může být inicializována pouze ve scope kde byla deklarována.
- Skvělé například pro DTO (Data Transfer Object).

DŘÍVE

```
class User
{
    public function __construct(private string $name) {}
    public function getName(): string {
        return $this->name;
    }
}
```

NYNÍ

```
class User
{
    public function __construct(public readonly string $name) {}
}
```

PHP 8.1 / First-class Callable Syntax

- Umožňuje získat referenci na jakoukoli funkci.

```
class Foo {
    public function method() {}
    public static function
staticmethod() {}
    public function __invoke() {}
}

$obj = new Foo();
$classStr = 'Foo';
$methodStr = 'method';
$staticmethodStr = 'staticmethod';
```

```
$f1 = strlen(...);
$f2 = $obj(...); // invokable object
$f3 = $obj->method(...);
$f4 = $obj->$methodStr(...);
$f5 = Foo::staticmethod(...);
$f6 = $classStr::$staticmethodStr(...);
```

UPOZORNĚNÍ: tři tečky (...) jsou součástí syntaxe!

PHP 8.1 / New in initializers

- Objekty (instance) můžou být použité jako:
  - defaultní hodnota parametru (funkce, kontruktoru),
  - statická proměnná,
  - globální konstanta,
  - argumenty atributů (anotace).

```
static $x = new Foo;

const C = new Foo;

function test($param = new Foo) {}

#[AnAttribute(new Foo)]
class Test {
    public function __construct(
        public $foo = new A,
        public $bar = new B(1),
        public $baz = new C(x: 2),
    ) {}
}
```

PHP 8.1 / Pure Intersection Types

- Použijeme intersection type pokud potřebujeme aby nějaká hodnota proměnné splňovala více rozhraní najednou.
- Lze použít pouze pro třídy a rozhraní.
- Ve verzi 8.1 nelze použít intersection a union typy společně.

```
// PHP 8.0
function count_and_iterate(Iterator $value) {
    if (!($value instanceof Countable)) {
        throw new TypeError(
            'value must be Countable'
        );
    }

    foreach ($value as $val) {
        echo $val;
    }

    count($value);
}
```

```
PHP
function count_and_iterate(
    Iterator&Countable $value
){
    foreach ($value as $val) {
        echo $val;
    }

    count($value);
}
```

PHP 8.1 / Never return type

- Návratový typ **never** pomáhá najít tzv. mrtvý kód.
- Jeho použití znamená, že z volané funkce se nikdy nevrátíme.
- Typické použití u funkcí které pouze vyhazují výjímku nebo volají `die()`, `exit()`, `trigger_error()` nebo něco obdobného.

```
function redirect(string $uri): never {
    header('Location: ' . $uri);
    exit();
}

function redirectToLoginPage(): never {
    redirect('/login');
    echo 'Hello'; // <- dead code detected by static analysis
}
```

PHP 8.1 / Final class constants

- Konstanty lze nově deklarovat jako **final**.
- Taková konstanta nelze předefinovat v potomkovi => fatal error.
- Privat konstanty nemůžou být **final** (nedává to smysl).
- Změna: konstanty z rozhraní lze nově přetěžovat.

```
class Foo {
    final public const TEST = '1';
}
class Bar extends Foo {
    public const TEST = '2';
}
// Fatal error: Bar::TEST cannot override final constant Foo::TEST in %s on line %d
```

```
class Foo {
    final private const TEST = '1';
}
// Fatal error: Private constant Foo::TEST cannot be final as it is never overridden in ... on line …
```

PHP 8.1 / Explicit Octal numeral notation

- Původní zápis čísel v osmičkové soustavě s nulou na začátku může být matoucí.
- Nový zápis je více explicitní a používa prefix 0o.

016 === 16; // false **016** je **14**

0o16 === 16; // pořád false, ale je to méně matoucí

- Nový zápis lze kombinovat s **Underscore numeric separator** (PHP 7.4)

```
0o3_705 === 19_89; // true
```

PHP 8.1 / Fibers

- Tzv. lehká vlákna jsou primitivum pro implementaci kooperativní souběžnosti.
- Mělo by nahrazovat `Promise::then()` nebo korutiny založené na generátorech.
- Fiber je kus kódu s vlastním zásobníkem (proměnné, stav).
- Vlákna jsou vytvářena, spouštěna, uspávána a ukončována přímo v kódu.

```
$fiber = new Fiber(function() {
    $last = Fiber::suspend(16);
    echo "Resuming with last value {$last}\n";
});
$last = $fiber->start();
echo "Suspended with last value {$last}\n";
$fiber->resume(42);

// Suspended with last value 16
// Resuming with last value 42
```

PHP 8.1 / Others fixes and improvements

- Array unpacking support for string-keyed arrays
  - dříve pouze pro indexová pole

```
$arrayA = ['a' => 1];
$arrayB = ['b' => 2];
$result = ['a' => 0, ...$arrayA, ...$arrayB];
```

- New **array_is_list** function
  - Funkce vrací true, pokud je pole list. A je list pokud:
    - Obsahuje pouze celočíselné indexy.
    - Začíná indexem 0 a žádný index není přeskočen.
  - Prázdné pole je list.

