

الجمهورية اليمنية

الجامعة الوطنية

كلية العلوم والهندسة

PHP 8

عمل

م. مودة الصلوي

إشراف

د. إبراهيم الشامي

**PHP 8.0 #: -**

PHP 8.0 هو تحديث إصدار رئيسي وإنجاز ملحوظ في PHP، لأنه يجلب العديد من الميزات الجديدة لنظام النوع، والنحو، ومعالجة الأخطاء، والسلاسل، والبرمجة الموجهة للكائنات، والمزيد.

PHP جهود مئات الأشخاص الذين اجتمعوا معاً لتشكيل مستقبل لغة البرمجة التي تدعم جزءاً كبيراً من مواقع الويب والتطبيقات على الإنترنت ، يحاول PHP أن يكون محافظاً مع التغييرات التي يمكن أن تؤدي إلى تعطل غالبية التطبيقات، ومع ذلك، فإنه يجلب العديد من الميزات الرئيسية الجديدة إلى PHP 8.0.

تؤدي الميزات مثل المعلمات المسماة ، و JIT ، والسمات ، وخصائص المنشئ إلى تحسينات كبيرة وتغييرات في بناء الجملة، في حين تساعد العديد من التحسينات الطفيفة مثل resource ترحيل الكائنات ، وتحسين معالجة الأخطاء، والتغييرات والتحسينات في المشغلات ومقارنات المحرك في تقليل فرص تجاهل الأخطاء بسلاسة.

# الميزات الجديدة الرئيسية (Major New Features): -

• وظائف وفئات جديدة (New Functions and Classes)

• تحسينات نظام النوع (Type System Improvements)

• تحسينات معالجة الأخطاء (Error Handling Improvements)

• هجرة الموارد إلى الكائنات (Resource to Object Migration)

• تغييرات البرمجة الموجهة للكائنات في PHP (PHP Object Oriented Programming Changes)

• التغييرات المتعلقة بالسلسلة (String – related changes)

# الميزات الجديدة الرئيسية: -

**\_ المعلمات المسماة: -**

يسمح PHP 8.0 بالمعلمات المسماة في استدعاءات الوظيفة/ الطريقة بالإضافة إلى المعلمات الموضعية التقليدية.

str\_contains(needle: 'Bar', haystack: 'Foobar');

يؤدي هذا إلى جعل أسماء معلمات الوظيفة/ الطريقة جزءًا من واجهة برمجة التطبيقات العامة.

يوضح DocBlock غير الموحد @no-named-arguments أن المكتبة لا توفر التوافق مع الإصدارات السابقة للمعلمات المسماة.

\_ صفات: -

تتيح السمات إعلان البيانات الوصفية للوظائف والفئات والخصائص والمعلمات.

يتم تعيين السمات إلى أسماء فئات PHP (المعلنة بسمة نفسها)، ويمكن جلبها برمجيًا باستخدام واجهة برمجة تطبيقات PHP Reflection.

[CustomAttribute] #

class Foo {

#[AnotherAttribute(42)]

public function bar(): void{} }

تجعل السمات عملية الإعلان عن السمات/التعليقات التوضيحية سهلة وفعالة، بعد أن كانت تتطلب في السابق تخزينها في تعليقات DocBlock، وتحليل السلسلة لاستنتاجها.

**\_ خصائص المنشئ: -**

بناء جملة جديد لإعلان خصائص الفئة مباشرة من منشئ الفئة (  \_\_constructالطريقة السحرية).

class User

{ public function \_\_construct(private string $name){} }

في المنشئ، يدعم PHP 8.0إعلان خاصية الرؤية والنوع ( public، privateأو protected)  سيتم تسجيل هذه الخصائص كخصائص فئة بنفس خاصية الرؤية والنوع المعلن عنها في المنشئ.

يمكن أن تساعد هذه الميزة غير المتوافقة مع الإصدارات السابقة في تقليل الكود النمطي عند إعلان فئات كائن القيمة.

**\_ يدعم PHP Opcache JIT : -**

وهو معطل افتراضياً، وإذا تم تمكينه، يقوم JIT بتجميع التعليمات الأصلية وتخزينها مؤقتاً.

