INSTITUTO SUPERIOR TECNOLÓGICO DEL SUR

CARRERA

DISEÑO Y PROGRAMACIÓN WEB

LENGUAJE WEB III

“NOVEDADES PHP 8”

PROFESOR(A): Amado Cerpa Juan Andrés

ALUMNO: Vilca Apaza Christian

SEMESTRE : V

26/05/2021

INSTITUTO DEL SUR

INDICE

INDICE 2
PHP 3
Novedades de PHP 8 3
Compilador JIT 3
Attributes v2(rfc, rfc) 5
Union Types 6
Constructor Property Promotion 6
Mixed Type 7
Nuevas funciones para trabajar con strings 8
Stringable 9
Weak Maps 10

PHP

PHP es un lenguaje de programación de uso general que se adapta especialmente al desarrollo web. Fue creado inicialmente por el programador danés-canadiense Rasmus Lerdorf en 1994. En la actualidad, la implementación de referencia de PHP es producida por The PHP Group.

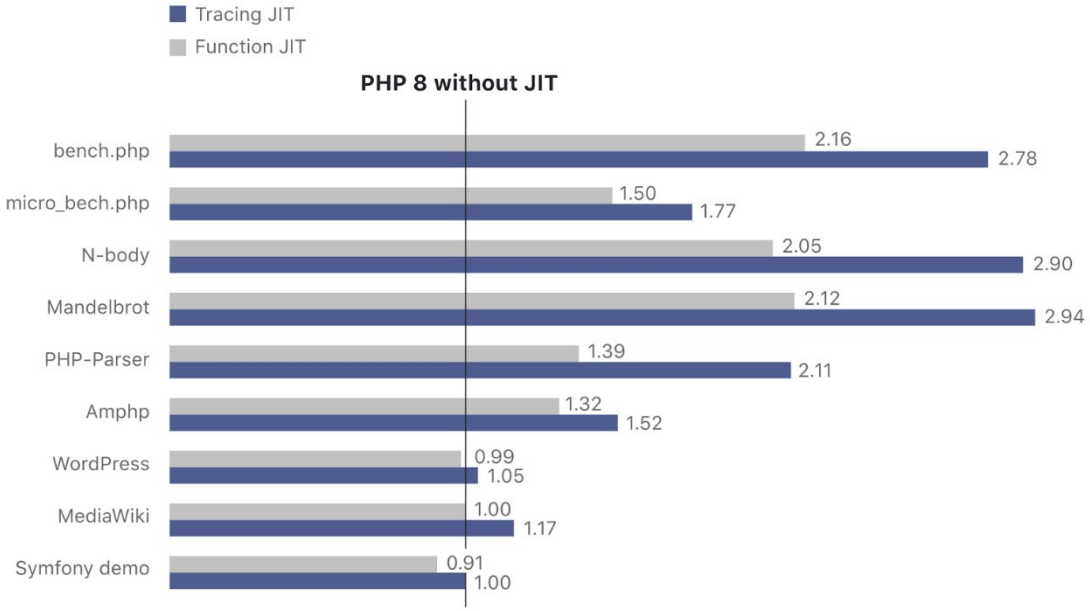
Novedades de PHP 8

Compilador JIT

La mayor novedad es el compilador JIT, que ha mejorado considerablemente el rendimiento. PHP no se compila, sino que se interpreta línea por línea. El compilador JIT (Just in Time) compila parte del código durante el tiempo de ejecución, por lo que funciona de manera muy similar a una versión en caché del código.
Esta característica de PHP 8 ya fue probada de manera muy exitosa por Pedro Escudero. Usó un script simple para comparar las versiones 5.3, 7.4 y 8 (con y sin JIT). Para ello, ejecutó el script cien veces en cada versión y calculó el tiempo promedio.
El resultado fueron los siguientes valores:

Versión	Tiempo en segundos
5.3	0,64574003219604
7.4	0,10253500938416
8 (sin JIT)	0,098223924636841
8 (con JIT)	0,053637981414795

Relative JIT contribution to PHP 8 performance



Attributes v2(rfc, rfc)

Una de las nuevas características de PHP 8 (que ha dado lugar a un gran debate en la comunidad PHP) son los atributos, conocidos también como "annotations" en muchos otros idiomas. Los atributos reemplazan la necesidad de almacenar metadatos con docblocks en PHP 8. Antes había que recurrir a esto para declarar los metadatos de las clases, métodos, funciones y argumentos de forma estructurada.

