Nek

Zeste de savoir

Sortie de PHP 8

6 décembre 2020

Table des matières

Tous les ans, les développeurs de PHP travaillent d'arrache pied pour nous sortir une nouvelle version de PHP en fin d'année. Et à chaque fois c'est beaucoup de nouvelles fonctionnalités très intéressantes qui sont intégrées au langage. Découvrons ensemble les nouvelles fonctionnalités de cette nouvelle version majeure.

1. Une nouvelle année de PHP

Tous les ans les équipes de PHP nous préparent une nouvelle version de PHP. Depuis quelques années PHP arbore une version 7. Cette dernière a marqué une nouvelle ère de performances car elle a intégré PHP NG, une modification profonde de PHP.

Ce travail était à la base réalisé dans l'idée de changer la structure de PHP pour y intégrer du JIT (Just In Time). Mais le résultat a été si performant et le travail était encore si long pour arriver à une version JIT que les équipes ont décidé de la sortir quand même.

Mais qu'à cela ne tienne, le travail acharné paie : et PHP 8.0 est cette version qui nous amène le JIT, mais pas que! La communauté a été très active sur PHP cette dernière année et beaucoup de nouvelles fonctionnalités sont arrivées en même temps.

Détaillons ensemble les nouvelles fonctionnalités majeures du langage.

2. Nouvelles fonctionnalités

2.1. Changements dans les casts de nombres

Commençons ce tour des nouveautés en douceur 🍊 .

Lorsque vous passez une chaîne de caractères en paramètre d'une fonction qui attend un nombre, PHP tente de tout convertir en nombre. En PHP 7, si cette chaîne n'était pas convertible, la fonction recevait 0.

Avec PHP 8 si la chaîne de caractère est effectivement un nombre, elle est convertie, sinon vous obtiendrez une exception.

```
function foo(int $i) {

}

foo('2 et autre chose'); // throw TypeError
foo('2'); // fonctionne toujours si vous n'avez pas spécifié le
    strict types
```

2.2. Les unions de types

Clairement l'une des nouvelles fonctionnalités qui me plaît le plus! Dans les dernières versions de PHP nous avons vu arriver le typage strict, mais lorsqu'on prend plusieurs types potentiels en paramètres il nous faut toujours re-vérifier le type car on ne peut pas réclamer cela à PHP. Dans cette nouvelle version de PHP on pourra donc faire des unions de type et s'éviter cette tâche.

Voici un exemple d'interface fonctionnant avec l'union de types:

```
interface ContainsNumberInterface
{
    public function setNumber(int|float $number);
    public function getNumber(): int|float;
}
```

2.3. Le type mixed

C'est un nouveau type qui vient s'ajouter aux types natifs existant dans PHP. Il a fait longtemps débat car d'un côté inutile, puisqu'il n'oblige à rien lors de l'utilisation d'une fonction ou d'un attribut. Mais PHP 8 rajoute ce type. Voyez là l'obligation de tout typer explicitement 🍊 .

```
class Something
{
    public mixed $userData;
}
```

Note : il a été décidé d'utiliser mixed et non pas `any` comme on peut voir dans d'autres langages car la communauté PHP entière s'accorde à utiliser ce wording pour spécifier un "multi-type" en documentation.

2.4. Les expressions match

Plus rapide à écrire, mais aussi plus rapide à exécuter qu'un switch, les expressions match ressemblent tout de même grandement à un switch:

— Elles retournent directement quelque chose
— Elles font une comparaison stricte contrairement au switch

En voici un exemple illustratif:

```
// En utilisant un switch
switch(8.0) {
  case "8.0":
    $result = "Oh non !";
    break;
  case 8.0:
    $result = "Ce que j'espère";
    break;
}
echo $result; // Affiche "Oh non !"

echo match (8.0) {
    "8.0" => "Oh non !",
    8.0 => "Ce que j'espère",
}; // Affiche "Ce que j'espère"
```

2.5. Les arguments nommés

Certaines fonctions ont une longue liste d'arguments optionnels, et devoir spécifier les arguments intermédiaires n'est pas toujours utile ou peut même porter à confusion! PHP 8 nous propose donc cette nouvelle fonctionnalité: les arguments nommés.

Voici un exemple:

```php
<?php
$items = [1, 2, 'foo'];
array_filter(array: $items, callback: function ($item) { return
    is_int($item); });
```

2.6. Les attributs

A partir de maintenant on s'accordera pour dire "propriété" de classe et "attribut" pour ce qui va suivre d'accord? 🍊

En PHP on a l'habitude d'avoir des commentaires contenant des annotations. Heureusement l'API de réflexion de PHP nous permet de simplifier le processus de parsing des annotations en commentaires... Mais ça n'était pas assez au goût général, et ça fait d'ailleurs depuis PHP 5.4 (environ) que les discussions sont ouvertes au sujet de faire quelque chose de "plus intégré à PHP". (pour les annotations Symfony et la plupart des frameworks utilisent un package nommé `doctrine/annotations`)