ولا يحدث فرقاً ملحوظاً في تطبيقات الويب المرتبطة بالإدخال/الإخراج، ولكنه يوفر تعزيزاً للأداء للتطبيقات التي تعتمد على وحدة المعالجة المركزية بشكل كبير.

# Enabling JIT in php.

ini opcache.enable=1

opcache.jit\_buffer\_size=100M

opcache.jit=tracing

لاحظ أن JIT لا يزال جديداً، وتم إصلاح الأخطاء فيه قبل يوم واحد فقط من إصدار PHP8.0.0، وهو يجعل تصحيح الأخطاء وإنشاء ملفات التعريف أكثر صعوبة مع الطبقة الإضافية المضافة.

**\_ أنواع الاتحاد : -**

توسع أنواع الاتحاد إعلانات النوع (أنواع الإرجاع والمعلمات وخصائص الفئة) لإعلان أكثر من نوع واحد.

function parse\_value(string|int|float): string|null {}

كما أنه يدعم false كنوع خاص (للمنطقي false)، وهي سمة سائدة في الكود القديم الذي لم يستخدم الاستثناءات.

**\_ عامل آمن للصفر : -**

يوفر عامل الأمان Null الأمان في سلسلة الطريقة/ الخاصية عندما يمكن أن تكون قيمة الإرجاع أو الخاصية null.

return $user->getAddress()?->getCountry()?->isoCode;

يقوم عامل الأمان الصفري ?-> بإغلاق بقية التعبيرات إذا واجه null قيمة، ويعود على الفور null دون التسبب في أي أخطاء.

**\_ تعبيرات match : -**

تتشابه تعبيرات المطابقة مع switch الكتل، ولكن  matchالكتل توفر مقارنات آمنة للنوع، وتدعم قيمة الإرجاع، ولا تتطلب  breakتقسيم البيانات، وتدعم قيم مطابقة متعددة.

كما تضمن مطابقة فرع واحد على الأقل، مما يضمن مراعاة جميع الحالات.

$response = match('test') {

'test' => $this->sendTestAlert(),

'send' => $this->sendNuclearAlert() };

قد لا يتم تحويل جميع  switchالكتل إلى  matchكتل بشكل جيد.

فالكود الذي يتطلب التوافق مع الإصدارات السابقة، أو switch الكتل التي تحتوي على عبارات متعددة (على عكس التعبيرات ذات السطر الواحد)، أو التي تتوقع وظيفة التجاوز، لا يزال يناسب  switchالعبارات.

**\_ خرائط ضعيفة : -**

تسمح لك الخريطة الضعيفة بتخزين قيم عشوائية لمفاتيح الكائنات وربطها، ولكن دون منع جامع القمامة من مسحها، يخرج الكائن من النطاق في أي مكان آخر.

A WeakMap مشابه لـ SplObjectStorage، كما هو الحال في كل من WeakMapو splObjectStorageاستخدم objectss كمفتاح، ويسمح بتخزين قيم عشوائية.

ومع ذلك، لا يمنع a WeakMap الكائن من أن يتم تجميعه في سلة المهملات.

**\_ وظائف وفئات جديدة : -**

يقدم PHP 8.0 بعض الوظائف الجديدة لتسهيل عمليات فحص السلسلة (تحتوي على، وتبدأ بسلسلة فرعية، أو تنتهي بسلسلة فرعية) لاستبدال strpos() !== false المكالمات الدقيقة التي تكون أقل قابلية للقراءة، وأكثر عرضة للخطأ بسبب مقارنات النوع الضعيفة.

**Stringable \_واجهة جديدة : -**

Stringableتمت إضافة الواجهة الجديدة تلقائيًا إلى جميع الفئات التي تنفذ \_\_toStringالطريقة، وتلك التي تعلن صراحةً أنها implements Stringable.

باستخدام  Stringableالواجهة، أصبح من السهل الآن إعلان الأنواع بالنسبة string|Stringableللوظائف التي يمكنها قبول/إرجاع السلاسل أو الكائنات باستخدام \_\_toString()طريقة ما.

