PHP8の新機能をどう教えるか
～実務での使い方を添えて～

by 古庄道明(がる/gallu)

自己紹介

- 古庄道明と申します
- 「がる(gallu)」というハンドルでふらついております
- 本職は技術者です。現役プログラマーやってます
  - バックエンド系なので、インフラとかDBとか運用とかも一通り
  - 最近はPM業も多いですねぇ
  - 「ヒアリングして技術提案」なんてこともさせていただいてます
- 教育も色々と携わっております(専門学校講師とか)
- 「PHP8技術者認定上級試験」「PHP8技術者認定初級試験」の作成を担当させていただいてます
- コラムとかblogとかeラーニングの教材とかで技術系の文章を書くこともちらほら

今日のお題

- PHP8の新しい機能から、いくつかピックアップしてお話をさせていただきます
  - 機能そのものだけではなく、「使いどころ」や「教え方」についても少し挟み込んでいきます
  - PHP 8.0は試験にも出る内容なので分厚めにお話をさせていただきます
  - 最新のPHP 8.3についても少しお話させていただきます

新機能をどう教えるか?

- ある程度キャリアがあると、新しい機能もいくつかは「あぁ、あの時に便利だっただろうなぁ」という予測が立つかと思います
  - その辺がまぁ「キャリア」とか「経験値」とかなのだろうと
- 一方で、その辺の経験が浅いと「……で、なにに使えるの?」と疑問が出る事も多いかと思います
- 一方では「なので用途と一緒に教える」方法もあるのですが、全ての機能で「こんな時に」がすぐに出るとも限らないと思います
- なのでせめて「実際に動くソースコード」があると、少しは印象に残るのではないか、と思います

- という訳で本日は「ソースコード多め」でお送りします‼
- 画面でコードを見るだけだとわかりにくいかと思うので、GitHubにコードを用意しました
  - PHP Open Textbook、という名前で運用しています(宣伝
- https://github.com/php-engineer-examination/SeminarMaterials
  - の中の「/20231206 PHP8で増えた文法をおさらいしましょうセミナー/古庄登壇資料」に入ってます

- また、いくつかの機能では「実務での使い方」を、実体験に基づいたり想像したりしながら、お話できればと思います

- ……削ったつもりなのですが結構量が増えたので、メリハリを付けてお話できれば、と思います‼

非推奨の機能と下位互換のない変更点

- これから新しく学ぶ人にとってはあんまり旨みのない情報……という訳でも、じつは、ありません
- 「こんな機能があり、そして今は推奨されていない」という情報がないと……
  - 「こんなことやりたい……ググって調べてみた」
  - 古い情報のサイトがひっかかり、非推奨の機能が書いてある
  - 「これ知らない。そうかこうやればいいのか!」(勘違い
- こんな事がおきてしまいます
  - mysql_connect()とかmd5()とかsha1()とか以下略
- なので「使ってはいけない」という知識もまた、大切な知識になります!!

PHP 8.0) 文字列と数値の比較

- この辺は最近の傾向や新機能とも一部かぶってくるのですが、現状のPHPはおそらく「型をしっかり意識する」ほうが、事故りにくいコードが書けると思います
  - 特に「*0 == "not-a-number" が false と見なされる*」変更については、(これを意図的に使っている所は少ないかとは思うのですが)コードによっては大きく注意したほうがよいかもしれません
- この辺りは「細かい規則を教える」より「型をしっかりと意識する」方向で教えたほうが早いし安全かと思います
- 01.php

クラス名と同じ名前のメソッドは、コンストラクタと解釈されなくなりました

- PHP7までは(文句を言われつつも)、クラス名と同じ名前のメソッドは「コンストラクタ」と解釈されていました
  - PHP8からは「何も言わずに放置される」ので気をつけましょう
  - 大分古いコードを「一気にPHP8に上げる」時に問題が出やすいです
- 「古い書き方のまま残っている所」でこの問題がある事があります
  - また「他言語だと"クラス名と同じ名前のメソッドがコンストラクタ"」という仕様も普通にあるので、人によっては疑問を持ちにくいのでその辺も注意しましょう
- 02.php

