Oop magic method

Syntax : __ method name

وهيا مجموعة methods بتتفعل بشكل تلقائى لما يحصل حدث معين مرتبط بيها ودا السحر اللى فيها نبدا نشرحهم واحده واحده

__construct

ودى جايه من construction معناها بناء او بنايه والفكره منها انها بتتنفذ اول ما تعمل object من الclass بتاعك بمعنى اصح هيا ال main method بتاعت الobject وانت بتعمل object تقدر تمررلها arguments لو انت حابب تعمل كدا او تخليها تتصل بالdatabase لو دا class الdatabase مثلا

Syntax:
Public function __construct(){}

__destruct

وزى ما عندنا بناء عندنا هدم وبالنسبه لل object هيا destruct وهيا بتتنفذ بعد اخر امر تم على ال object بمعنى انى لو عملت object ونفذت عليه كذا امر وليكن ضفت قيمه لproperty او نديت على method فا destruct هتتنفذ بعذ اخر عمليه او امر نفذته على ال object مثال عملى انا لو عندى class لل database هعمل construct عشان اول ما اعمل من ال class دا object يتصل بال database وبعد ما اخلص اللى انا عايزه هعمل destruct عشان تقفلى اتصال ال database

Syntax:
Public function __destruct(){}

__call

وهيا بتستخدم لما انت تنادى على method جوا ال object وهيا مش موجوده او not accessible مثلا private او protected فى الحاله دى هيا بيتعملها invoke وبيتنفذ الاوامر اللى انت كاتبها جواها لما يحصل حاجه زى كدا وهيا بتاخد 2 arguments الاول هو ال method او اى اسم براحتك التانى هو ال params ال parameters

Syntax:
Public function __call(){}

__get

ودى بستخدمها لحالات زى ان يحصل ندا ل property مش موجوده جوا ال object او انها تكون not accessible هنا هيا بتتفعل وبتنفذ الامر اللى انا كاتبه فيها لما يحصل حاجه زى كدا مثلا انى اظهر رساله للى بيحاول ينادى على ال property انها مش موجوده او ملوش صلاحيات عليها وهيا بتاخد 1 argument ممكن تسميهname او زى ما تحب الفكره انها بتاخد argument واحد

Syntax:
Public function __get(){}

__set

ودى بستعملها لما اكون بحاول اسند قيمه او set value ل property مش موجوده او مليش access عليها وقتها بتتفعل وبتنفذ الامر اللى كتبته وانا بعملها وهيا بتاخد 2 arguments الاول هو ال name التانى هو ال value

Syntax:

Public function __set(){}

__clone

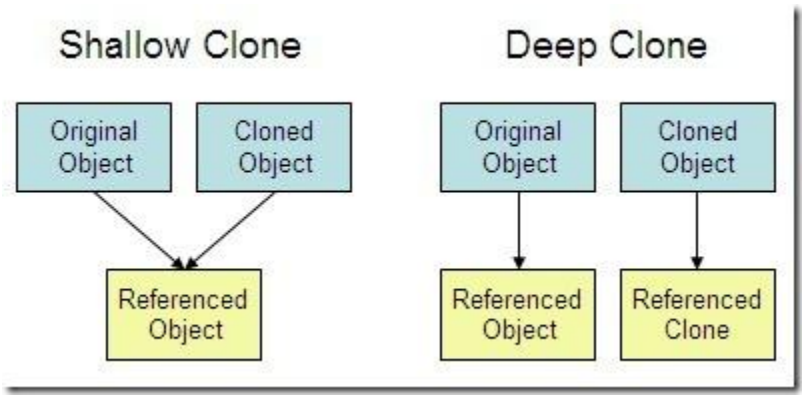
عايز اشرح حاجه بسيطه قبل ما اتكلم عنها وهيا انا عندى لما بعمل استنساخ ل object فى العادى بالشكل دا

```
$obj = new obj();
$obj2 = $obj;
```

اللى بيحصل ان ال object التانى بيبقى نسخه من الاول ولكن نسخه سطحيه بمعنى ان لما يحصل اى تغير على ال object الاول هيحصل على التانى والعكس ودا اسمه shallow copy وهنا النسخ بيبقى by reference بمعنى ان الاتنين بيشاورو على نفس ال address فى الرام فا بالتالى بيتأثرو ببعض طيب دلوقت انا لو عايزهم ميتأثروش ببعض اعمل ايه؟
هنا بقى فى حاجه اسمها clone و هيا keyword بستعملها وانا بعمل النسخ ودى بتخلينى اعرف اعمل حاجه اسمها deep copy وهيا انى بنسخ ال object بشكل كامل وب address جديد فا بالتالى مش بيتأثرو ببعض لو حصل تغير فى اى واحد فيهم بس هنا فى مشلكه النسخ فى ال deep copy بيتم على ال data type المعروفه زى string / int / float / boolean بس انا دلوقت لو عندى ال object بتاعى فى property عباره عن object من class تانى كدا يبقى انا خدت نسخه deep copy بس فى نفس الوقت هيا كمان shallow copy فى ال property دى هنا بقى تيبجى ميزة ال magic method __clone وهيا بيتم نذأها لما بيحصل copy لل object وممكن انت بقى تستعملها ف عمل نسخه من ال property دى جوا ال object الجديد فا كدا ال object الجديد هو نسخه كامله لل object الاساسى وجواه نسخه من ال property اللى كانت فى ال object الاساسى
ومش دى بس الطريقه اللى تقدر تعمل بيها deep copy فى built in function فى php تقدر تخليك تعمل كدا وهيا serialize وبعدين استعمل unserialize او عن طريق انى احولها لكود json بعدين ارجعها لكود php عن طريق json_encode وبعدين استخدم json_decode

