What's New in PHP 8.4!

ConFoo 2025

Derick Rethans

derick@php.net — @derickr@phpc.social

https://derickrethans.nl/talks/php-confoo25

About Me

Derick Rethans

- I'm European, living in London
- ~~PHP 7.4 **Release Manager**~~
- **PHP Foundation**
- **Xdebug** & PHP's Date/Time support
- I ♥🌍 maps, I ♥🍺 beer, I ♥🥃 whisky
- mastodon: @derickr@phpc.social

PHP 8.4: Asymmetric Visibility

```php
<?php
class Reader
{
    /**
     * @var array[Track]
     */
    private array $tracks;

    function __construct( string $fileName )
    {
        $this->parseGpx( $fileName );
    }

    private function parseGpx( $fileName )
    {
        // sets $this->tracks
    }

    public function getTracks() : array
    {
        return $this->tracks;
    }
}
?>
```

PHP 8.4: Asymmetric Visibility

```php
<?php
class Reader
{
    /**
     * @var array[Track]
     */
    public private(set) array $tracks;

    function __construct( string $fileName )
    {
        $this->parseGpx( $fileName );
    }

    private function parseGpx( $fileName )
    {
        // sets $this->tracks
    }




}
?>
```

PHP 8.4: Asymmetric Visibility

```php
<?php
class Reader
{
    /**
     * @var array[Track]
     */
    private(set) array $tracks;

    function __construct( string $fileName )
    {
        $this->parseGpx( $fileName );
    }

    private function parseGpx( $fileName )
    {
        // sets $this->tracks
    }




}
?>
```

PHP 8.4: Asymmetric Visibility

- Only works on typed properties
- `set` visibility must be equal or lesser than main (`get`) visbility
- Using `$obj->array[]` follows `set` visibility
- Inherited properties must have the same time, and might have a wider visibility
- `private(set)` also implies `final`
- `readonly` (without `private(set)`) is analogous to `protected(set)` (+ write once semantics)

PHP 8.4: Property Hooks

```php
<?php
class User implements Named
{
    private bool $isModified = false;

    public function __construct(private string $first, private string $last) {}

    public string $fullName {
        // Override the "read" action with arbitrary logic.
        get => $this->first . " " . $this->last;

        // Override the "write" action with arbitrary logic.
        set {
            [$this->first, $this->last] = explode(' ', $value);
            $this->isModified = true;
        }
    }
}
?>
```

PHP 7.4: Typed Properties

```
<?php
class ezcMailImapTransportOptions
{
    protected $properties;

    public function __construct( array $options = array() )
    {
        $this->uidReferencing = false;
    }

    public function __set( $name, $value )
    {
        switch ( $name ) {
            case 'uidReferencing':
                if ( !is_bool( $value ) ) {
                    throw new ezcBaseValueException( $name, $value, 'bool' );
                }
                $this->properties[$name] = $value;
                break;
        }
    }

    // …
}
?>
```

PHP 7.4: Typed Properties

```php
<?php
class ezcMailImapTransportOptions
{
    public bool $uidReferencing;

    public function __construct( array $options = array() )
    {
        $this->uidReferencing = false;
    }

?>
```

PHP 8.4: Real Life Example with Property Hooks

```php
<?php
class ezcMailImapTransportOptions
{
    protected $properties;

    public function __construct( array $options = array() )
    {
        $this->listLimit = 0;
    }

    public function __set( $name, $value )
    {
        switch ( $name ) {
            case 'listLimit':
                if ( !is_int( $value ) || $value < 0 ) {
                    throw new ezcBaseValueException( $name, $value, 'int >= 0' );
                }
                $this->properties[$name] = $value;
                break;
        }
    }

    // …
}
?>
```

PHP 8.4: Real Life Example with Property Hooks

```php
<?php
class ezcMailImapTransportOptions
{
    public int $listLimit {
        set {
            if ( $value < 0 ) {
                throw new ValueError( "listLimit set to <{$value}>, but must be >=
            }
        }
    }

    public function __construct( array $options = array() )
    {
        $this->listLimit = 0;
    }
}
?>
```

PHP 8.4: New Without Parentheses

```php
<?php
class Request implements Psr\Http\Message\RequestInterface
{
    // ...
}

$request = (new Request())->withMethod('GET')->withUri('/hello-world');
?>
```