**PhpToken \_ فئة جديدة من Tokenizer : -**

توفر الفئة الجديدة PhpToken واجهة كائنية التوجه أكثر سلاسة كبديل للوظيفة القديمة المبنية على token\_get\_allالمصفوفة.

يقوم PHP 8.0 بتحسين نظام الكتابة من خلال إضافة أنواع الاتحاد والنوع mixed.

**\_ نوع زائف جديد mixed : -**

يقدم PHP 8.0 mixedالنوع، الذي كان يُستخدم بالفعل على نطاق واسع في تعليقات DocBlock.

function dd(mixed $var): void { var\_dump($var) };

mixedيمكن استخدام النوع للإشارة إلى أنه يقبل أي نوع، أو يمكنه إرجاع أي نوع. في سياق الفئة/الواجهة، mixedيعمل النوع وفقًا لنفس قواعد مبدأ استبدال Liskov .

**\_نوع الإرجاع لأساليب الفصل static : -**

Static نوع الإرجاع، الذي يتم دعمه بالفعل كنوع إرجاع DocBlock، أصبح الآن مدعوماً في PHP 8.0. staticيعلن نوع الإرجاع أن كائنًا من الفئة التي تم استدعاؤها سيتم إرجاعها.

class Foo {

public static function getInstance(): static {

return new static()}};

**\_ تحسينات معالجة الأخطاء**

التغيير الرئيسي وغير المتوافق مع الإصدارات السابقة في PHP هو أن الوظائف الداخلية الآن تلقي استثناءات على أخطاء النوع أو أخطاء القيمة.

يصحح هذا السلوك التاريخي لـ PHP المتمثل في إصدار تحذير والعودة nullعندما يواجه قيمة لا يمكنه استخدامها. غالبًا ما يكون هذا السلوك غير مرغوب فيه لأن تحذيرات PHP لا توقف تنفيذ الكتلة المتبقية.

**\_ تحذيرات الوظيفة الداخلية الآن TypeErrorتطرح ValueErrorاستثناءات : -**

تفرض الآن جميع وظائف PHP الداخلية تقريبًا فحص النوع ، وبدلاً من التحذيرات، يطرح PHP الآن استثناءات TypeErrorValueError . في قواعد التعليمات البرمجية القديمة، قد يتسبب هذا التغيير في حدوث مشكلات، الآن بعد أن أصبحت الأخطاء أكثر جرأة ووضوحاً.

**\_في التعبيرات - : throw**

قبل PHP 8.0، لم يكن من الممكن throw استثناء تعبير ما (على سبيل المثال عبارة ثلاثية ). أصبح هذا مسموحًا به الآن في PHP 8.0.

value = isset($\_GET['value']) ? $\_GET['value'] : $

throw new \InvalidArgumentException('value not set');

**\_ استثناءات حسب النوع فقط catch : -**

من الممكن وجود catchاستثناءات حسب نوعها، دون التقاط كائن الاستثناء.

try {} catch(TypeError) { // Did not catch the $exception object }

**\_ تم تعيين إعداد تقرير الخطأ الافتراضي علىE\_ALL : -**

الإعداد الافتراضي لـ PHP 8.0 هو إظهار جميع رسائل الخطأ افتراضيًا. تم تكوينه لإخفاء التحذيرات القديمة والصارمة في الإصدارات الأقدم.

**\_ يتم عرض أخطاء بدء تشغيل PHP بشكل افتراضي : -**

يعرض PHP الآن أخطاء بدء التشغيل (فشل في تحميل الامتدادات الديناميكية، تكوينات INI غير صالحة، وما إلى ذلك) بشكل افتراضي.

**\_ التأكيدات تطرح استثناءات بشكل افتراضي : -**

تطرح تأكيدات PHP ( assert()) الآن استثناءات عند فشل التأكيد. قبل PHP 8.0، كانت تتطلب تكوينًا صريحًا تم تعطيله افتراضيًا.

