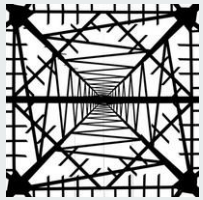
PHP 7.4
Nouveautés

Julien Vinber - Meetup AFUP Montpellier
28 janvier 2020 - Crealead

Julien Vinber



- Lead Dev chez Qape
- Dev PHP/Symfony
- Dev depuis plus de 15 ans
- Coordinateur AFUP Montpellier

Slack Mth : @julienvinber
LinkedIn : @julienvinber

Rappel

Currently Supported Versions

Branch	Initial Release		Active Support Until		Security Support Until	
7.2	30 Nov 2017	*2 years, 1 month ago*	30 Nov 2019	*1 month ago*	30 Nov 2020	*in 10 months*
7.3	6 Dec 2018	*1 year, 1 month ago*	6 Dec 2020	*in 10 months*	6 Dec 2021	*in 1 year, 10 months*
7.4	28 Nov 2019	*1 month ago*	28 Nov 2021	*in 1 year, 10 months*	28 Nov 2022	*in 2 years, 10 months*

Or, visualised as a calendar:

Key

Active support	A release that is being actively supported. Reported bugs and security issues are fixed and regular point releases are made.
Security fixes only	A release that is supported for critical security issues only. Releases are only made on an as-needed basis.
End of life	A release that is no longer supported. Users of this release should upgrade as soon as possible, as they may be exposed to unpatched security vulnerabilities.

Modification structurelle.

Typed properties

```php
<?php

class Utilisateur73
{
    /**
     * @var int
     */
    public $id;

    /**
     * @var string
     */
    public $nom;

    /**
     * @var string|null
     */
    public $prenom;
}
```

```php
<?php

class Utilisateur74
{
    public int $id;
    public string $nom;
    public ?string $prenom;
}
```

https://wiki.php.net/rfc/typed_properties_v2

Type de retour covariant

```php
interface CollectionInterface
{
    public function premier(): ItemInterface;
}

class CollectionUtilisateur implements CollectionInterface
{
    public function premier(): Utilisateur
    {   }
}
```

```php
class Item
{
    public string $nom;
}

class Utilisateur extends Item
{}
```

https://wiki.php.net/rfc/covariant-returns-and-contravariant-parameters

Type de paramètre contravariant

```php
interface CollectionInterface
{
    public function ajouter(Item $item): void;
}
```

```php
class CollectionUtilisateur implements CollectionInterface
{
    public function ajouter(Utilisateur $utilisateur): void;
    {    }
}
```

```php
class Item
{
    public string $nom;
}
```

```php
class Utilisateur extends Item
{}
```

https://wiki.php.net/rfc/covariant-returns-and-contravariant-parameters

```php
class CollectionParent
{
   public function ajouter(Utilisateur $utilisateur):void
   {print($utilisateur);}
}

class CollectionEnfant extends CollectionParent
{
   public function ajouter(Item $utilisateur):void
   {print($utilisateur);}
}
```

```php
class Item
{
   public string $nom;
}

class Utilisateur extends Item
{}
```

Nouvelle solution pour sérialiser un objet

https://wiki.php.net/rfc/custom_object_serialization

Serializable

```php
class A implements Serializable {
    private $prop;
    public function serialize() {
        return serialize($this->prop);
    }
    public function unserialize($payload) {
        $this->prop = unserialize($payload);
    }
}
class B extends A {
    private $prop;
    public function serialize() {
        return serialize([$this->prop, parent::serialize()])
    }
    public function unserialize($payload) {
        [$prop, $parent] = unserialize($payload);
        parent::unserialize($parent);
        $this->prop = $prop;
    }
}
```

__sleep / __wakeup

```php
class A implements Serializable {
    private $ttc;
    private $ht;
    public function __sleep() {
        return ['ht'];
    }
    public function __wakeup() {
        $this->ttc = $ht * 1.2;
    }
}
```

Nouvelle solution :

```php
class A {
    private $prop_a;
    public function __serialize(): array {
        return ["prop_a" => $this->prop_a];
    }
    public function __unserialize(array $data) {
        $this->prop_a = $data["prop_a"];
    }
}
class B extends A {
    private $prop_b;
    public function __serialize(): array {
        return [
            "prop_b" => $this->prop_b,
            "parent_data" => parent::__serialize(),
        ];
    }
    public function __unserialize(array $data) {
        parent::__unserialize($data["parent_data"]);
        $this->prop_b = $data["prop_b"];
    }
}
```

