What's new in PHP 8.2?

by Enrico Zimuel

December 15 2022 - PHP User Group Torino (Italy)

PUG Torino

- PHP User Group Torino is a group of web developers interested in the PHP language (and not only)
- We are part of GrUSP association
- On meetup.com we are about 589 members
- We have also a mailing list with more than 100 members
- In the past we organized conferences like PHP.TO.START (2011, 2012, 2013) and Zend Framework Day (2013)
- Toolbox Coworking is sponsoring the group

TOOLBOX
COWORKING

PHP 8.2

- Release date: 8 December 2022
- New features: Readonly classes, Disjunctive Normal Form (DNF) types, new "Random" extension, constants in Traits, etc
- 118 bug fixes

PHP supported versions

Branch	Initial Release		Active Support Until		Security Support Until	
8.0	26 Nov 2020	*2 years ago*	26 Nov 2022	*18 days ago*	26 Nov 2023	*in 11 months*
8.1	25 Nov 2021	*1 year ago*	25 Nov 2023	*in 11 months*	25 Nov 2024	*in 1 year, 11 months*
8.2	8 Dec 2022	*6 days ago*	8 Dec 2024	*in 1 year, 11 months*	8 Dec 2025	*in 2 years, 11 months*

Readonly properties (PHP 8.1+)

```php
class Foo
{
    public string $baz;

    public function __construct(readonly public string $bar)
    { }
}

$foo = new Foo('hello');
echo $foo->bar; // hello

$foo->baz = 'hi';
$foo->bar = 'hi'; // Fatal error: Cannot modify readonly property Foo::$bar
```

Immutability

- A readonly property does not provide immutability for objects

```php
class User {
    public string $username;
}
class Post {
    public function __construct(readonly public User $author)
    { }
}
$user = new User();
$user->username = 'Foo';

$post = new Post($user);
$user->username = 'Bar';
echo $post->author->username; // Bar
```

Providing immutability

```php
class User {
    public string $username;
}
class Post {
    readonly public User $author;
    public function __construct(User $author)
    {
        $this->author = clone $author;
    }
}
$user = new User();
$user->username = 'Foo';
$post = new Post($user);
$user->username = 'Bar';
echo $post->author->username; // Foo
```

Readonly classes

```php
readonly class Foo
{
    public function __construct(public string $bar)
    { }
}

$foo = new Foo('hello');
echo $foo->bar; // hello

$foo->bar = 'hi';
// Fatal error: Cannot modify readonly property Foo::$bar
```

Readonly property rules

- Only be initialized from within the class scope
- Cannot be modified once initialized
- Cannot be unset once initialized
- Must be a typed property
- Cannot be static
- Cannot have default values

Readonly class rules

- Must only contain typed properties
- Must not use dynamic properties
- Must not use #[AllowDynamicProperties] attribute
- Cannot opt-out of the read-only status
- A subclass must be declared readonly

Disjunctive Normal Form (DNF) type

- Before PHP 8.2

```php
interface A {}
interface B {}

class Foo
{
   public function setBar(mixed $bar)
   {
        if (($bar instanceof A && $bar instanceof B) || $bar === null) {
            $this->bar = $bar;
            return;
        }
        throw new Exception('Invalid type');
   }
}
```

DNF type with PHP 8.2

- DNF uses union and intersection types, following a strict rule: when combining union and intersection, this last must be grouped with brackets

```php
interface A {}
interface B {}

class Foo
{
    public function setBar((A&B)|null $bar)
    {
        $this->bar = $bar;
    }
}
```

null, false and true as stand-alone types

```
function alwaysReturnsFalse(): false {}

function alwaysReturnsNull(): null {}

function alwaysReturnsTrue(): true {}

function foo(true|null $bar) {}
```

Random extension

- PHP 8.2 refactored all of the RNG-related functionality to a new extension named Random
- The Random extension provides the same functionality without breaking any APIs, so the existing rand, mt_rand, random_bytes, and random_int functions continue to work with no changes
- It also provides a new OOP API to generate random numbers with a pluggable architecture, so it is now easy to mock the RNG and provide new RNGs, making PHP applications secure and easy to test

Constants in traits

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

class Bar
{
    use Foo;
}

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

Deprecate dynamic properties

```php
class User
{
    public $name;
}

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

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

Deprecate dynamic properties (2)

- The creation of dynamic properties is deprecated to help avoid mistakes and typos
- A class can allow dynamic properties using the **#[\AllowDynamicProperties]** attribute
- stdClass allows dynamic properties
- Usage of the __get() and __set() magic methods is not affected by this change

Allow dynamic properties

```php
#[\AllowDynamicProperties]
class User
{
    public $name;
}

$user = new User();
$user->last_name = 'Doe'; // No deprecation notice
```

Sensitive parameters

- In PHP 8.2 we can use the **#[\SensitiveParameter]** attribute to mark a parameter that is sensitive and should have its value redacted if present in a stack trace

Example: sensitive parameter