staticでないメソッドを、staticメソッドとしてコールできる機能が削除

- PHP7までは、Deprecatedなどで怒られつつも「とはいえ呼び出す事はできた」のですが。PHP8からは「Fatal errorになる」という挙動になるので注意しましょう
- 学習途中だと「なぜエラーになるか分からない」で学習が止まってしまう事も予想されるかと思います
- staticに限らず「言語のルールは正しく理解し、素直に使う」事を心がけると綺麗なコードになりやすいと思います
- 03.php

create_function() 関数は削除されました

- 昔は「匿名関数」と書かれていて、「無名関数との違い」などレクチャーをしておりました……が、PHP 7.2で非推奨になった後、PHP8で削除になりました
- こちらも、古いコードや記事には一部残っている事があるようなので注意しましょう
- これに限らず「非推奨になったもの」はこのように「削除される事がある」ので、早めに対応をしておくと安心です
- 04.php

数値キーnを持つ配列は、nが負の値でも次の暗黙のキーは n+1 を使うようになります

- PHP7までは、負の値の後で暗黙のキーを使うと「0から」になりましたが、その挙動が変更になりました
- この挙動も「それを意図的に使っている」コードはあまり見た事がないのですが、「暗黙の挙動を前提としたコード」はこのように「どこかで仕様が変わることもある」ので、注意しておくとよいでしょう
- 05.php

親クラスがないクラスでparentを使うと、致命的なエラーが発生するようになりました

- PHP7までは「そのメソッドが呼ばれるまではとりあえず動いていた」のですが、PHP8からは「コンパイル時にFatal errorが発生する」ようになりました
- あまりないとは思うのですが「使われていなかったために(たまたま)エラーになっていなかったコード」が、PHP8になった時にエラーになる、ケースの一因としてありえるかもしれません
- 06.php

大文字小文字を区別しない定数を定義できる機能が削除されました

- こちらも、PHP 7.3で非推奨だった機能の削除です
- この機能をもし「使っている」場合は、早めに探して潰しておくとよいでしょう
  - なので、「非推奨」のエラーも、開発環境ではちゃんと出るようにしておくとチェックしやすくなるでしょう
- 07.php

#[は、コメントの開始として解釈されなくなりました

- これは、新機能「アトリビュート」の追加によるものです
- シャープ# と角括弧 [の間に、1つスペースがあれば、PHP7までと同様に「1行コメント」と見なされます
  - そもそも # を使うコード、自分の周りではとんと見なくなったのですが、全体としてはどうなんですかね?
- エディタにもよる、と思うのですが、ある程度以上のエディタ(とIDE)なら、ちゃんと見分けて色づけしてくれるので、よく見ると(コメント扱いになっていない事が)わかるかと思います
- 08.php

ビットシフトや加算、減算に対する連結演算子の優先順位が変更されました

- 基本的には「文字列演算子」の優先順位が下がりました
  - https://www.php.net/manual/ja/language.operators.precedence.php
- こういった事も「稀」にあるので、個人的には「異なる演算子がある時は、優先順位を明示的に括弧で書く」ほうが安全であろう、と思っています
- OSSの沢山のコードの静的解析では「問題ない」という調査結果が出たそう、ですが、自分のコードがどうか?は確認をしておいたほうがよいかもしれません
- 09.php

たくさんの警告が Error 例外に変換されるようになりました

- 「今までは(文句を言いつつも)動いていたもの」から「エラーなんだけど出方が変わったもの」まで色々あります
- ただ「今までは文句一つもいわないもの」がいきなりエラーになっていることはないので、やはり開発環境では「エラーの出力レベルを E_ALL(または、-1)」にして、全ての問題を明らかにしたほうがよいでしょう
- 10.php

多くのnoticeが警告に変換されるようになりました

- 今までNoticeだったものがWarningになりました
  - 元々「Noticeなら残ってても良い」ものでもなかったようには思いますが
- いずれにしても「ちゃんと書けば直る」ものなので、この辺は丁寧に書いていきたいところだと思います
- 11.php

オフセットを指定してアクセスするための波括弧のサポートが削除されました

- これも「元々非推奨だった」古い記述の1つです
- 以前は「非推奨ではあるが動いていた」記述ですが、PHP8以降は「Fatal error」になります
- 12.php