Weak References

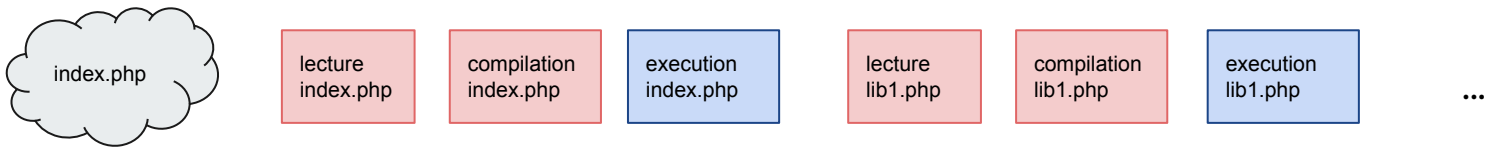
```php
$object = new stdClass;
$weakRef = WeakReference::create($object);

var_dump($weakRef->get());  //object(stdClass)#1 (0) {}
unset($object);
var_dump($weakRef->get()); //null
```

https://wiki.php.net/rfc/weakrefs

Les performances

Preloading

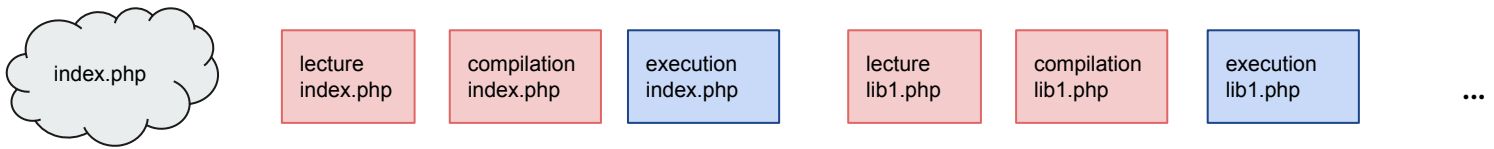
La version simple



https://wiki.php.net/rfc/preload

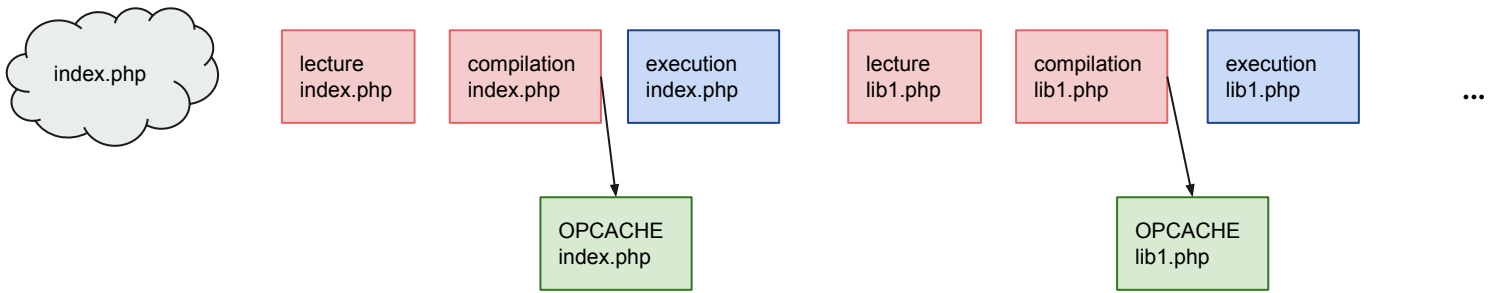
Preloading

Avec OPCache



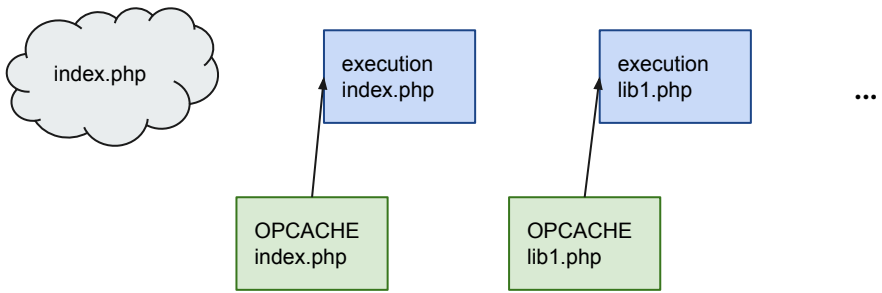
Preloading

Avec OPCache : premier fois



Preloading

Avec OPCache : ensuite



Preloading

C’est mieux, mais il reste la première fois.