PHP 8.4: New Without Parentheses

```php
<?php
class Request implements Psr\Http\Message\RequestInterface
{
    // ...
}

$request = new Request()->withMethod('GET')->withUri('/hello-world');
?>
```

PHP 8.4: New Without Parentheses

```php
<?php
class MyClass extends ArrayObject
{
    const CONSTANT = 'constant';
    public static $staticProperty = 'staticProperty';
    public static function staticMethod(): string { return 'staticMethod'; }
    public $property = 'property';
    public function method(): string { return 'method'; }
    public function __invoke(): string { return '__invoke'; }
}

var_dump(
    new MyClass()::CONSTANT,         // string(8)  "constant"
    new MyClass()::$staticProperty, // string(14) "staticProperty"
    new MyClass()::staticMethod(),  // string(12) "staticMethod"
    new MyClass()->property,         // string(8)  "property"
    new MyClass()->method(),         // string(6)  "method"
    new MyClass()(),                 // string(8)  "__invoke"
    new MyClass(['value'])[0],       // string(5)  "value"
);
?>
```

PHP 8.4: Parsing HTML5

HTML 5 parsing, through the following new class hierachy:

```
<?php
namespace DOM {
    abstract class Document extends DOM\Node implements DOM\ParentNode {
        /* all properties and methods that are common and sensible for both XML
         * & HTML documents */
    }

    final class XMLDocument extends Document {
        /* XML specific properties and methods */
    }

    final class HTMLDocument extends Document {
        /* HTML specific properties and methods */
    }
}

class DOMDocument extends DOM\Document {
    /* Keep methods, properties, and constructor the same as they are now */
}
?>
```

PHP 8.4: Lazy Objects

PHP 8.4: Lazy Objects

PHP 8.0: Just-In-Time (JIT) Compiler

- Compiles PHP code to x86 machine code
- Performance improvements mostly apply to math heavy code
- Part of opcache:

```
opcache.jit=on
opcache.jit_buffer_size=128M
```

PHP 8.4: Just-In-Time (JIT) Compiler

- Now based on a separately developed Intermediate Representation
- Should be easier to maintain, and support multiple targets
- Still part of opcache, and replaces the old JIT

Also changes to default settings:

```
opcache.jit=disable
opcache.jit_buffer_size=64M
```

PHP 8.4: #[Deprecated] Attribute

```php
<?php
#[\Deprecated("use test() instead", since: "2.4")]
function test3() {
}

test3();
?>
```

Result:

```
Deprecated:  Function test3() is deprecated since 2.4, use test() instead in /home/
```

PHP 8.4: #[Deprecated] Attribute

```php
<?php
class Clazz {
    #[\Deprecated]
    public const OLD_WAY = 'foo';

    #[\Deprecated("use test() instead", since: "2024-10-08")]
    function test2() {
    }
}

$c = new Clazz()->test2();
echo Clazz::OLD_WAY, "\n";
?>
```

Result:

```
Deprecated:  Method Clazz::test2() is deprecated since 2024-10-08, use test() inst


Deprecated:  Constant Clazz::OLD_WAY is deprecated in /home/httpd/pres2/show2.php(

foo
```

PHP 8.4: #[Deprecated] Attribute

```
<?php
enum MyEnum {
    #[\Deprecated]
    case OldCase;
}

var_dump(MyEnum::OldCase);
?>
```

Result:

```
Deprecated:  Enum case MyEnum::OldCase is deprecated in /home/httpd/pres2/show2.ph

enum(MyEnum::OldCase)
```

PHP 8.4: #[Deprecated] Attribute

```php
<?php
enum MyEnum {
    #[\Deprecated]
    case OldCase;
}

var_dump(MyEnum::OldCase);
?>
```

Result:

```
Deprecated:  Enum case MyEnum::OldCase is deprecated in /home/httpd/pres2/show2.ph

enum(MyEnum::OldCase)
```

- Can be set on: Methods, Functions, Constants
- The `since` parameter is a free-form string
- Can also be used by PHP's internals

PHP 8.4: PDO Subclasses

Each PDO driver now has it's own class, extending `\PDO`, containing (a copy of) driver specific elements

- `Pdo\Dblib`, `Pdo\Firebird`, and `Pdo\Odbc` are just children, without any changes.
- `Pdo\Mysql` has `getWarningCount()` defined on it
- `Pdo\Pgsql` and `Pdo\Sqlite` have a whole bunch of constants and methods
- `pdo_oci` has been moved to PECL: `https://github.com/php/pecl-database-pdo_oci`