三項演算子をネストする場合、明示的に括弧が必要になりました

- 三項演算子は元々「割と物議を醸し出しやすい」演算子ではありました
- その物議の一つに「ネストした時の解決順番」がありました
  - 「他の言語」と「PHP」で解決順番が異なるので、混乱しやすかったように思います
- そのため「そもそも、括弧のないネストした三項演算子」を書けないようにしました
  - 他でも書きましたが「優先順位は明示する」ほうが安全です
- 13.php

数値形式の文字列の扱いが変更されました

- 「先頭が数値の場合」以上に、「先頭が数値ではない場合」の扱いが大きく変わりました
- 以前はWarningだったため、最悪、行の先頭に@(エラー抑止演算子)を付けることで「エラーを回避」している、といった処理がある場合、PHP8以降は「Fatal error」で処理が止まります
- エラーを「場当たりに回避する」と後々困るので、エラー抑止演算子も、あまり乱用しないよう注意しましょう
- 14.php

strpos()等, strstr(), strchr()等の needle 引数は常に文字列として解釈されるようになりました

- 対象の関数名は以下です
  - strpos(), strrpos(), stripos(), strripos(), strstr(), strchr(), strrchr(), stristr()
- 過去、needle引数は「数値を渡すと、ASCIIコードとして解釈する」という仕様がありましたが、これが無くなりました
  - これを「積極的に使っているコード」も比較的少ないような気がしますが
- ここでも「型をちゃんと意識する」事で避けられるものがあります
- 16.php

非推奨)デフォルト値を持つ引数の後に必須の引数が続く

- デフォルト値を持つ引数は「引数の右側に寄せる」必要があります
- しかし以前は「宣言時にはなにも言われなかった」ので、PHP 8.0でDeprecatedが出るようになりました
- 15.php

PHP 8.1) htmlspecialchars()等 のデフォルト値が、 ENT_COMPAT から変更されました

- 対象の関数は以下の通りです
  - htmlspecialchars(), htmlentities(), htmlspecialchars_decode(), html_entity_decode(), get_html_translation_table()
- これらの ENT_COMPAT がデフォルト値になっている引数が ENT_QUOTES | ENT_SUBSTITUTE に変更されました。
  - 元々「ENT_COMPATでは駄目なのでENT_QUOTESの指定は最低限必須」だったので、嬉しい変更になります
- ただこれに慣れた後で古いバージョンに戻った時は、少し気をつけておくとよいでしょう
- 17.php

暗黙の float から int への変換

- 明示的にキャストしている時はよいのですが、暗黙的な変換で「データが切り捨てられる」ような変換になる時に、Deprecatedが発生します
- この辺も「型を意識する」事で防げます
- 18.php

falseな変数への、配列を自動で生成する処理

- マニュアルには以下のように記載されています。
  - **false な変数を自動的に復活させる挙動(Autovivification)**
    Autovivification とは、値を追加しようとする際に配列を自動で生成する処理のことです。スカラー型の値からこのような処理を行うことは禁止されていますが、false だけは例外でした。このバージョンから、この例外も推奨されなくなります。
- この辺りも、出来るだけ「(配列で)初期化してから使う」といった癖を付けると面倒が避けられると思います
- 19.php

PHP 8.2)動的なプロパティの利用

- 「プロパティを特に宣言せず、動的にプロパティを生やす」挙動が非推奨になりました
  - この挙動、typoで簡単に「変なプロパティが生える」ので、個人的には本当に「止めて欲しい」機能でした……
- プロパティを定義するか、或いは __get()等のマジックメソッドを使うなど、で対応をしましょう
  - アトリビュート #[¥AllowDynamicProperties] も使えます
- 20.php

PHP 8.3) 加算子/減算子 に対する変更

- 空文字列や、数値形式でない文字列、そして英数字でない文字列に対して、加算子 (++) を使うことは推奨されなくなりました。
  - たまに、どちらかというと「テストコード」とかそういったちょっとしたコードで文字列へのインクリメントは見かける気がします
  - 新しく関数が用意されているので、そちらを使いましょう
- 21.php

引数を渡さずに get_class()/get_parent_class() をコールする

- 「クラスの内部で呼ぶ時は、引数を省略できる(クラスの内部で object を省略すると、そのクラスの名前を返す)」という仕様がありますが、この仕様に対してDeprecatedが出るようになりました
- 個人的にはここの省略はあまり見かけないのですが($thisを渡す)、省略をよく使う人は気にしておくとよいかもしれません
- 22.php

- 「非推奨の機能と下位互換のない変更点」を確認していきました
- かいつまんだつもりですが、結構な分量になりましたね

- 次は嬉しい楽しい「新機能」です!!