Preloading

Nouveau avec PHP 7.4 le preloading :

; php.ini
```
opcache.preload=/path/monfichier.php
```

Preloading

Exemple avec SF5/SF4.4

; php.ini

```
opcache.preload=/path/to/project/var/cache/prod/App_KernelProdContainer.preload.php
```

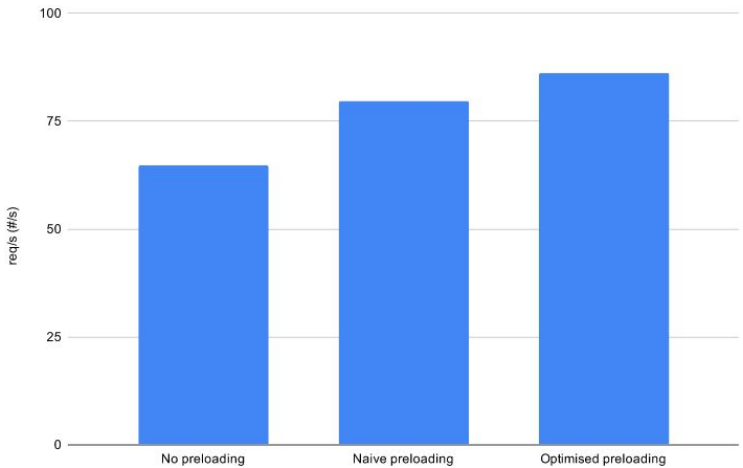
```php
/**
* Get files that specified suffix
* @param $path
* @param array $files
* @return array
*/
function getfiles( $path , &$files = array() ) {
   if ( !is_dir( $path ) ) return null;
   $handle = opendir( $path );
   while ( false !== ( $file = readdir( $handle ) ) ) {
       if ( $file != '.' && $file != '..' ) {
           $path2 = $path . '/' . $file;
           if ( is_dir( $path2 ) ) {
               getfiles( $path2 , $files );
           } else {
               if ( preg_match( "/\.(php|php5)$/i" , $file ) ) {
                   $files[] = $path2;
               }
           }
       }
   }
   return $files;
}
$files = getfiles('/png/www/example.com/public_html/app/wordpress');
$br = (php_sapi_name() == "cli") ? "\n" : "<br />";
foreach($files as $file){
   opcache_compile_file($file);
   echo $file.$br;
}
```

Preloading

ATTENTION : à réserver pour la prod

MAJ => recharger PHP

En chiffres



https://stitcher.io/blog/php-preload-benchmarks

Un peu de sucre...

...

Existe depuis la 5.6

Permet de “packager” des arguments

```php
function test($arg1, $arg2, $arg3 = null) {
   var_dump($arg1, $arg2, $arg3);
}

test(...[1, 2]);       // 1, 2
test(...[1, 2, 3]);    // 1, 2, 3
test(...[1, 2, 3, 4]); // 1, 2, 3 (remaining arg is not captured by the function declaration)
```

https://wiki.php.net/rfc/spread_operator_for_array

...

En PHP 7.4

Étendu sur les Array

```php
$parts = ['apple', 'pear'];

$fruits = ['banana', 'orange', ...$parts, 'watermelon'];

var_dump($fruits);
```

...