```
function login(string $username, string $password)
{
    debug_print_backtrace();
}
login("test", "secret");
#0 /path/to/script.php(8): login('test', 'secret')
```

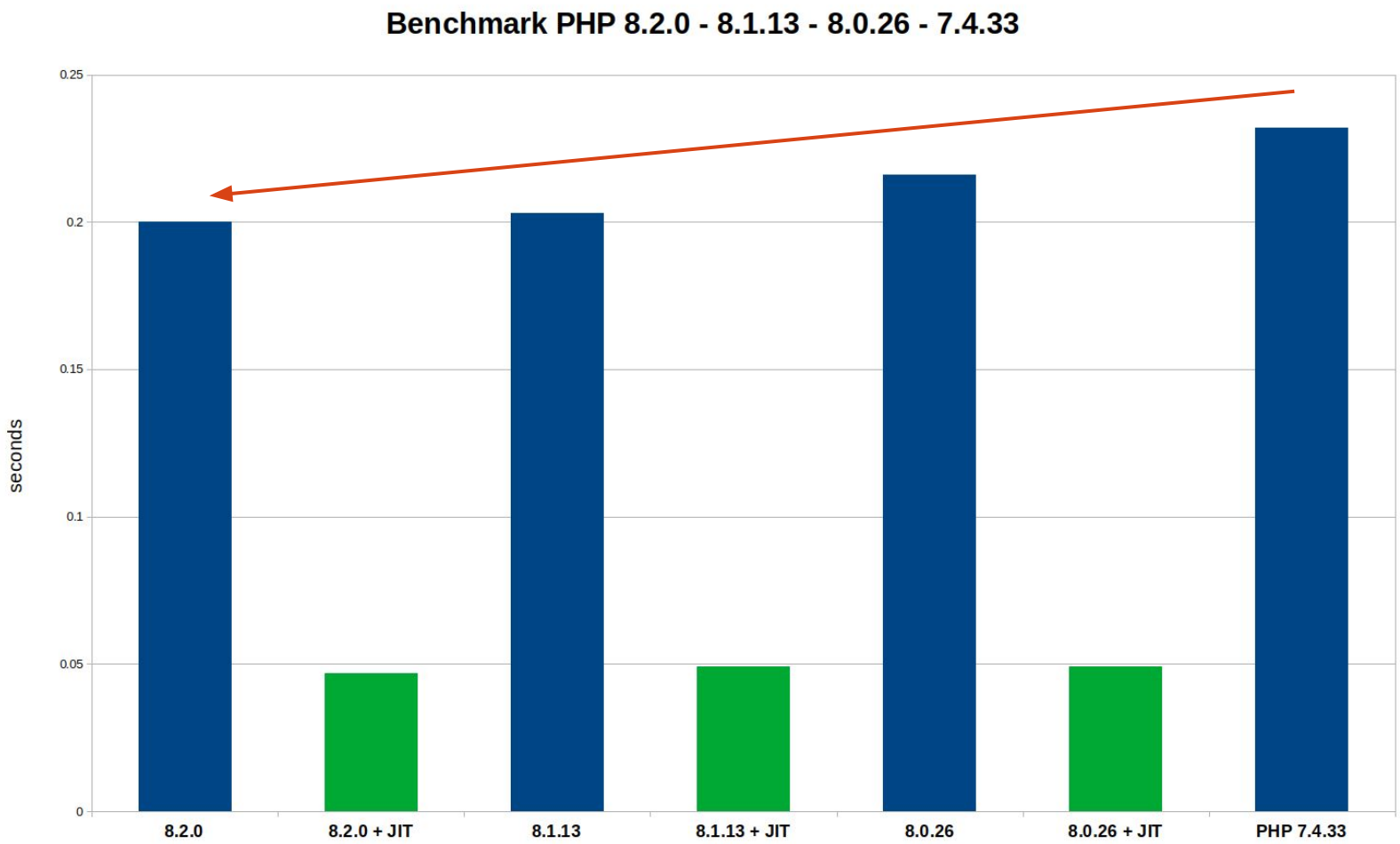
```
function login(string $username, #[\SensitiveParameter] string $password)
{
    debug_print_backtrace();
}
login("test", "secret");
#0 /path/to/script.php(8): login('test', Object(SensitiveParameterValue))
```

utf8_encode/decode deprecated

- utf8_encode() and utf8_decode() functions convert strings between **ISO-8859-1** (“Latin 1”) and **UTF-8** encodings
- These functions do not attempt to detect the actual character encoding and always convert between Latin 1 and UTF-8, even if the source text is not encoded in Latin 1
- Use mbstring extension and mb_convert_encoding() function
- There’s also a polyfill library for mbstring

Benchmark PHP 8.2

- I did a benchmark of PHP 8.2 comparing with 8.1.13, 8.0.26 and 7.4.33
- I took the execution times of Zend\bench.php script
- This script tests the PHP core language using math operators, nested loops, array, strings and recursive functions
- Results:
  - PHP 8.2.0 is **15% faster** than PHP 7.4.33
  - PHP 8.2.0 is **8% faster** than PHP 8.0.26
  - PHP 8.2.0 is the same of PHP 8.1.13


Benchmark PHP 8.2.0 - 8.1.13 - 8.0.26 - 7.4.33
seconds
0.25
0.2
0.15
0.1
0.05
0
8.2.0
8.2.0 + JIT
8.1.13
8.1.13 + JIT
8.0.26
8.0.26 + JIT
PHP 7.4.33

References

- The official PHP 8.2 release note
- Brent, What's new in PHP 8.2
- PHP.Watch, PHP 8.2: What's New and Changed
- PHP.Watch, PHP 8.2: readonly Classes
- PHP.Watch, PHP 8.2: New Random Extension
- Parvej Ahammad, Features that are accepted for PHP 8.2, Medium
- Ignas R., What's New in PHP 8.2: New Features, Deprecations, and Bug Fixes, Hostinger blog
- Enrico Zimuel, Benchmarking PHP 8.2

Thanks!

Contact:
https://torino.grusp.org/