名前付き引数

- 引数の順番もそうなのですが「この引数の意味」を明示する意味も含めて、書いておくと色々とコードを特に「読む」人が楽になると思います
  - コードは、書くより数倍「読まれる」ものである事を意識するとよいと思います
  - ただ、そのため「関数宣言の引数名」はしっかり考えたほうがよいでしょう
- 01.php

アトリビュート

- すでに「動的なプロパティ」の所で使っていましたが。コードに対して「なにがしか指定をする」ような時に使います
  - 「メタデータ(情報)を埋め込む」なんて言い方をします
- 独自のアトリビュートも作成できますが、PHP公式のアトリビュートも出てきているので、これから増えていくんだろうなぁ、と思われます

コンストラクタのプロパティ昇格機能

- サンプルコードで見かける事が増えたように思います
  - 行数減るから、サンプル書くのに最適です(笑
- 「コンストラクタのプロモーション」という言い方もされているので、なんとなく覚えておくと便利かと思います
- 使用頻度は増えると思うので。当面は「最低限、読めるようにする」「書けるとベター」って感じなんじゃないかと思います
  - 「引数の最後のカンマの許容」が地味に有り難いです
    - *引数のリストに付ける最後のカンマも、オプションで許可されるようになりました*
- 02.php

union 型

- 関数の型宣言で、union型が使えるようになりました
  - 複合型、とも書かれます
  - 引数で「型A、または型B」等、or条件で型を指定できます
- これで「最低限」型安全なコードを書く準備が整ったんじゃないか、と思います
- 03.php

match 式

- match 式がサポートされました
  - 「厳密な比較」をしてくれるswitch文のようなものです
    - switchが「文」でmatchが「式」なのもポイントではあります
- 基本とても便利です
  - 個人的には、switchは「比較が緩やか」なので色々躊躇していたので、matchは躊躇なく使えます
- 04.php

nullsafe 演算子

- インスタンスや演算の途中でnullがあった時に、それ以降のチェインされた演算を行わずnullを返してくれます
- コードがものすごくすっきりします
  - この機能になれたあとPHP7以下の環境に戻ると大変にストレスになるので、気をつけましょう(苦笑
- 05.php

オブジェクトのクラス名を $object::class と書くことで取得できるようになりました

- 06.php

throw が式として使えるようになりました

- これによって「文の一部に埋め込める」ので、シンプルにコードを書くことができるようになりました
- 07.php

str_contains(), str_starts_with(), str_ends_with() が追加されました

- 文字列比較用の関数です
- 戻り値がboolなので大変に使いやすいので、PHP8以降であれば積極的に使いたい関数です
- 08.php

get_debug_type() 関数が追加されました

- 近い関数として gettype() がありますが、get_debug_type()は色々と改良されて使いやすくなっていると思います
- 特にインスタンスが「クラス名を出力」してくれるので、ちょっとした用途で非常に便利に使えると思います
- 08-2.php

PHP 8.1) 8進数の整数リテラルのプレフィックス指定

- 8進数の整数値の書き方が増えました
  - ゼロオー(0o 、0O)が増えました
- 09.php

文字列をキーとして持つ配列のアンパック

- 文字キーの配列のアンパックが出来るようになりました
  - 「数値キー配列のアンパック」はPHP 7.4で導入されました
- 「配列のアンパック」の挙動は array_merge() に準じますので、文字キーは上書き、数値キーは振り直されます
- 10.php

引数を展開した後の名前付き引数

- 関数(やメソッド)を呼ぶ時に、配列を引数として展開することが、これはPHP8からできます
- このとき「一部を配列からの展開で、残りを名前付き引数で」という風に書く事が出来るようになりました
- 11.php

列挙型(Enum)

- 一部(多数?)の人にとって、待望しすぎて待ちすぎて首がろくろっ首になってしまうくらい垂涎の機能かと思います
  - ググると、先人の苦労と苦心の跡が山盛りに見つかります
- 色々と仕様があるので、まず「簡単な使い方」程度のサンプルコードを置いておきます
- 12.php