A new factory method, `PDO::connect` returns an object representing one of the new subclasses:

```
<?php
$db = PDO::connect('sqlite::memory:');
if (!$db instanceof Pdo\Sqlite) {
    // ERROR ZONE
}
```

PHP 8.4: Rounding Modes

```php
<?php
$values = [ -8.5, -8.45, 8.45, 8.5 ];

foreach (RoundingMode::cases() as $mode) {
    printf( '%20s: ', $mode->name );

    foreach ($values as $value) {
        printf( '%5s', round($value, 1, $mode) );
    }
    foreach ($values as $value) {
        printf( '%5s', round($value, 0, $mode) );
    }
    echo "\n";
}
```

Result:

```
    HalfAwayFromZero:  -8.5 -8.5  8.5  8.5   -9   -8    8    9
     HalfTowardsZero:  -8.5 -8.4  8.4  8.5   -8   -8    8    8
            HalfEven:  -8.5 -8.4  8.4  8.5   -8   -8    8    8
             HalfOdd:  -8.5 -8.5  8.5  8.5   -9   -8    8    9
          TowardsZero:  -8.5 -8.4  8.4  8.5   -8   -8    8    8
         AwayFromZero:  -8.5 -8.5  8.5  8.5   -9   -9    9    9
    NegativeInfinity:  -8.5 -8.5  8.4  8.5   -9   -9    8    8
    PositiveInfinity:  -8.5 -8.4  8.5  8.5   -8   -8    9    9
```

PHP 8.4: BCrypt Cost

PHP's default BCrypt cost for `password_hash()` is unchanged since the addition of the password hashing API in PHP 5.5 which was 11 years ago.

- The current cost is `10`
- Increasing the cost by one, doubles the time
- New cost in PHP 8.4: `12` (~less than 330ms per hash)

PHP 8.4: grapheme_str_split

```php
<?php
foreach (str_split("🤦") as $element) {
    printf("%8s : %s\n", bin2hex($element), $element);
}
?>
```

PHP 8.4: grapheme_str_split

```
<?php
foreach (str_split("🤦") as $element) {
    printf("%8s : %s\n", bin2hex($element), $element);
}
?>
```

Result:

```
      f0 : 
      9f : 
      99 : 
      87 : 
      e2 : 
      80 : 
      8d : 
      e2 : 
      99 : 
      82 : 
      ef : 
      b8 : 
      8f : 
```

PHP 8.4: grapheme_str_split

```php
<?php
foreach (mb_str_split("🤦") as $element) {
    printf("%8s : %s\n", bin2hex($element), $element);
}
?>
```

PHP 8.4: grapheme_str_split

```php
<?php
foreach (mb_str_split("🤦‍♂️") as $element) {
    printf("%8s : %s\n", bin2hex($element), $element);
}
?>
```

Result:

```
f09f9987 : 🤦
  e2808d :
  e29982 : ♂
  efb88f :
```

PHP 8.4: grapheme_str_split

```php
<?php
foreach (grapheme_str_split("🤦‍♂️") as $element) {
    printf("%8s : %s\n", bin2hex($element), $element);
}
```

PHP 8.4: grapheme_str_split

```php
<?php
foreach (grapheme_str_split("🤦") as $element) {
    printf("%8s : %s\n", bin2hex($element), $element);
}
```

Result:

```
f09f9987e2808de29982efb88f : 🤦
```

PHP 8.4: grapheme_str_split

```php
<?php
foreach (grapheme_str_split("🤦") as $element) {
    printf("%8s : %s\n", bin2hex($element), $element);
}
```

Result:

```
f09f9987e2808de29982efb88f : 🤦
```

```php
<?php
foreach (grapheme_str_split("Hå̛llo 🇫🇷!") as $element) {
    printf("%20s : %s\n", bin2hex($element), $element);
}
```

PHP 8.4: grapheme_str_split

```php
<?php
foreach (grapheme_str_split("🤦") as $element) {
    printf("%8s : %s\n", bin2hex($element), $element);
}
```

Result:

```
f09f9987e2808de29982efb88f : 🤦
```

```php
<?php
foreach (grapheme_str_split("Hą̆llo 🇫🇷!") as $element) {
    printf("%20s : %s\n", bin2hex($element), $element);
}
```