Syntax :

Public function __clone(){}



بستخدم ال keyword clone بالشكل دى

```
$obj = new obj();
$obj2 = clone $obj;
```

__tostring

وهيا باينه من اسمها شويه بيحصلها call بشكل تلقائى لما احاول اتعامل مع ال object بتاعى على انه string فى حالة لو انا مش مستعملها هيطلعلى error و الmagic method دى ال output بتاعها لازم يكون string ودا من اول php 7.4 غير كدا هيطلع error بس مع php 8 بقى فى حل للمشكله دى وهيا انك تعطل ال strict type بالشكل دا

```
declare(strict_types = 0);
```

Syntax:

```
Public function __tostring(){}
```

__callstatic

وهيا شبيها ب ال __call magic method الفرق انها بتتعامل مع ال static methods لكن call مع ال methods العاديه

```
Public function __callstatic(){}
```

__invoke

بستخدمها لما تكون هتعامل مع ال object بتاعى او اعمله call على انه function وبتيح ليا انى استعمل ال built in function اللى فى php عليه وهنا ال object بيبقى اسمه functor او function object

Syntax:

```
Public function __invoke(){}
```

__isset

لما بستعمل isset / empty على property مش موجود فى ال object بتاعى او inaccessible ال magic method __isset بتتنفذ

Syntax:

```
Public function __isset(){}
```

__unset

زى isset لما بحاول اعمل unset على property مش موجوده فى ال object او inaccessible بيتعملها call وتنفذ ال action اللى بكتبها فيها

Syntax:

Public function __unset(){}

__debuginfo

ودى بيتعملها call لما بستعمل ال object مع var_dump وهيا خاصه بال debuging

Syntax:

Public function __debuginfo(){}

__set_state

ودى هنا بتستخدم لما تعمل serialize ل object ب var_export فى php ولان var_export ال output بتاعها هو valid php code فا انت مش محتاج تعمل unserialize ليها بمعنى انها مش محتاجه حاجه ترجع بياناتها زى ما كانت لانها مفهومه بالنسبه ل php بس عشان استعمل var_export مع object لازم استعمل magic method __set_state

__sleep / __serilaize

الاتنين شبه بعض وبيقموا بنفس الوظيفه ولكن لو افترضنا انى عندى object فى الاتنين فا الاولوية هتكون ل magic method __serialize اما بالنسبه للوظيفه فا لو تفتكر كلامنا فى ال clone عن ال built in function serialize اننا نقدر نستعملها مع ال object عشان نعمل بيها deep copy اللى بيحصل مع ال magic methods دى ان فى حالة لو استعملت serialize على ال object هيتعملهم call وهما وظيفتهم يحددو الlogic اللى هتقوم عليها العمليه مثلا او انك تحدد ايه جوا ال object هيتعمله serialize ودا يجيبنا لنقطه تانه ايه هيحصل لو انا معملتش return وانا بستعمل serialize اللى هيحصل ان هيرجعلى null وبردو لازم يكون ال output بتاعها عباره عن array بالشكل دا

Syntax:

Public function __serialize(){

Return [‘property name’ => ‘property value’];

}

__unserialize / __wakeup

طالما عملنا serialize يبقى هنعمل unserialize كلام منطقى وسليم وبنفس الشكل زى sleep / serialize بردو الأولوية هنا بين wakeup و unserialize بتكون ل unserialize والاتنين بيعملوا unserialize للobject اللى بيحصل هنا ان بعد عمل unserialize الobject بيبقى نسخه deep copy من ال object الاساسى ولو ال class بتاع ال object دا مش موجود هيعمله وهيبقى php_incomplete_class

نبذه عن ال serialize وهيا انى بحول ال object ل string عباره عن byte stream لبيانات ال object