La source peut être un tableau, un Traversable ou générateur.

```php
function generator() {
    for ($i = 3; $i <= 5; $i++) {
        yield $i;
    }
}
$arr1 = [0, 1, 2, ...generator()];
```

Fonctions anonymes courts

Les fonctions anonymes sont arrivées en 5.3

```php
function array_values_from_keys($arr, $keys) {
   return array_map(function ($x) use ($arr) { return $arr[$x]; }, $keys);
}
```

En 7.4

```php
function array_values_from_keys($arr, $keys) {
   return array_map(fn($x) => $arr[$x], $keys);
}
```

https://wiki.php.net/rfc/arrow_functions_v2

Assignation Coalesce Null

Depuis 7.0

```php
$username = $_GET['user'] ?? 'nobody';
```

En 7.4

```php
$valeur ??= 'nada';
```

https://wiki.php.net/rfc/null_coalesce_equal_operator

Séparateur neutre pour les nombres.

```php
$nombre = 1000000000;

$nombre = 1_000_000_000;
```

Attention : à utiliser uniquement entre 2 chiffres “100_” n’est pas autorisé.

https://wiki.php.net/rfc/numeric_literal_separator

FFI

FFI - Foreign Function Interface

Permet de s'interfacer directement en PHP à des bibliothèques externes (.so, .dll)

Sans extension PHP à écrite dans un Autre langage et à installer spécifiquement sur le serveur.

https://wiki.php.net/rfc/ffi

Comment cela marche?

- On prend une librairie existante ou nous créons notre propre librairie.
- On récupère ou on crée le fichiers .h de cette librairie
- On charge la librairie.
- On peut l'utiliser.

Un exemple ~~volé~~ emprunté chez jolicode

Utilisation de la librairie libuuid

https://jolicode.com/blog/php-7-4-et-ffi-ce-quil-faut-retenir

Le fichier .h origine.

```
# ...
# Quelques déclarations de constantes :
#define UUID_VARIANT_NCS    0
#define UUID_VARIANT_DCE    1
#define UUID_VARIANT_MICROSOFT  2
#define UUID_VARIANT_OTHER  3

# Quelques declarations de fonctions :
void uuid_generate(uuid_t out);
int uuid_compare(const uuid_t uu1, const uuid_t uu2);
# ...
```

Le fichier .h nétoyer

```
#define FFI_LIB "libuuid.so.1"

typedef unsigned char uuid_t[16];

extern void uuid_generate_time(uuid_t out); // v1
extern void uuid_generate_md5(uuid_t out, const uuid_t ns, const char *name, size_t len); // v3
extern void uuid_generate_random(uuid_t out); // v4
extern void uuid_generate_sha1(uuid_t out, const uuid_t ns, const char *name, size_t len); // v5
```

Utilisation en PHP

```php
<?php

$ffi = FFI::load(__DIR__ . '/include/uuid-php.h');

$output = $ffi->new('uuid_t');

$ffi->uuid_generate_random($output);

$uuid = sprintf('%02x%02x%02x%02x-%02x%02x-%02x%02x-%02x%02x-%02x%02x%02x%02x%02x%02x', ...$output);
```

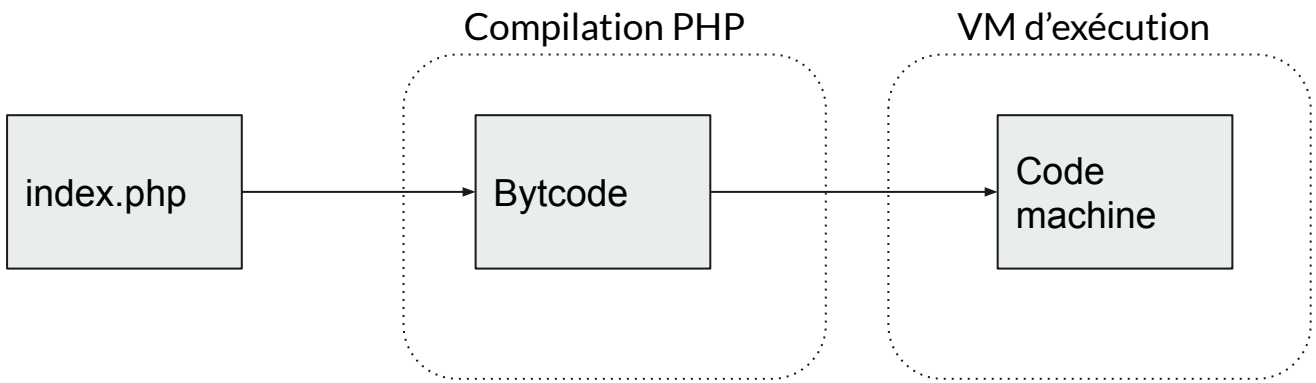
PHP 8

Union types

```php
class Number
{
    private int|float $number;

    public function setNumber(int|float $number): void {
        $this->number = $number;
    }

    public
        function getNumber(): int|float {
        return $this->number;
    }
}
```

