Magic Method

- **__get ($property)**

لما تحاول توصل لخاصية private/protected **أو مش موجودة ،** PHP **بيعملك مشكلة .**
هنا ييجي دور __get **، ودي بتشتغل أوتوماتيك عشان ترجعلك القيمة أو حتى تقولك الخاصية دي مش موجودة .**

مثال :

```php
class Car {
    private $model = "Toyota";

    public function __get($property) {
        if (property_exists($this, $property)) {
            return $this->$property;
        } else {
            return "Property '$property' does not exist!";
        }
    }
}

$myCar = new Car();
echo $myCar->model; // Output: Toyota
echo $myCar->price; // Output: Property 'price' does not exist!
```

الفكرة؟

- **لو الـ** property **موجودة يرجعها.**
- **لو مش موجودة يقولك كده : " الخاصية مش موجودة " .**

- **__set($property , $value)**

لما تحاول تضيف قيمة لخاصية مش موجودة أو محمية (private/protected) ، PHP هيزعل .
هنا الميثود دي بتدخل وتهدي اللعب وتتحكم في اللي يحصل .

مثال :

```php

class Car {
    private $model;

    public function __set($property, $value) {
        if (property_exists($this, $property)) {
            $this->$property = $value;
        } else {
            echo "Can't set value to '$property
' - Property doesn't exist!<br>";
        }
    }
}

$myCar = new Car();
$myCar->model = "Honda"; // __set is called
echo $myCar->model; // Output: Honda
$myCar->color = "Red";
// Output: Can't set value to 'color' - Property doesn't exist!

```

الفكرة؟

- **تحمي الكود من التعديل على حاجات مش موجودة أو محمية .**

- **__call ($method , $arguments)**

لما تستدعي دالة مش موجودة، PHP هيرفض.
الميثود دي بتديك الفرصة تتحكم في الموضوع.

مثال :

```php

class Calculator {
    public function __call($method, $arguments) {
        if ($method == "add") {
            return array_sum($arguments);
        }
        return "Method '$method' not found!";
    }
}

$calc = new Calculator();
echo $calc->add(5, 10, 15); // Output: 30
echo $calc->subtract(10, 5); // Output: Method 'subtract' not found!

```

الفكرة؟

- تتعامل مع الدوال المفقودة بشكل مرتب بدل ما الكود يكسر.

- **__callStatic ($method , $arguments)**

زي __call بالظبط ، بس مع الدوال الثابتة (Static Methods).

مثال :

```php

class Calculator {
    public static function __callStatic($method, $arguments) {
        if ($method == "multiply") {
            return array_product($arguments);
        }
        return "Static method '$method' not found!";
    }
}

echo Calculator::multiply(2, 3, 4); // Output: 24
echo Calculator::divide(10, 2);
// Output: Static method 'divide' not found!

```

الفكرة؟

- **الميثود __callStatic بتخليك تتحكم في أي استدعاء لدالة ثابتة مش موجودة بدل ما الكود يكسر .**

بالتالي:

- **الكود مش هيكسر.**
- **تقدر تضيف وظيفة مخصصة زي تنفيذ عمليات معينة أو عرض رسائل مفهومة.**

- **__clone()**

لما تعمل نسخة جديدة من object **باستخدام** clone، PHP **بينفذ الكود جوة الميثود دي.**

مثال :

```php
class Car {
    public $model;

    public function __clone() {
        $this->model = "Cloned " . $this->model;
    }
}

$car1 = new Car();
$car1->model = "Toyota";

$car2 = clone $car1;
echo $car2->model; // Output: Cloned Toyota
```

الفكرة؟

- __clone **بتستخدم لتعديل أو تهيئة خصائص الكائن الجديد أثناء عملية النسخ باستخدام** clone.
- **في المثال، الكود يضيف كلمة** "Cloned" **قبل اسم النموذج** (model) **للكائن الجديد بشكل تلقائي.**

- **__toString()**

لما تحاول تطبع الكائن على إنه نص، PHP هيطلب تمثيل نصي للكائن ، وده اللي بتعمله __toString.

مثال :

```php

class Car {
    private $model;

    public function __construct($model) {
        $this->model = $model;
    }

    public function __toString() {
        return "Car Model: " . $this->model;
    }
}

$myCar = new Car("Toyota");
echo $myCar; // Output: Car Model: Toyota

```

الفكرة؟

- **تحول الكائن لحاجة مفهومة لما تحاول تطبعه.**

- **_invoke()**

لما تستخدم الكائن نفسه كأنه دالة ، PHP بينفذ الكود اللي جوة الميثود دي .

مثال :

```php
class Printer {
    public function __invoke($message) {
        echo "Printing: $message";
    }
}

$printer = new Printer();
$printer("Hello, World!"); // Output: Printing: Hello, World!
```

الفكرة؟

- **تستخدم الكائن كأنه** Callable.