Syntax __unserialize(){}

ودا لينك بيشرحهم مع امثله عليهم

https://www.geeksforgeeks.org/what-are-magic-methods-and-how-to-use-them-in-php/

$this vs self vs static

نبدا ب self

1. بتشير لل class اللى انا فيه
2. ليها access على ال static members وتقدر تستعملها بشكل مباشر من ال class منغير ما اضطر اعمل object
3. مش بتسعمل معاها ($) لانها بتتعامل مع static data
4. بستخدم معاها ال resolution scope وهيا عباره عن (::)

تانى حاجه معانا هيا $this

1. بتشير لل object اللى انا فيه
2. بتتعامل مع ال non static object members
3. بستعمل معاها ($) واللى بتوصف انك بتتعامل مع متفير

واخر حاجه هيا مقارنه بين static و self الاتنين شبه بعض كا syntax الاتنين بيتعاملوا مع ثوابت بس فى اختلافات بسيطه بينهم وهيا انك لما تستعمل self فى ال class الرئيسى كا output مثلا ال ouptu دا هيتورث بنفس القيمه اللى موجود بيها فى ال class الرئيسى الاب ولكن مع static الoutput هيبقى على حسب القيمه بتاعته فى ال class اللى انا فيه بمعنى انى لو عندى method فى ال class الرئيسى زى دى

```
Public function once(){
Return self::name;
}
```

بما انى مستعمل self فا لما انادى ال method دى من اى class هتبقى قيمتها زى ما هيا فى ال class الاب فى اىclass فرعى هيورث منه لكن لو نفس ال method استعملت static اللى هيحصل ان القيمه هتختلف حسب قيمة ال property فى كل class

Functional vs OOP vs Procedural programming

نبدا ب procedural programming

هو نمط برمجى بيعتمد على انى بنفذ الجزء اللى انا عايزه بس من الكود مغير مضطر اشغل البرنامج كله عشان جزء معين والطريقه دى بردو بيبقى اسمها routines / sub routines وهيا انى بستدعى الاجراء اللى انا عايزه بس عشان يتنفذ

وتانى حاجه معانا وهيا functional programming

وهيا نمط برمجى باين من اسمها انها بتعتمد على مفهوم ال function كا الية عمل فيها وال function هيا عباره عن template بياخد معطيات وبيكون جواها اجراءات بتتنفذ وبعدين تعمل return لقيمه معينه وممكن بردو متعملش return وبشكل عام كل اللغات البرمجية بيبقى فيها خواص ال functional programming لانها هيكل اساسى فى لغات البرمجه

واخر حاجه معانا وهيا ال oop programming

وهيا بتعتمد على مفهوم ال object بمعنى الكائن واللى بيفسره اسمها object oriented programming البرمجه كائنية التوجه وهيا فى الحقيقه نمط هدفه مختلف عن النمطين اللى فاتوا هيا هدفها بشكل اكبر انها تساعد فى تنظيم بناء المشروع بشكل جيد عشان ميحصلش اخطاء او لغبطه لو كان مشروع كبير وعشان يبقى سهل بالنسبه للمبرمجين الشغل كفريق بردو لان فى معايير معينه بيمشوا عليها فى بناء المشروع وعشان يبقى سهل انى احدد الاخطاء واعالجها او انى احسن الكود بعدين والفكره انى بتعامل ب 4 قواعد بتكون ال oop وهيا abstraction / inheritance / polymorphism / encapsulation

Abstraction :

وهيا التجريد انى ابص لعناصر المشروع بتاعى بشكل مجرد ابص لل user على انه كائن وال product على انه كائن و ال cart على انها كائن لو انا بعمل مشروع ecommerce مثلا

Polymorphism:

تعدد الاوجه او الاشكال وهيا ببسطه انى عندى مفهوم معين مثلا button ولكن لما بضغط عليه الحدث اللى بيحصل بيختلف الحاجه اللى هو مربوط بيها مثلا لو مربوط بأضاءه فا لما اضغط عليه النور هيشتغل لو مربوط مثلا بريموت لما اضغط عليه هيقلب القنوات مع اه مفهوم واحد انه button

Encapsulation:

العزل وهيا انى اقدر احمى او اعمل عزل الصفات الخاصه بيا او قيمه مش عايز اورثها او ان حد يقدر ينادى عليها او يغيرها ودا بيتهم من خلال ال access modifiers وهما

Private / protected / public

وهما اللى بيحددو اساس التعامل مع ال class members ال private متقدرش تنادى عليها من برا ال class او تغير قيمتها من برا ال class وبردو متقدرش تورثها ل class تانى protected متقدرش تنادى عليها برا الclass او تغير قيمتها من برا ال class ولكن تقدر تورثها public تقدر تورثها وتنادى عليها من برا ال class او تغير قيمتها من برا ال class عادى تقدر تتعامل معايا عادى فى كل حاجه

Inheritance:

التوريث وهيا انى اعرف اورث الخصائص بتاعتى كا كائن زى properties او methods ل class مشتق منى