What's New in PHP 8.[45]

International PHP Conference

Derick Rethans

derick@php.net — @derickr@phpc.social

https://derickrethans.nl/talks/php-ipc25

About Me

Derick Rethans

- I'm European, living in London
- **PHP Foundation**
- **Xdebug** & PHP's Date/Time support
- I am groot!
- 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 type, and might have a wider visibility
- `private(set)` also implies `final`
- `readonly` (without `private(set)`) is analogous to `protected(set)` (+ write once semantics)

PHP 8.5: Asymmetric Visibility for Statics

Was missing in PHP 8.4, but turned out to be easy.

```
<?php
class Example
{
    public private(set) static string $classTitle = 'Example class';

    // Implicitly public-read, just like object properties.
    protected(set) static int $counter = 0;

    public static function changeName(string $name): void
    {
        // From private scope, so this is allowed.
        self::$classTitle = $name;
    }
}

print Example::$classTitle; // Allowed.

Example::$classTitle = 'Nope'; // Disallowed.
?>
```

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 8.5: Closures in Constant Expressions

```php
<?php
function slugger(
    string $input,
    array $callbacks = [
        static function ($value) { return \strtolower($value); },
        static function ($value) { return \preg_replace('/[^a-z]/', '-', $value); },
        static function ($value) { return \trim($value, '-'); },
        static function ($value) { return \preg_replace('/-+/', '-', $value); },
    ]
) {
    foreach ($callbacks as $callback) {
        $input = $callback($input);
    }
    return $input;
}

var_dump(slugger('Hello, World!')); // string(11) "hello-world"
?>
```

- They must be `static`
- They may not include variables from the surrounding scope
  no: `use($foo)`, nor: short closures (`fn ... =>`)

PHP 8.5: First Class Callables in Constant Expressions

```php
<?php
function slugger(
    string $input,
    array $callbacks = [
        \strtolower(...),
        static function ($value) { return \preg_replace('/[^a-z]/', '-', $value); },
        static function ($value) { return \trim($value, '-'); },
        static function ($value) { return \preg_replace('/-+/', '-', $value); },
    ]
) {
    foreach ($callbacks as $callback) {
        $input = $callback($input);
    }
    return $input;
}

var_dump(slugger('Hello, World!')); // string(11) "hello-world"
?>
```

- They must be free-standing functions or `static` methods
- `__callStatic()` methods,
  or `[ClassName::class, 'methodName'](...)` is not supported

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 8.0: 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 8.5: Attributes on Constants

Since PHP 8.0, attributes can be set on:

- functions *(including closures and short closures)*
- classes *(including anonymous classes), interfaces, traits*
- class constants
- class properties
- class methods
- function/method parameters

PHP 8.5: Attributes on Constants

PHP 8.5 adds:

- non-class constants:

```
<?php
#[\MyAttribute]
const Example1 = 1;
?>
```

PHP 8.5: Attributes on Constants

PHP 8.5 adds:

- non-class constants:

```
<?php
#[\MyAttribute]
const Example1 = 1;
?>
```

Not allowed:

```
<?php
#[\MyAttribute]
const Example2 = 2,
    Example3 = 3;

#[\MyAttribute]
define('Example1', 1);
?>
```

PHP 8.5: Attributes on Constants

PHP 8.5 adds:

- non-class constants:

```
<?php
#[\MyAttribute]
const Example1 = 1;
?>
```

Not allowed:

```
<?php
#[\MyAttribute]
const Example2 = 2,
    Example3 = 3;

#[\MyAttribute]
define('Example1', 1);
?>
```

This also adds:

```
\Attribute::TARGET_CONSTANT
```

PHP 8.5: #[NoDiscard] Attribute

A new attribute for functions (`TARGET_FUNCTION`) and methods (`TARGET_METHOD`).

It enforces that the calling function consumes the return value.

```
<?php
$a = new DateTimeImmutable();
$a->setDate(2025, 5, 4);
?>
```

```
Warning: The return value of method DateTimeImmutable::setDate()
    should either be used or intentionally ignored by casting it as
    (void), as DateTimeImmutable::setDate() does not modify the
    object itself in /tmp/discard.php on line 3
```

PHP 8.5: #[NoDiscard] Attribute

A new attribute for functions (`TARGET_FUNCTION`) and methods (`TARGET_METHOD`).

It enforces that the calling function consumes the return value.

A `(void)` cast disables it:

```php
<?php
$a = new DateTimeImmutable();
(void) $a->setDate(2025, 5, 4);
?>
```

You can't use `#[Discard]` when:

- The function is typed with `: void` or `: never`
- For magic methods (including the constructor) that require `: void`, or have no return type at all
- For property hooks

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: 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
<?php
foreach (mb_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);
}
```

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 8.5: Levenstein

```php
<?php
$myName =         'Derick';
$notMyName =      'Derrick';
$alsoNotMyName = 'Dirk';

echo levenshtein($myName, $notMyName), "\n";
echo levenshtein($myName, $alsoNotMyName, 100, 10, 1), "\n";
?>
```

```
  1
012
||└ deletion
|└─ replacement
└── insertion

Derick
Dirk
```

PHP 8.5: Levenstein

```php
<?php
$flagUA = '🇺🇦';
$flagPS = '🇵🇸';

echo levenshtein($flagUA, $flagPS, 1, 10, 100), "\n";
?>
```

PHP 8.5: Levenstein