Nous y voici donc, on peut à présent utiliser les **attributs** pour remplacer les annotations. Voyons comment on peut déclarer un attribut.

```php
#[Attribute]
class Column
{
    public string $name;

    public function __construct(string $name = null)
    {
        $this->name = $name;
    }
}
```

Voici un exemple d'utilisation de notre attribut. (Bientôt nos entités doctrine ressembleront à cela !)

```php
final class User
{
    #[ORM\Id()]
    #[ORM\Column("id")]
    private int $id;
}
```

Et pour finir on utilise l'API de réflexion de PHP :

```
$reflectionClass = new ReflectionClass(User::class);
$attributes = $reflectionClass->getMethods()[0]-
    >getAttributes(Column::class); //
```

Note: Les attributs doivent être supportées par votre librairie, PHP ne fera pas la conversion automatiquement entre annotation et attributs.

2.7. Nouveautés sur le throw

Dans un premier temps, la clause throw est devenue une expression. C'est tout bête, mais ça permet par exemple de rendre le code suivant valide :

```
$value = $nullableValue ?? throw new InvalidArgumentException();
```

Et une petite modification a été faite sur le catch : le nom de variable est maintenant optionnel si vous n'avez pas besoin de cette dernière.

```
try {
    throw new TypeError();
} catch (TypeError) {
    echo 'oups erreur de type';
}
```

2.8. L'opérateur « Null Safe »

C'est l'une des fonctionnalités qui m'attire le plus : elle permet de sortir de ce qu'on appelle communément le «null hell ».

Concrètement, vous allez pouvoir remplacer ce code :

```
$country =  null;

if ($session !== null) {
    $user = $session->user;

    if ($user !== null) {
        $address = $user->getAddress();

        if ($address !== null) {
```

```
            $country = $address->country;
        }
    }
}
```

Par celui-ci :

```
$country = $session?->user?->getAddress()?->country;
```

N'oubliez cependant pas que moins vous aurez de `null`, mieux vous vous porterez psychologiquement!

2.9. Promotion des propriétés du constructeur

Cette fonctionnalité est en réalité une nouvelle façon de déclarer les propriétés de classe : PHP nous permet de les déclarer directement dans le constructeur, cela dans le but d'éviter le code redondant que l'on voit habituellement dans les classes.

Voici un exemple avant la nouvelle fonctionnalité:

```
class User
{
    private string $name;
    private string $email;

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

Et après:

```
class User
{
    public function __construct(
        private string $name,
        private string $email
    ) {}
}
```

Plutôt cool non?

3. PHP 8 is JIT

3.1. PHP et le JIT, une longue histoire

En réalité ça fait plusieurs années que l'équipe de PHP, et plus particulièrement Dmitry Stogov qui s'est acharné à faire fonctionner PHP en JIT. Et ça n'a pas été sans problème. Le tout a été recodé plusieurs fois, l'une d'entre elles a d'ailleurs donné naissance à PHP 7! Si vous avez entendu parler de php ng, il s'agit en réalité d'une tentative de clean pour faire fonctionner PHP en JIT. Mais l'avancée a été telle qu'on a pu sortir une nouvelle version qui boost les perfs sans même passer en JIT.

Bref, nous y sommes. PHP 8 est JIT (Just In Time).

3.2. JIT, donc perf, n'est-ce pas ?

Et bien non. C'est une nouveauté qui a donné beaucoup de fil à retordre à l'équipe de PHP mais qui n'a qu'assez peu d'impact sur les performances de PHP au final. Par exemple si vous avez une API qui répond en 10ms, et bien il y a peu de chances que PHP8 vous aide : php a toujours été très optimisé pour cela, et PHP 7.4 avec le preloading a encore poussé la vitesse de chargement de PHP. Pas vraiment d'amélioration , mais PHP est déjà très rapide.

Là où vous aurez les meilleurs gains de performances c'est sur les process longs. Par exemple si vous voulez faire des traitements mathématiques, ou du machine learning. Bref, des processus longs. En voici d'ailleurs la preuve.

Élément externe (Video) —
Consultez cet élément à l'adresse `https://www.youtube.com/embed/dWH65pmnsrI?feature=oembed`.
Démonstration des gains de performances sur des calculs mathématiques longs

---

Cependant, il faut reconnaître qu'il y a une légère amélioration de performances tout de même dans le cas général.

3.3. Fun fact

Le typage généralisé arrivé ces dernières années avait un coût en terme de perf, car ça ajoutait un check supplémentaire nécessaire. Alors qu'avec PHP 8 et JIT, c'est une optimisation.

Ne cherchez pas à typer pour la perf : c'est marginal. Mais c'est rigolo.

Une belle nouvelle version de PHP, malgré la petite taille de la core team de PHP, ces derniers nous sortent encore de belles nouvelles version, et c'est agréable!

3. PHP 8 is JIT

Avant de vous laisser j'aimerais vous glisser un dernier mot: récemment nous avons également eu droit à une nouvelle version de composer . Le gestionnaire de paquet de PHP est à présent plus rapide, et il utilise moins de mémoire.

Liste des abréviations

JIT Just In Time : compilation à la volée. 1, 7, 8