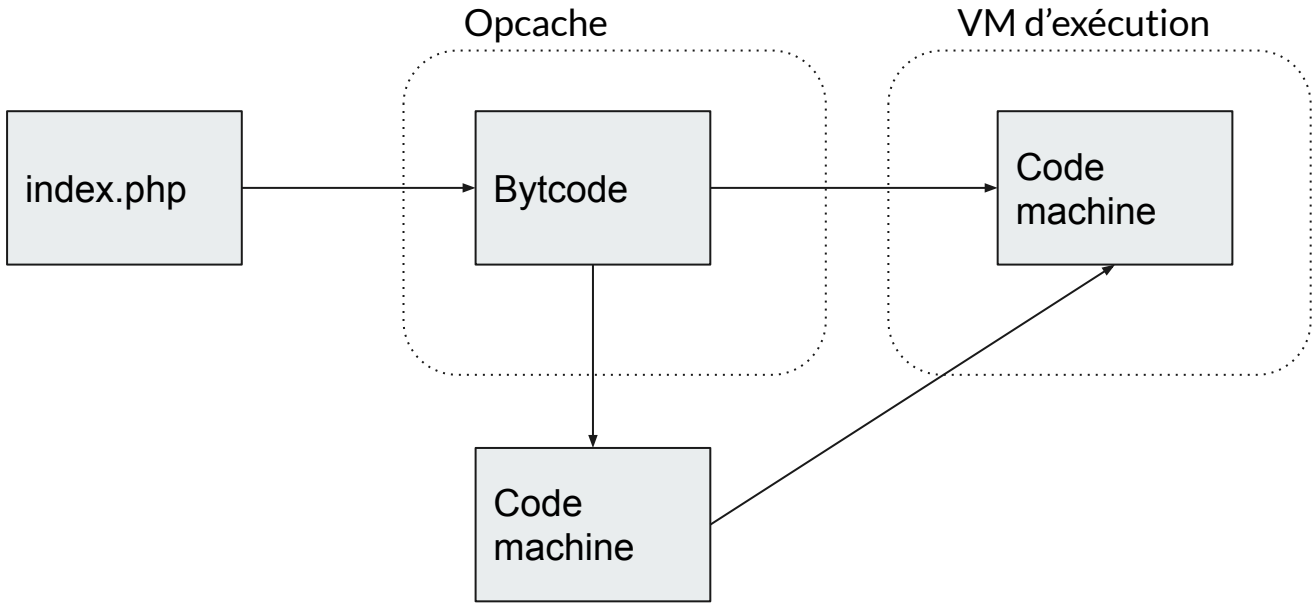
https://wiki.php.net/rfc/union_types_v2

Fonctionnement d’un compilateur comme PHP



https://wiki.php.net/rfc/jit

Avec JIT


Opcache
VM d'exécution
index.php
Bytcode
Code machine
Code machine

Priorité de concaténation

```php
echo "sum: " . $a + $b;
```

???

https://wiki.php.net/rfc/concatenation_precedence

Priorité de concaténation

```php
echo "sum: " . $a + $b;

echo $a + $b . " €";
```

Aujourd'hui :

- "." et "+" au même niveau
- évaluation de gauche à droite)

Priorité de concaténation

```php
echo "sum: " . $a + $b;
```

=> Aujourd'hui

```php
echo ("sum: " . $a) + $b;
```

=>

Warning

```php
echo $a + $b . " €";
```

=> Aujourd'hui

```php
echo ($a + $b) . " €";
```

=>

OK

Priorité de concaténation

```php
echo "sum: " . $a + $b;

echo $a + $b . " €";
```

PHP 8.0 :

- “+” à une priorité supérieur à “.”

Priorité de concaténation

```php
echo "sum: " . $a + $b;
```

=> Aujourd'hui

```php
echo "sum: " . ($a + $b);
```

=>

OK

```php
echo $a + $b . " €";
```

=> Aujourd'hui

```php
echo ($a + $b) . " €";
```

=>

OK

Quelques liens

RFC :

https://wiki.php.net/rfc#php_74

Changelog :

https://www.php.net/manual/fr/doc.changelog.php

Github :

https://github.com/php/php-src/blob/PHP-7.4/UPGRADING

Merci

Des questions ?