**\_ لا يقوم مشغل قمع الأخطاء بإخفاء الأخطاء القاتلة @: -**

يقوم PHP 8.0 بتصحيح سلوك @عامل قمع الأخطاء، حيث يقوم بإسكات الأخطاء القاتلة، التي تؤدي إلى فشل البرنامج النصي، لأن @العامل لا يمنع الأخطاء القاتلة، بل يخفي عرض رسالة الخطأ.

كان أحد الجهود الطويلة الأمد في تطوير PHP هو الابتعاد عن resourceالأنواع . كان التعامل معها صعبًا، وحتى في PHP 8.0، لا تقدم دعمًا للأنواع.

resourceلا تعمل كائنات PHP بشكل جيد مع Garbage-Collector أيضًا، مما يؤدي إلى حدوث تسريبات للذاكرة في resourceكائنات مثل xml.

في PHP 8.0، تنتقل بعض تغييرات الامتدادات الأكثر استخدامًا من resourceالكائنات التقليدية إلى فئات PHP القياسية.

في PHP 8.0، تعمل هذه الفئات ككائنات ذات قيمة على عكس الفئات ذات الميزات الكاملة التي تحتوي على طرق. لا تسمح غالبية هذه الفئات بإنشاء مثيلات باستخدام new Foo()البنية أيضًا، ويجب إنشاؤها باستخدام الوظائف الموجودة التي أرجعت resourceالكائنات في الإصدارات السابقة.

تعتبر عملية هجرة PHP 8.0 resourceإلى الكائنات سلسة للغاية، كما هو الحال في جميع الوظائف التي ترجع وتقبل الكائنات الجديدة، وتتصرف بنفس دلالات الكائنات السابقة resource.

**\_ تغييرات البرمجة الموجهة للكائنات في PHP : -**

PHP 8.0 هو أول إصدار رئيسي يتسم بالصرامة فيما يتعلق بانتهاكات مبدأ استبدال Liskov. قبل PHP 8.0، لم تكن PHP متسقة في كيفية التعامل مع توقيعات الطريقة غير المتوافقة.

في PHP 8.0، تؤدي جميع حالات عدم تطابق التوقيع، بما في ذلك السمات المجردة، إلى حدوث خطأ فادح. علاوة على ذلك، فإنه يفرض التوقيعات على طرق PHP السحرية.

**\_ أخطاء قاتلة في توقيعات الطريقة غير المتوافقة : -**

يُحدث PHP 8.0 أخطاءً قاتلة عندما لا يتم اتباع مبدأ استبدال Liskov عند توسيع الفئات أو تنفيذ الواجهات.

قبل PHP 8.0، كانت التوقيعات غير المتوافقة تصدر تحذيرًا فقط.

class Foo {

public function process(stdClass $item): array {}

} class SuperFoo extends Foo{

public function process(array $items): array{}

}

**\_ يتم فرض توقيعات طريقة السحر الخاصة بالفئة بشكل صارم : -**

بدءًا من PHP 8.0 والإصدارات الأحدث، يجب على الطرق السحرية (مثل \_\_toString()، \_\_get()، إلخ)، إذا أعلنت عن أنواع، أن تنفذ التوقيع الذي تتوقعه PHP. وهذا لتجنب أدنى فرصة لإعلان المستخدم عن طريقة سحرية لا تتبع المعنى الدلالي.

class Foo {

public function \_\_toString(): object{}}

كانت الإعلانات مثل تلك Foo::\_\_toString(): objectمسموحة في إصدارات PHP السابقة، لكن PHP 8.0 لا يسمح بذلك ويلقي استثناءً إذا كان التوقيع لا يلبي المتطلبات .

**\_ يؤدي استدعاء أساليب الفئة غير الثابتة بشكل ثابت إلى حدوث خطأ فادح**

لم يعد PHP 8.0 يسمح باستدعاء طرق الفئة كطريقة ثابتة.

class Foo {

public function bar{}

}

;()Foo::bar

أصدرت الإصدارات السابقة إشعارًا بالإهمال، ولكن بدءًا من PHP 8.0 والإصدارات الأحدث، يؤدي هذا إلى حدوث خطأ فادح.