第一級callableを生成する記法

- 無名関数を簡単に作成できます
- これも記述が簡単になるので、(慣れると)読みやすいコードになると思います
- この手の機能は、共通ライブラリなどを作る時にちょいちょい便利に役立ちました
- 13.php

Never 型

- 大分とニッチな型ですが、「exitでコードが止まる、または例外が投げられる」のみの挙動をする時用の戻り値の型指定です
- 14.php

読み取り専用プロパティ

- プロパティに対して「読み取り専用(上書き禁止)」の指定をすることが出来るようになりました
- privateにして「コンストラクタ以外から書き込みのルートを潰す」と、外からは似たような事ができますが、readonlyは「中からも上書きが出来ない」ようにする事ができます
- 15.php

PHP 8.2)バックトレースから機密情報を削除するためのアトリビュート

- 機密情報をよりしっかり守る手段として
  #[¥SensitiveParameter] が追加されました
- 最近は、スタックトレースなどを外部に記録する事も増えたかと思うので、こういった機能を知っているとよいでしょう
- 16.php

型システムの改善

- null と false が独立した型として使えるようになりました
- true 型が追加されました
- いずれも「使うシーンはそこそこニッチ」かとは思うのですが「可能である」事は武器が増える事なので、覚えておいて損はないかと思います
- 17.php

トレイトで定数

- トレイトで定数を定義することが出来るようになりました
- 18.php

読み取り専用クラス

- 「プロパティに対するreadonly」があったかと思いますが、今度は「クラス全体にreadonly」が出来るようになりました
  - 所謂「イミュータブル」なクラスが作りたい時とかあったので、要所要所、大変便利かと思います
- 19.php

Random 拡張モジュール

- 乱数を「生成エンジン」と「取得のメソッド」に切り分ける事で、エンジンさえ入れ替えれば「お好みとお望みの乱数を使う」事が出来るようになりました
- 乱数は、速度から品質まで色々と要求があるので、インスタンス生成時に簡単に入れ替えられるのは非常に面白いと思います
  - とはいえSecure()がデフォルトにはなるんでしょうが
- 20.php

PHP 8.3

- PHP 8.3はまだ業務では未使用なのですが、いくつかかいつまんで「興味深い」辺りを拾い出してみます
  - ページ数が結構タイトになっているので、本当にわずかですが……

readonly 機能の修正

- 修正のうち「無名クラスも、readonly としてマークできるようになりました」については、用途のイメージがつきます
  - メソッドの戻り値に「無名クラス」というケースがあるかと思いますが、その戻り値のインスタンスをreadonlyにしたい時、とかに便利そうに思いました
- 21.php

クラス定数への動的なアクセス構文

- これ、言われるまで「……出来なかったんだ?」と思ってしまいました
  - 確かに実験してみると、PHP 8.2まではParse errorになります
- 即時に用途が思いつくわけではないのですが、柔軟なコードを書く一助にはなろうかと思います
- 22.php

static 変数の初期化

- 「 static 変数を初期化する際に、任意の式が使えるようになりました」
- これもちょっとした変更ですが、便利そうだと思いました
- 23.php

DateError と DateException の階層に例外とエラーを追加しました

- テスト環境が作成できなくてまだ実験をしていないのですが、以下の例外が大変に気になりました
  - https://www.php.net/manual/ja/class.daterangeerror.php
    DateRangeError クラス
    32ビットプラットフォーム上で、日付オブジェクトが符号付き32ビットの範囲外の日付を表していた場合にスローされます。
- いわゆる2038年問題、これで少し光が見えてくる……といいですよねぇ……

まとめ

- 駆け足、かつかいつまんでですが、如何でしたか?
- 新しい機能は、少なからず「出来る事が増える」「安全性が増す」などのメリットがあるので。無理のない範囲で取り込んでいきたい所です
- そして、PHP8上級試験はこんな風に(もうちょっと山盛りですが)「沢山のコード」が出てきます
  - 自然言語で「誤解を生むかもしれない記述」より、コードを書いたほうが「正確」だし「一目瞭然」だと思いませんか?
- 仕事でもコード読みはあるはずなので。コードを読む事にも「ある程度、慣れる」ようにしておくとよいでしょう !!
- みなさまの技術学習の一助になれば幸いです