Result:

```
                  48 : H
      61cc86cca2cca5 : ą̆
                  6c : l
                  6c : l
                  6f : o
                  20 :
    f09f87abf09f87b7 : 🇫🇷
                  21 : !
```

PHP 8.4: Deprecate Implicit Nullability

```php
<?php
class Rst
{
    public function __construct( RstOptions $options = null )
    {
    }
}
```

```
Deprecated: Rst::__construct(): Implicitly marking parameter $options as nullable is deprecated, the explicit nullable type must be used
```

PHP 8.4: Deprecate Implicit Nullability

```php
<?php
class Rst
{
    public function __construct( RstOptions $options = null )
    {
    }
}
```

```
Deprecated: Rst::__construct(): Implicitly marking parameter $options as nullable is deprecated, the explicit nullable type must be used
```

```php
<?php
class Rst
{
    public function __construct( ?RstOptions $options = null )
    {
    }
}
```

PHP 8.4: Deprecate Implicit Nullability

```php
<?php
class Rst
{
    public function __construct( RstOptions $options = null )
    {
    }
}
```

```
Deprecated: Rst::__construct(): Implicitly marking parameter $options as nullable is deprecated, the explicit nullable type must be used
```

```php
<?php
class Rst
{
    public function __construct( ?RstOptions $options = null )
    {
    }
}
```

```php
<?php
class Rst
{
    public function __construct( RstOptions|null $options = null )
    {
    }
}
```

Policies and Procedures

- Information hidden away in many RFCs
- No consolidated documentation

https://github.com/php/policies

Policies and Procedures

- Information hidden away in many RFCs
- No consolidated documentation

`https://github.com/php/policies`

Still to do:

- Rewrite documents to be statements of facts instead of intent
- Figure out which information isn't written down authoritativily yet

Release Cycle Update

Current:

- 2 + 1 years of support
- 3 alphas, 3, betas, 6 RCs
- No new minor features in beta releases
- New minor features could be added during RC and bug fix releases
- Support ends exactly 3 years after initial release

Release Cycle Update

New:

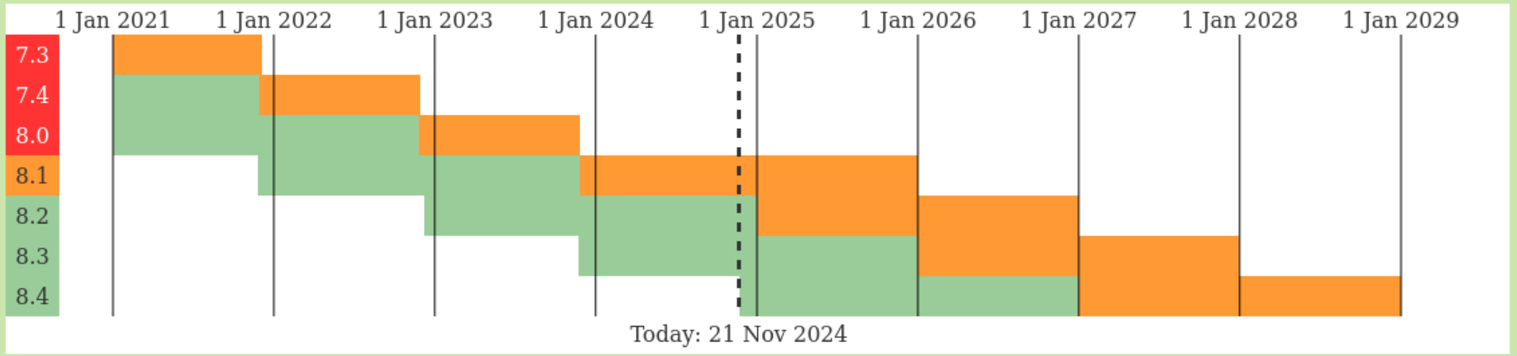
- 2 + 2 years of support
- 3 alphas, 3, betas, 4 RCs
- New minor features in beta releases
- New minor features may not be added during RC and bug fix releases
- Support ends at the end of year 4 after initial release

Release Cycle Update

New:

- 2 + 2 years of support
- 3 alphas, 3, betas, 4 RCs
- New minor features in beta releases
- New minor features may not be added during RC and bug fix releases
- Support ends at the end of year 4 after initial release

Effective Immediately



Any Queries?

Resources

Slides

`https://derickrethans.nl/talks/php-confoo25`

`https://xdebug.cloud`

Derick Rethans — @derickr@phpc.social — derick@php.net