**\_ أساليب الفئة Private : -**

يخفف PHP 8.0 من فرض التوقيع abstractو، staticوالعلامة على privateطرق الفئة. يأتي هذا التغيير من المنطق القائل بأن privateالطرق هي فقط خاصة.

ابتداءً من PHP 8.0، أصبح من المسموح الآن للفئات الفرعية أن تعلن عن طرق مجردة، وتغيير العلامات الثابتة privateللطرق.

**\_ يتم الآن دعم الثابت السحري على الكائنات :: class : -**

يعيد الثابت ::classالسحري اسم الفئة المؤهل بالكامل. كان هذا مسموحًا به فقط في أسماء الفئات (مثل Foo\Bar::class)، ولكن في PHP 8.0، ::classيعمل الثابت السحري على الكائنات المتماثلة أيضًا.

**\_ التغييرات المتعلقة بالسلسلة : -**

في PHP 8.0، هناك العديد من التغييرات الدقيقة التي قد لا تكون واضحة في البداية، ولكنها قد تؤدي إلى نتائج غير متوقعة تمامًا.

الفرق الرئيسي في PHP 8.0 هو أن PHP الآن يعتبر أن هناك سلسلة فارغة بين كل حرف في سلسلة معينة.

قبل PHP 8.0، لم يكن من المسموح التحقق من وجود سلسلة فارغة (" ")، ولكن في PHP 8.0، سيقبل PHP ذلك بكل سرور، ويعود بأنه يوجد بالفعل سلسلة فارغة بين كل حرف.

لا تزال معالجة البايتات المتعددة أو الوظائف مثل strlenلا تزال تعيد نفس القيم مثل الإصدارات القديمة، ولكن يتم تغيير جميع الوظائف التي تتحقق من وجود سلسلة فرعية في سلسلة معينة.

بالإضافة إلى ذلك ، يقوم PHP 8.0 بتغيير كيفية تحديد أولويات عامل ربط السلسلة، ويدعم المعدلات الجديدة في sprintfالوظائف مثل %hالمعدلات%H ومعدل

substr, iconv\_substr, grapheme\_substrإرجاع سلسلة فارغة عند الإزاحات خارج الحدود

تقوم هذه الوظائف بربط معلمات الإبرة والإزاحة بطول السلسلة، وإرجاع سلسلة فارغة بدلاً من إرجاع false.

substr('FooBar', 42, 3); // "" in PHP >=8.0, false in PHP < 8.0 mb\_substr('FooBar', 42, 3); // "" in PHP >=8.0, false in PHP < 8.0 iconv\_substr('FooBar', 42, 3); // "" in PHP >=8.0, false in PHP < 8.0 grapheme\_substr('FooBar', 42, 3); // "" in PHP >=8.0, false in PHP < 8.0

+/ -المشغلات لها أولوية أعلى عند استخدامها مع .المشغل concat ( )

عند استخدام عاملي + و- الرياضيين في نفس التعبير مع عامل التجميع ( .)، فإن عاملي + و- يأخذان الأولوية الأعلى. وقد أدى هذا إلى إشعار بالإهمال في إصدارات PHP السابقة للإصدار 8.0، ولكن الآن يحدث ذلك بصمت ووفقًا للتحذير.

echo 35 + 7 . '.' . 0 + 5;

// 42.5 in PHP >= 8.0

// 47 in PHP <= 8.0

لا يعتمد على الموقع الجغرافي  floatفي عملية SubString

عندما floatيتم فرض قيمة على string، لم يعد PHP يستخدم الإعدادات المحلية. قبل PHP 8.0، كان PHP يأخذ في الاعتبار الإعدادات المحلية الحالية (التي ليست آمنة للخيوط) أو إعدادات النظام المحلية عند القيام بذلك. أدى هذا إلى إخراج سلسلة غير متسقة لأن بعض الإعدادات المحلية، وخاصة الأوروبية منها، تستبدل فاصل الآلاف والعلامة العشرية من الإعدادات المحلية الأمريكية. لتحويل الإعدادات المحلية float، stringيمكن استخدام %fالمعدِّل في printfفئة الوظائف .