PHP 8.1 / Others fixes and improvements

- $_FILES: New **full_path** value for directory-uploads
  - Pole po nahrání souboru obsahuje celou cestu.
  - ```
    ["full_path"]=> array(1) {
          [0]=> string(19) "foo/test1/file.txt"
     }
    ```
- MySQLi: Bind in Execute
  - Funkce `mysqli_stmt::execute` podporuje parametr `$params`, čímž se přibližuje metodě `PDOStatement::execute`

    ```
    $statement = $db->prepare('SELECT * FROM posts WHERE pid = ?');
    ```
  - ```
    $statement->execute([$postId]);
    ```

PHP 8.2

8. 12. 2022 - 8. 12. 2025

PHP 8.2 / Readonly classes

- Od verze PHP 8.2 je povoleno deklarovat celou třídu jako readonly.
- Potom všechny property dané třídy jsou automaticky readonly (platí stejná pravidla).
- Pokud je třída readonly, tak:
  - musí mít všechny proměnné datový typ,
  - nesmí obsahovat žádné dynamické proměnné,
  - vlastnosti se nedá zřeknout.
- Potomek readonly třídy musí být také readonly.

PHP 8.2 / Readonly classes

```
// PHP 8.1
class User {
    public readonly int $uid;
    public readonly string $username;
}


// PHP 8.2
readonly class User {
    public int $uid;
    public string $username;
}
```

PHP 8.2 / DNF Types

- Disjunctive Normal Form
- Dovoluje kombinovat Union types a Intersection types
  - `(A&B)|null`
- `Nejsou dovolené duplicity (A&B)|(B&A)`

- Toto je DNF
  - `A|B|C`
  - `A|B|(C&D)`
  - `(A&B&C)|null`

- Toto není DNF
  - `A&(B|C)`
  - `A|(B&(C|D))`

PHP 8.2 / null, false, and true as stand-alone types

- Datové typy **null** a **false** mohli být součástí union typu (string|null).
  - **False** byl přidán hlavně kvůli starším PHP funkcím, které vrací hodnotu nebo **false** při chybě.
  - **Null** nemůže být kombinován s nullable opetátorem (~~?null~~)
- Nyní je možné je používat samostatně.
- Duplicita mezi návratovou hodnotou **null** a **void** (jiná sémantika).
- **True** bylo přidáno protože prostě proto :-D
- **True** ani **false** nemůžou být kombinovány s **bool** ani mezi sebou vzájemně (~~bool|false|true~~)

```
class User {}

interface UserFinder
{
    function findUserByEmail(): User|null;
}

class AlwaysNullUserFinder implements UserFinder
{
    function findUserByEmail(): null {
        return null;
    }
}
```

PHP 8.2 / New "Random" extension

- Nová core extension random, všechny funkce jsou přesunuty pod ní.
- Nový namespace \Random a nová třída Random\Randomizer.

```
$randomizer = new Random\Randomizer();

echo $randomizer->getInt(1, 100); // 42

echo $randomizer->shuffleBytes('lorem ipsum'); // "ols mpeurim"

$randomizer->shuffleArray(['apple', 'banana', 'orange']);
```

PHP 8.2 / Constants in traits

- Podpora deklarace konstant přímo v traitu.
- Pro přístup ke konstantě musíme používat název třídy využívající trait.

```
trait Foo {
    public const CONSTANT = 1;
}

class Bar {
    use Foo;
}

var_dump(Bar::CONSTANT); // 1
var_dump(Foo::CONSTANT); // Error

var_dump(Bar::C === Baz::C);
```

PHP 8.2 / Deprecate dynamic properties

- Dynamické vytváření proměnných objektu je nově zastaralé.
- Magické funkce **__get()** a **__set()** jsou stále povoleny.
- Dynamické proměnné u **stdClass** jsou také povolené.
- Lze povolit attributem **#[\AllowDynamicProperties]**, který je také nově přidaný.

```
class User {
    public $name;
}

$user = new User();
$user->last_name = 'Doe'; // Deprecated notice

$user = new stdClass();
$user->last_name = 'Doe'; // Still allowed
```