```php
<?php
$flagUA = '🇺🇦';
$flagPS = '🇵🇸';

echo levenshtein($flagUA, $flagPS, 1, 10, 100), "\n";
?>
```

```
20
```

```
020
||└ insertion
|└─ replacement
└── deletion
```

PHP 8.5: grapheme_levenshtein

```php
<?php
$flagUA = '🇺🇦';
$flagPS = '🇵🇸';

echo grapheme_levenshtein($flagUA, $flagPS, 1, 10, 100), "\n";
?>
```

```
10
```

```
010
||└ insertion
|└─ replacement
└── deletion
```

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`

Release Cycle Update

Before:

- 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

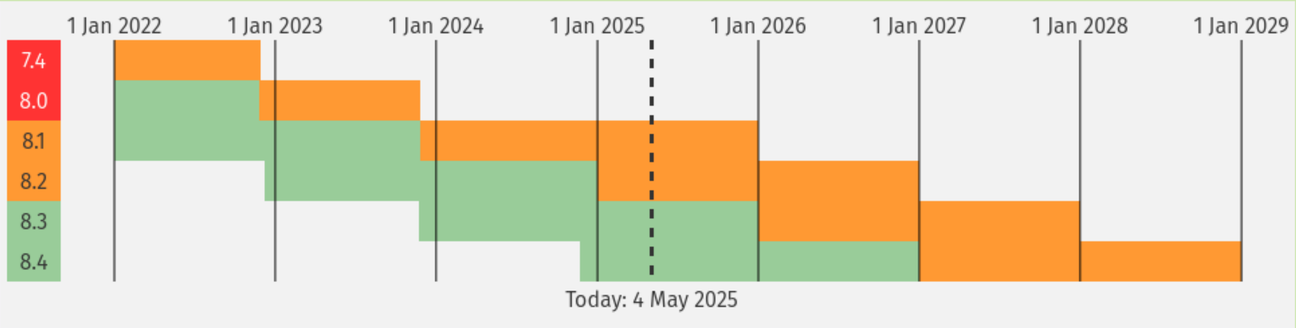
New:

- 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



Any Queries?

Véronique Debord-Lazaro — https://www.flickr.com/photos/debord/4932655275

Resources

Slides

`https://derickrethans.nl/talks/php-ipc25`

`https://xdebug.cloud`

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

Resources

Slides

`https://derickrethans.nl/talks/php-ipc25`

`https://xdebug.cloud`

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

Resources

Slides

`https://derickrethans.nl/talks/php-ipc25`

`https://xdebug.cloud`

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

Resources

Slides

`https://derickrethans.nl/talks/php-ipc25`

`https://xdebug.cloud`

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

Warning: Undefined array key 31 in **/home/httpd/pres2/show2.php** on line **215**

Notice: DOMDocument::load(): Read of 8192 bytes failed with errno=21 Is a directory in **/home/httpd/pres2/show2.php** on line **215**

Warning: DOMDocument::load(): Document is empty in /home/httpd/presentations, line: 1 in **/home/httpd/pres2/show2.php** on line **215**

Warning: Undefined array key 31 in **/home/httpd/pres2/show2.php** on line **216**

Fatal error: Uncaught ezcTemplateRuntimeException: The external (use) variable 'node' is not set in template: /home/httpd/pres2/presentations/templates/default/slide.ezt and called from the application code in /tmp/template-cache/compiled_templates/xhtml-updqr0/slide-5eb6f7fc995a21e41e7fdbd1e4869726.php:9 Stack trace: #0 /home/httpd/pres2/vendor/zetacomponents/template/src/compiled_code.php(188): include() #1 /home/httpd/pres2/vendor/zetacomponents/template/src/template.php(341): ezcTemplateCompiledCode->execute() #2 /home/httpd/pres2/show2.php(228): ezcTemplate->process('slide.ezt') #3 /home/httpd/pres2/show2.php(154): Presentation->display('31') #4 {main} thrown in **/tmp/template-cache/compiled_templates/xhtml-updqr0/slide-5eb6f7fc995a21e41e7fdbd1e4869726.php** on line **9**

Warning: Undefined array key 32 in **/home/httpd/pres2/show2.php** on line **215**

Notice: DOMDocument::load(): Read of 8192 bytes failed with errno=21 Is a directory in **/home/httpd/pres2/show2.php** on line **215**

Warning: DOMDocument::load(): Document is empty in /home/httpd/presentations, line: 1 in **/home/httpd/pres2/show2.php** on line **215**

Warning: Undefined array key 32 in **/home/httpd/pres2/show2.php** on line **216**

Fatal error: Uncaught ezcTemplateRuntimeException: The external (use) variable 'node' is not set in template: /home/httpd/pres2/presentations/templates/default/slide.ezt and called from the application code in /tmp/template-cache/compiled_templates/xhtml-updqr0/slide-5eb6f7fc995a21e41e7fdbd1e4869726.php:9 Stack trace: #0 /home/httpd/pres2/vendor/zetacomponents/template/src/compiled_code.php(188): include() #1 /home/httpd/pres2/vendor/zetacomponents/template/src/template.php(341): ezcTemplateCompiledCode->execute() #2 /home/httpd/pres2/show2.php(228): ezcTemplate->process('slide.ezt') #3 /home/httpd/pres2/show2.php(154): Presentation->display('32') #4 {main} thrown in **/tmp/template-cache/compiled_templates/xhtml-updqr0/slide-5eb6f7fc995a21e41e7fdbd1e4869726.php** on line **9**