Como puedes imaginar, utilizar los comentarios del código para aplicar los metadatos no era lo ideal, pero funcionaba. Afortunadamente, ahora no tendremos ese problema.

Los atributos pueden anuaciarse introduciendo Syntax #[...] .

A continuación, se presentan algunos ejemplos de la RFC sobre cómo se pueden aplicar los atributos a diferentes tipos de datos.

```php
#[ExampleAttribute]
class Foo
{
    #[ExampleAttribute]
    public const FOO = 'foo';

    #[ExampleAttribute]
    public $x;

    #[ExampleAttribute]
    public function foo(#[ExampleAttribute] $bar) { }
}

$object = new #[ExampleAttribute] class () { };

#[ExampleAttribute]
function f1() { }

$f2 = #[ExampleAttribute] function () { };

$f3 = #[ExampleAttribute] fn () => 1;
```

Union Types

Una de las características más demandadas desde que comenzamos a poder "tipar" los argumentos de funciones y métodos era la posibilidad de especificar varios tipos para un argumento y el retorno de una función.
Los "union types" son una colección de dos o más tipos de modo que indiquemos que cualquiera de ellos es válido cuando especifiquemos el valor de ese argumento o el tipo de retorno de la función.

```php
public function unionTypesFunction(ClassA|ClassB $foo): int|string;
```

Constructor Property Promotion

PHP 8 también trae lo que se conocen como "azucarillos sintáticos" que nos permiten simplificar el código que escribimos por versiones más cortas.
Uno de estos nuevos "sintactic sugars" es el "Constructor Property Promotion" que nos permite pasar de construir nuestros DTO's o "value objects" de esta forma:

```php
class User {
    public string $name;
    public string $email;
    public int $age;

    public function __construct(
        string $name = '',
        string $email = '',
        int $age = 0,
    ) {
        $this->name = $name;
        $this->email = $email;
        $this->age = $age;
    }
}
```

A esta mucho más reducida:

```php
class User {
    public function __construct(
        public string $name = '',
        public string $email = '',
        public int $age = 0
    ) {}
}
```

Un gran cambio y que nos permite simplificar muchísimo la creación de este tipo de objetos además de poner de relieve el interés del equipo de PHP en crear un lenguaje agradable y moderno.

Mixed Type

Seguro que alguna vez habéis tratado de tipar un argumento o el valor de retorno de una función mediante mixed pensando que era un tipo válido y, sin embargo, habéis recibido un error al hacerlo. Por ejemplo:
una función que devuelve void o null .
un argumento que puede recibir valores de varios tipos.
Con PHP 8 ya podremos tipar mediante mixed argumentos, propiedades y valores de retorno para representar varios de estos valores:
array
bool
callable
int
float
null
object
resource
string
Sí, como veis mixed también incluye null por lo que no será válido un "tipado" como el siguiente: ?mixed .
::class en objetos
Otro "sintactic sugar" que nos trae PHP8 es la posibilidad de emplear ::class sobre objetos y no sólo sobre el FQDN de una clase, de modo que ya no sea necesario emplear la función get_class :

```php
class User {
    public function __construct(
        public string $name = '',
        public string $email = '',
        public int $age = 0
    ) {}
}

$user = new User('Gerardo', 'info@mail.com', 34);

print_r($user::class);
```