PHP 8.2 / New mysqli::execute_query method

- Nová metoda, přímočařejší použití.
- Existuje samozřejmě i alias v podobě neobjektové funkce **mysqli_execute_query**.

```
// dříve
$query = 'SELECT uid, username FROM users WHERE uid = ?';
$statement = $connection->prepare($query);
$statement->bind_param('s', $uid);
$statement->execute();
$result = $statement->get_result();
```

```
// nyní
$query = 'SELECT uid, username FROM users WHERE uid = ?';
$result = $mysqli->execute_query($sql, [$uid]);
```

PHP 8.2 / New #[\SensitiveParameter] attribute

- Umožnuje zakázat výpis citlivých hodnot ve stack trace.

```
function login(string $username, #[\SensitiveParameter] string $password) {
    throw new \Exception('Error');
}
```

```
// PHP 8.1
Fatal error: Uncaught Exception: error in /home/user/scripts/code.php:20
Stack trace:
#0 /home/user/scripts/code.php(24): login('a', 'heslo')
#1 {main} thrown in /home/user/scripts/code.php on line 20

// PHP 8.2
Fatal error: Uncaught Exception: error in /home/user/scripts/code.php:20
Stack trace:
#0 /home/user/scripts/code.php(24): login('a', Object(SensitiveParameterValue))
#1 {main} thrown in /home/user/scripts/code.php on line 20
```

PHP 8.2 / Others fixes and improvements

- Deprecated ${} string interpolation.
  - místo echo("Hello ${name} ${$var}");
  - nově echo("Hello {$name} {$$var}");
- New **ReflectionFunction::isAnonymous** and **ReflectionMethod::hasPrototype** methods.
- Funkce **strtolower()** a **strtoupper()** již neovlivňuje nastaveni locale.

PHP 8.3

PHP 8.3 / Added json_validate function

- Vrací true pokud JSON string je validní.
- Chyby vrací pomocí existujicích funkcí **json_last_error()** a **json_last_error_msg()**.
- Jako flag je možné použít pouze **JSON_INVALID_UTF8_IGNORE**, pokud je použit funkce json_validate() ignoruje UTF-8 znaky.
    - `json_validate("[\"\xc1\xc1\",\"a\"]"); // false`

```
/**
 * Validates a given string to be valid JSON.
 *
 * @param string $json String to validate
 * @param int $depth Set the maximum depth. Must be greater than zero.
 * @param int $flags Bitmask of flags.
 *
 * @return bool True if $json contains a valid JSON string, false otherwise.
 */
function json_validate(string $json, int $depth = 512, int $flags = 0): bool {}
```

PHP 8.3 / Dynamic class constant fetch support

- Je možné volat kontantu nebo Enum pomocí proměnné

```
class MyClass {
    public const MY_CONST = 42;
}

$constName = 'MY_CONST';
echo MyClass::{$constName};


enum MyEnum: int {
    case MyMember = 42;
}

$enumName = 'MyMember';
echo MyEnum::{$enumName}->value;
```

PHP 8.3 / extending gc_status() function

- Funkce gc_status() vrací nyní 4 hodnoty:
  - runs (int, kolikrát GC běžel),
  - collected (int, počet sesbíraných objektů),
  - threshold (int, počet kořenů v bufferu, které spustí GC),
  - roots (int, aktuální počet kořenů v bufferu).
- Verze 8.3 přidá 4 nové hodnoty:
  - running (bool, true pokud GC aktuálně běží),
  - protected (bool, true pokud GC je true if the garbage collector is protected and root additions are forbidden),
  - full (bool, true po GC buffer dosáhne maximalní velikosti GC_MAX_BUF_SIZE),
  - buffer_size (int, aktuální velikost GC bufferu).

PHP 8.3 / New \Random\Randomizer methods

- **\Random\Randomizer::getBytesFromString()**
  - Vrací náhodnou sekveci bytů určené délky ze zadaného řetezce.
  - ```
    $rng = new Random\Randomizer();
    $rng->getBytesFromString('AEIOU', 10); // "IEAUAIIOUE"
    ```
- **\Random\Randomizer::getFloat**(float $min, *float* $max, $boundary**)** a **nextFloat()**
  - Metoda getFloat vrací náhodné desetinné číslo mezi min a max.
  - Metoda nextFloat vrací náhodné desetinné číslo mezi 0 a 1 (\Random\IntervalBoundary::ClosedOpen).
    ```

Díky
&
RTFM