Nuevas funciones para trabajar con strings

PHP 8 trae también una serie de nuevas funciones para simplificar la forma en que trabajamos con el tipo string .
Seguramente la más interesante de ellas sea str_contains , la cual ¡por fin! nos soluciona tener que emplear la función strpos para comprobar si un string contiene a otro.
Es decir, se acabó hacer esto:

```php
if (strpos('En un lugar de la Mancha de cuyo ... ', 'Mancha') !== false) {
  /* … */
}
```

Ya que gracias a str_contains quedará mucho más simplificado y legible:

```php
if (str_contains('En un lugar de la Mancha de cuyo ... ', 'Mancha')) {
  /* … */
}
```

Además de esta función, PHP 8 incorpora dos más:
str_starts_with para comprobar si un string comienza con un determinado string.
str_ends_with para comprobar si un string termina con un determinado string.

```php
if (str_starts_with('En un lugar de la Mancha de cuyo ... ', 'En un lugar')) {
  /* … */
}

if (str_end_with('En un lugar de la Mancha de cuyo ... ', 'de cuyo  ... ')) {
  /* … */
}
```

Stringable

PHP 8 añade la interfaz Stringable para que podamos tipar aquellos argumentos o valores de retorno que son o bien un string o bien implementan el método __toString() .
Además, a partir de ahora cualquier clase que declare el método __toString() implementará automáticamente esta interfaz sin necesidad de que lo especifiquemos a mano.

```php
class User {
    public function __construct(
        public string $name = '',
        public string $email = '',
        public int $age = 0
    ) {}

    public function __toString(): string
    {
        return $this->name;
    }
}

$user = new User('Gerardo', 'info@mail.com', 34);


function printMessage(Stringable $stringable) {
    /* … */
}

printMessage(new User());
printMessage('abc');
```

Catch sin especificar la variable que recoge la excepción
Otra mejora que trae PHP 8 es la posibilidad de especificar bloques try / catch sin que el catch tenga que declarar la variable que recoge la excepción, ya que a veces no se emplea para nada, añadiendo ruido visual al código o incluso siendo señalado como warning por el editor:

```php
try {
    // an exception is thrown
} catch (ApiException) {
    // do something but not with the exception
}
```

Weak Maps

Finalmente, PHP 8 introduce los WeakMaps , basados en las "Weak References" de PHP 7.4 sobre las cuales podéis leer en la propia documentación de PHP:

Los WeakMaps nos permiten almacenar referencias a objetos pero sin impedir que estos sean eliminados por el "garbage collector". Seguramente esto sea un añadido muy interesante para librerías encargadas de cachear objetos como por ejemplo hace Doctrine con las entidades que recupera y que, almacenándolas en WeakMaps permitirían liberar memoria de forma más eficiente cuando esas entidades dejan de tener referencias en algún momento. Un ejemplo de su uso lo podemos ver en la RFC de los WeakMap :

```php
class FooBar {
    private WeakMap $cache;

    public function getSomethingWithCaching(object $obj) {
        return $this->cache[$obj] ??= $this->computeSomethingExpensive($obj);
    }

    // ...
}
```

Conclusión

Php ha estado evolucionando en los últimos años de manera asombrosa desde su gran salto de php 5 a php 7 se ve un crecimiento y performance muy grande tanto así que seria uno de los lenguajes mas demandados y de moda en estos tiempos, sus ultimas adiciones en la programación orientada a objetos y el manejo y la velocidad de procesamiento han superado expectativas de muchos en lo general a mi me gusta mucho su forma de trabajar con la programación orientada a objetos en conclusión PHP seguirá mejorando y viviendo por mucho tiempo.

Bibliografía

https://es.stackoverflow.com/questions/409247/cuales-son-las-novedades-en-php-8
https://raidboxes.io/es/blog/webdesign-development/php-8/
https://latteandcode.medium.com/las-principales-novedades-de-php-8-3396697b65bd
https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/php-8/
https://kinsta.com/es/blog/php-8/
https://victorroblesweb.es/2021/02/09/php-8-novedades-y-mejoras/